2ej
7

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

"LA PROBLEMATICA DE LA ANIMACION POR COMPUTADORA"

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

M A T E M A T I C A

P R E S E N T A

CARMEN ARMINDA HERNANDEZ GALLEGOS

MEXICO, D.F. 1987

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

CAPITULO III. SISTEMAS DE ANIMACION DE ALTA TECNICA

CAPITULO IV. ANIMACION EN MICROCOMPUTADORAS

I N T R O D U C C I O N

Convencionalmente las figuras se han animado mediante el trabajo de muchos hombres que dibujan, corrigen y filman las imágenes una a una,para simular el movimiento.

Este proceso puede llevar varios meses de trabajo de muchas personas y es por esto que se ha planteado que la computadora puede ser una alternativa para agilizar el proceso de animación.

La animación por computadora se ha venido desarrollando desde 1960.

Este documento contiene una concentración de la información, asi como un análisis y consideraciones acerca del tema de animación por computadora.

Hasta ahora se pueden encontrar algunos articulos, libros y revistas que hablan del tema en alguno de sus aspectos, pero se considera que es necesario contar con un documento que explique clara y detalladamente los siguientes puntos:

- Cuáles son los tipos de animación por computadora que existen.
- Cuáles son los requerimientos de equipo y programas necesarios para producir animación.
- Cuáles son las caracteristicas de los programas existentes para la animación.
- Qué experiencias han tenido las personas que han trabajado en el area.
- Cuáles son las limitaciones de la animación por computadora.

En base a estas interrogantes, se planteó el objetivo de este trabajo el cual puede resumirse como: Definir con certeza cuales son los principales problemas, limitaciones y alcances de la animación por computadora.

Como ya se planteó anteriormente se contaba con la información dispersa, sin orden, ni análisis, simplemente eran resultados obtenidos en casos particulares. Es por esto que se hizo necesario hacer una recopilación y descripción formal de estos resultados con el fin de apoyar a todas las personas que estén interesadas en producir trabajos en el area.

La metodologia que se siguió para el desarrollo de esta tesis, consistió principalemnte en la investigación bibliográfica de diversas fuentes, que se mencionan al final del trabajo.

En segundo lugar se hizo una investigación de campo en México para localizar a las personas o instituciones que están trabajando en el area y finalmente se implementaron algunos programas muy simples en una microcomputadora compatible.

El trabajo consta de cuatro capitulos organizados de la siguiente manera :

El primer capitulo trata sobre la historia de la animación y la forma como se han elaborado las peliculas animadas convencionalmente, es decir sin el uso de la computadora.

El segundo capitulo presenta la descripción del equipo necesario para hacer animación, como son los dispositivos de entrada y salida. Plantea también algunos problemas para la animación en tiempo real que se deben principalmente a la velocidad de acceso a memoria y a la velocidad de procesamiento de los dispositivos actuales.

El tercer capitulo contiene las descripciones de los principales sistemas de animación que ya se han implementado. Presenta también un panorama del equipo y los programas utilizado en cada sistema asi como sus problemas y limitaciones .

Por último el cuarto capitulo está enfocado a la animación en microcomputadoras. En los últimos años, éstas máquinas han invadido el mercado actual, por lo que es importante determinar sus posibilidades en los diferentes niveles de la animación.

Estamos seguros de que el presente estudio será de utilidad para las personas que proyecten realizar animación en una computadora, ya que les apoyará en la definición de aspectos importantes acerca del equipo y los programas necesarios, y de las limitaciones que se tendrán en cada caso.

A mis padres Carmen y Armando
Por su amor, por lo que fui, por lo que soy, por lo que seré

A mis hermanos
Rosalva, Alejandra, Susana y Roberto ,por permitirme llamarlos asi.

A Marco Antonio
Por llenar mi vida de amor.

A Roberto Maldonado
Debi juzgarlo por lo hechos y no por las palabras. Gracias por esos hechos.

A Ana Maria Gonzalez
Por su cooperación, su confianza y su amistad.

A mis compañeros de doblaje
Por tantos años de ayuda en mi educación.

A las familias Gallegos, Hernández y Medina
Por estar siempre unidos por los buenos y malos momentos.

A Ana Luisa
Por aceptar dirigir este trabajo y ayudarme en todo momento.

A los profesores integrantes del jurado
Por su participación en la revisión de esta tesis y por sus enseñanzas.

Gracias nuevamente

Arminda

CAPITULO I.
ANIMACION CONVENCIONAL

El concepto de animación se introdujo en la cultura desde el siglo pasado. El hombre observando los fenómenos cotidianos, encontró la forma de "dar vida" a los objetos inanimados. Sus técnicas han ido evolucionando a través del tiempo desde las máquinas primitivas hasta las fabulosas películas actuales en donde se observan imágenes extraordinarias.

En esta sección se presenta un breve esbozo del desarrollo de las técnicas de animación, describiendo como se efectúa este proceso normalmente.

1.1 QUE ES ANIMACION ??

Se pueden encontrar varias definiciones de este término :

1. Dar movimiento, calor y vida a algo.
2. Infundir vigor a cosas inanimadas.
3. El arte de dar vida o espíritu.
4. Simular vida artificial en imágenes que se mueven en películas o en pantallas de computadoras.
5. Sucesión o secuencia de dibujos cada cual un poco diferente del anterior, cuando pasan por un proyector o por una pantalla .
6. Proceso de creación de imágenes que parecen moverse, aunque en realidad no sea así.

La animación es una técnica relativamente antigua, muy popular pero que muy pocas personas comprenden realmente. El impacto que tiene en nuestros días se refleja en nuestra propia casa cuando encendemos el televisor o cuando vamos al cine.

Pero por qué es posible que el ojo humano vea figuras animadas cuando en realidad no se mueven?

1.2 PERSISTENCIA DE VISION

Cuando las cosas se mueven mas rápido de una velocidad entre 18 y

24 veces por segundo, ocurre un fenómeno llamado persistencia de visión en el cual el ojo retiene momentáneamente la imágen anterior encimándola con la imagen actual.

Una imagen es retenida por el cerebro y registrada por la retina. Si entonces aparece una segunda imagen en un mínimo de tiempo (50 milisegundos), el cerebro, que aun tiene la imagen anterior, mezcla ambas.

Cuando las imágenes se presentan con diferencia de pequeños intervalos de tiempo , se causa un efecto de movimiento continuo. Esta ilusión óptica es el fundamento del cine actual.

La persistencia de visión depende de la intensidad de la luz. En iluminaciones fuertes, es de 1/48 de segundo y en débiles de 1/20 de segundo.

El número de imágenes presentadas por segundo determina la velocidad de centelleo o parpadeo. Eventualmente cuando el tiempo entre un cuadro y otro es muy grande, el ojo puede diferenciar cada uno de los cuadros. También ocurre cuando la diferencia de posición de una figura a otra es muy grande.

En las peliculas estandar de 35 milimetros, se trabaja con una velocidad de 24 cuadros por segundo (a esta velocidad no se percibe el parpadeo). En una cámara de 8 mm. a 18 cuadros por segundo, se empieza a notar parpadeo.

La velocidad a la cual los objetos aparecen en movimiento es una función del número de dibujos usados y la distancia entre la posición de los objetos en los cuadros vecinos.

1.3 BREVE HISTORIA DE LA ANIMACION

La primera máquina de animación fue el llamado "Thaumatrope" que fue inventado por el inglés Dr. John Paris a mediados de 1820.

Por medio de una cuerda se hacia girar un disco que tenia una foto en cada lado. Cuando el disco giraba, se podian ver dos fotos al mismo tiempo, dando la apariencia de movimiento.

Alrededor de 1832 Joseph Plateau inventó el "Phenakistocospe" o rueda animada, constituida por un disco con diferentes dibujos radiales representando fases de un determinado movimiento y otro disco colocado al frente el cual tenia varias rendijas. Al colocarse el disco móvil frente a un espejo y mirando a través de las rendijas se ofrece la ilusión del movimiento.

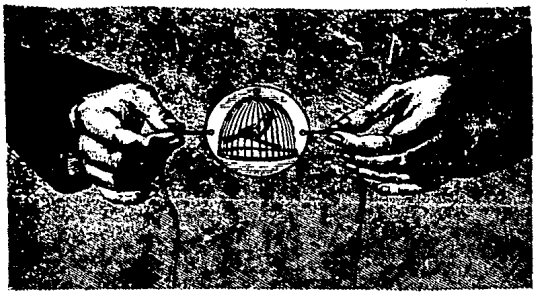


Fig 1.1 El primer dispositivo de animación el "Thaumatrope"

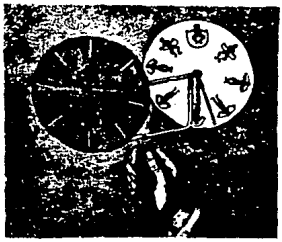


Fig 1.2 El "Phenakistoscope"

Las rendijas funcionaban como el obturador de un proyector de cine permitiéndole ver cada marco por solo una fracción de segundo.

En 1833 William Horner en Inglaterra formó con los dibujos una cinta cilíndrica que girara en determinado sentido alrededor de un eje y cuyo movimiento era observado a través de las rendijas de otro cilindro concéntrico girando en sentido contrario. A este aparato se el dio el nombre de disco del diablo o Zoetrope.

Fig. 1.3 El Zoetrope o disco del diablo

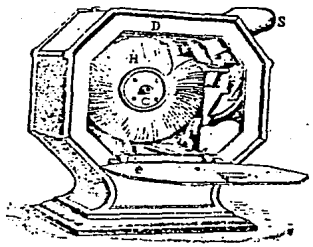
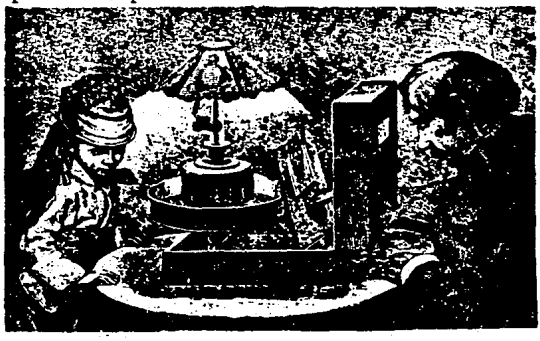


Fig. 1.4 El Mutoscopio

Este aparato fue rediseñado en Francia por Pierre Desvignes en 1860. Esta nueva versión constaba de un cilindro giratorio con imágenes en su interior. Al igual que el anterior, el tambor tenia incisiones en sus lados.

En 1866 apareció el mutoscopio en que la serie de dibujos estaban colocados en unas tarjetas , que se observaban por un tubo.

En 1877 Reynaud ideó el Praxinoscopio que vino a reemplazar al Zoetrope, en el que una serie de dibujos en las facetas planas de un tronco de pirámide se reflejaban en las facetas exteriores de un tronco de prisma. Los dos troncos giran en sentidos contrarios. Reynaud llegó a comercializar el aparato y abrió el primer "teatro en movimiento" en Paris en 1892. El espectáculo duraba muy poco tiempo.



En el año de 1877 Edweard Muybridge deseaba fotografiar un caballo que galopaba. Ingeniosamente conectó los disparadores de varias cámaras a una cadena, la cual estaba sobre la pista por donde corria el caballo, el cual accionaba los disparadores con sus patas.

Posteriormente Muybridge desarrolló el Zoopraxiscopio para proyectar sus estudios sobre movimiento en una pantalla.

Utilizó ruedas de vidrio con las imágenes en las orillas. El disco giraba en un proyector que mostraba el movimiento. Un ciclo completo duraba cerca de medio segundo.
El procedimiento fue perfeccionado por Anschutz (1885) en la obturación mecánica, pero el que puede considerarse como inventor del cinematógrafo es Marey, el cual construyó un fusil fotográfico para tomar una sucesión de vistas del vuelo de un ave.
Marey tomaba las vistas primero sobre una placa fotográfica y después sobre una cinta de papel recubierto de una capa sensible.

En 1888 Marey construyó el primer cinematógrafo . En 1889 Friese Green introdujo la pelicula de celuloide. En 1890 los hermanos Lumiere empezaron su popularización y le dieron nombre.

Otro francés, Emile Cohl fue pionero de los dibujos animados. En 1908 realizó dibujos en color negro sobre hojas de papel y los fotografió.En la pantalla se utilizaban los negativos para mostrar figuras blancas que se movian en un fondo negro.

En los siguientes 5 años se produjeron nuevas figuras animadas incluyendo Gertie "El dinosaurio amaestrado" (1909) y en 1917 el memorable personaje, Felix el gato.

Algunas de las técnicas utilizadas fueron :

- Peliculas de siluetas. Figuras negras en fondos blancos. Estas figuras eran fáciles de dibujar.

- Animación de fase. En este enfoque las figuras fueron superpuestas en un primer plano.

- Animación en celuloide. Se utilizaron transparencias de celuloide para el primer plano y una superposición de éstas sobre un fondo opaco.

A principios de 1920 el trabajo de dibujar los fondos se separó de la tarea de dibujar las figuras.

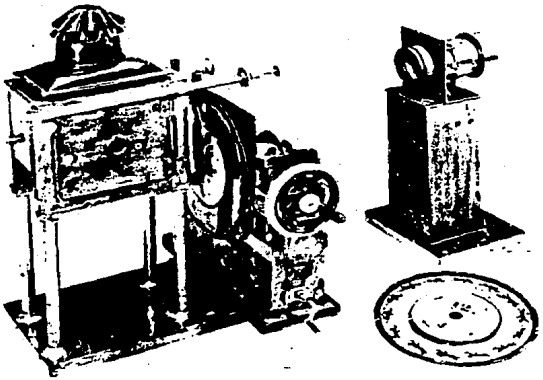


Fig 1.6 El zoopraxiscopio de Muybridge

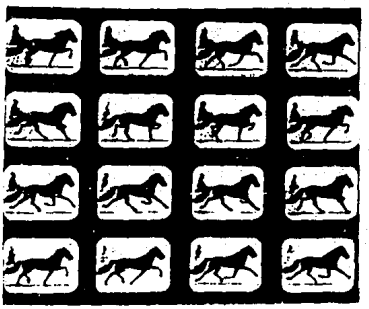


Fig 1.7 Los caballos de Muybridge

Personas expertas en dibujo de fondos los perfeccionaban y los expertos en animación colocaban sus figuras encima.

En 1928 los estudios de Walt Disney empezaron a producir sus famosos cartones animados. Algunas de estas famosas figuras son "Popeye" (1933), El ratón miguelito, "Pinocchio", "Dumbo", El pato Donald, "Tom y Jerry" y "Bugs Bunny" entre otros.

En 1960 dos científicos de los laboratorios Bell desarrollaron la primera animación por computadora. Estas fueron las primeras etapas de la animación de alta técnica en computadora.

Desde 1970 la animación por computadora ha tenido un gran desarrollo y se han descubierto nuevas técnicas para manipulación de figuras.

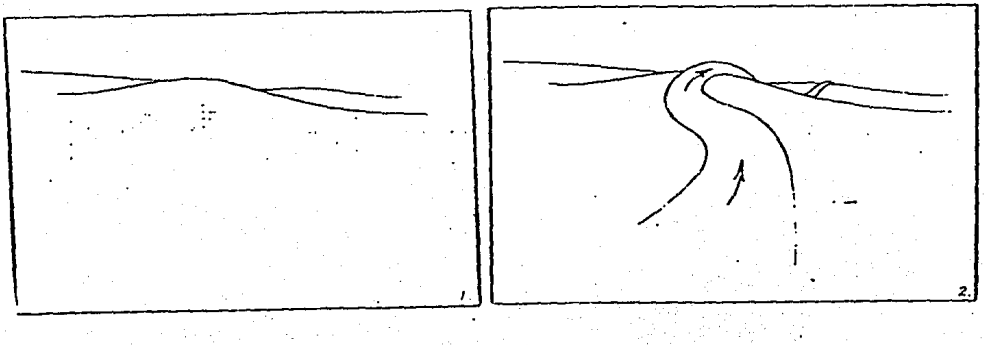
1.4 PROCESO PARA LA PRODUCCION DE UNA PELICULA ANIMADA.

Este proceso empieza con la preparación de un libreto, para los comerciales animados. El libreto se desarrolla normalmente en capitulos ilustrados que se presentan de una manera cómica. Este texto es llamado historieta (en Inglès, storyboard). Un paso muy importante en la producción de la pelicula es la grabación de la pista de sonido que complementa las imàgenes visuales.

Cuando el sonido se graba posteriormente al proceso de fotografiado, se le denomina sincronia posterior. Esto causa una serie de problemas para sincronizar la fotografia con el sonido, provocando un gran trabajo de edición. Siempre es preferible grabar primero sonido y luego la imagen. De cualquier forma ambos elementos deberán integrarse.

La lectora de sonido como su nombre lo indica es un instrumento que analiza el contenido de la pista de sonido y el sincronizador es un aparato que mide la longitud de la pelicula.

Con la ayuda de estos instrumentos se analizan las palabras y silabas que contiene y se obtienen las hojas guia que son las hojas que dirigirán todas las fases de producción indicando el número exacto de cuadros que se ocupan en cada acción, y el tiempo para cada palabra en el diálogo grabado.



Fondo

Ruta de acción

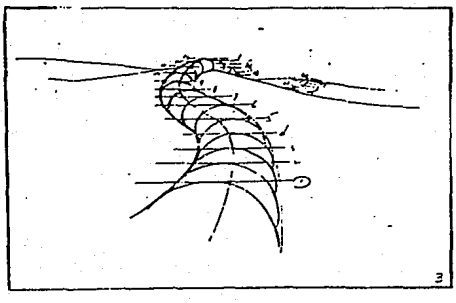
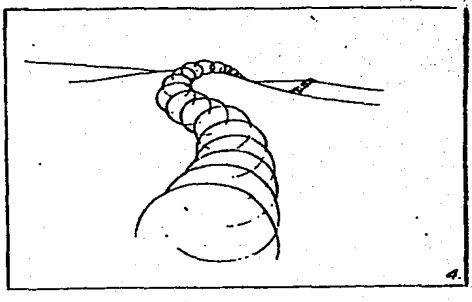


Diagrama de espaciamiento

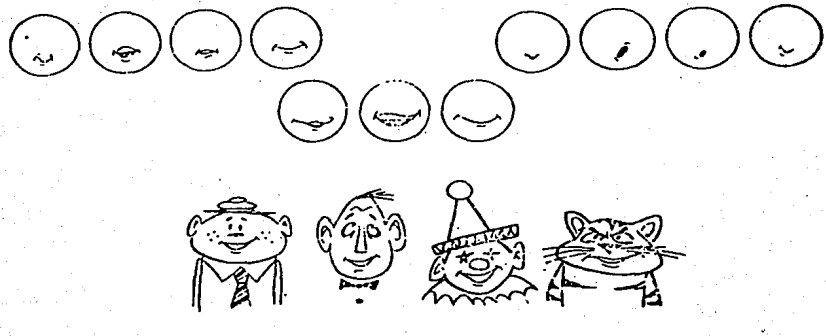


Posiciones para la animación

Fig 1.8 Etapas donde se calcula cada cuadro para realizar la animación

La sincronia del sonido con el movimiento de la boca es muy importante ya que la atención del público está centrada en los rostros durante los diálogos.

El animador no dibuja todos los cuadros que corresponden a una acción, solamente los dos extremos. Los dibujantes tienen instrucciones especiales que los guian para la elaboración de los cuadros intermedios necesarios para completar la acción.



1.9 Diferentes expresiones para las figuras

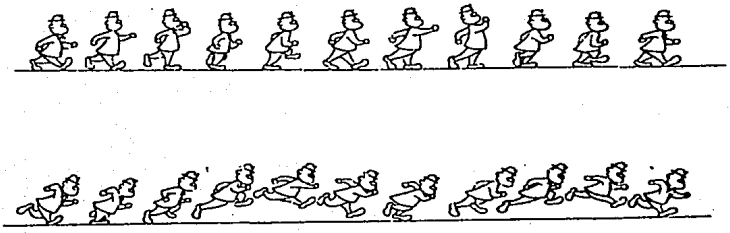


Fig 1.10 Secuencias de animación

Un método para verificar la apariencia de la secuencia es fotografiando los dibujos mediante una cámara suspendida sobre una superficie con un motor. El proceso de fotografiado de los dibujos se denomina prueba de lápiz.

Posteriormente la película se ve en una "moviola", en la cual el sonido y la imagen avanzan paralelamente.

En la siguiente fase de la producción, los dibujos se trazan en acetatos de 8 1/2 " x 11 ". Este proceso se efectúa con tintas de varios colores. Una vez terminado se aplican colores de agua opacos en la parte trasera del acetato, lo cual previene contra posibles corrimientos de la tinta y elimina las marcas del pincel. Otras personas especialistas en la materia elaboran los fondos sobre los cuales se colocan los dibujos anteriores. Se tiene que revisar que los fondos coincidan con los dibujos.

Finalmente, el camarógrafo coloca cada uno de los acetatos sobre el fondo. En este punto se dan los efectos de acercamiento, alejamiento y cambio de escena. Esta película se envía al laboratorio para su procesamiento.

Si la animación va a combinarse con secuencias de la vida real como en algunos comerciales, si se van a colocar títulos o se necesitan efectos especiales, es necesario utilizar un impresor óptico, que es un dispositivo de proyección que opera simultaneamente con la cámara y puede fotografiar partes de la película para producir los efectos especiales.

Si ya no hay más correcciones el editor de la película sincroniza la imagen y el sonido. En la etapa final se mezclan la música y los efectos.

Algunos animadores son verdaderos artistas pudiendo mencionar a Walt Disney, Max Fleischer, John Bray, Paul Terry y Winsor Mckay por nombrar algunos. Es necesario mencionar a John Oxberry el cual empezó su carrera como caricaturista y fue pionero en el

desarrollo de equipo técnico que ayudó a esta industria a salir de sus fases iniciales.

1.5 ANIMACION POR COMPUTADORA

Anteriormente a la introducción de las computadoras, las peliculas animadas se realizaban mediante el trabajo de varios dibujantes, los cuales pintaban a mano cada uno de los cuadros. Haciendo un cálculo de la cantidad de cuadros necesarios para una pelicula de 2 horas se tiene:

24 cuadros por segundo = 1,440 por minuto = 86,400 por hora
= 172,800 en 2 horas.

Actualmente la computadora mas popular para animación (VAX de DIGITAL), necesita cerca de 10 minutos para generar un cuadro de animación de una escena mas o menos compleja.

Para obtener 5 minutos de animación se necesitan :

5 x 60 x 24 = 7,200 cuadros en 5 minutos.

Cada cuadro tarda en generarse 10 minutos, por lo que se tiene un total de 72,000 minutos

Para escenas mas complejas puede llevarse hasta 4 horas por cuadro.

Comúnmente se piensa que un animador es una persona que dibuja cuadros para simular el movimiento. En realidad la persona que realiza animación es un artista, ya que no solamente se trata de dar movimiento a las figuras, sino de darles vida, de hacerles transmitir un sentimiento, una expresión.

Clasificación de la animación por computadora

Existen dos clases de animación por computadora :

a) Animación de alta técnica.

Este tipo de animación es la que se utiliza para las peliculas y que tradicionalmente se ha hecho con dibujantes.
Al utilizar la computadora, el artista elaborará solamente el

primero y último cuadros y la computadora producirá los cuadros intermedios.

Los programas que producen la animacion trabajan en base a fórmulas matemáticas complejas que emplean grandes bases de datos en las cuales se define el objeto tal como existe en el espacio matemático.

Para elaborar los dibujos, se necesitan profesionales y especialistas que manejen técnicas de alto grado de dificultad como pueden ser transformaciones en tres dimensiones, algoritmos de lineas ocultas, curvaturas, etc...

Este tipo de animación requiere de equipos muy caros con pantallas de color especiales de tipo rastreador los cuales tienen asociada una area de memoria llamada "frame buffer".

En el "frame buffer" se guarda la imágen temporalmente para su despliegue posterior en la pantalla.
Puede usarse una cámara para filmar directamente de la pantalla , pero para obtener una mejor calidad, se utiliza una grabadora de peliculas. La computadora registra la posición, color,etc... de la figura y envia esta información a la grabadora, la cual la traslada a la pelicula para cada cuadro.

Cuando se termina toda la secuencia, se verifica y de ser necesario se corrige, se reprocesa y se regraba.

b) Animación de bajo costo o animación en tiempo real.

Se utiliza principalmente en las computadoras personales, para juegos y programas educacionales.

Las microcomputadoras utilizan su memoria para guardar la imagen y desplegarla en la pantalla. La diferencia entre esta animación y la anterior es que en este caso se efectua en tiempo real, es decir , no hay que esperar al proceso de captura y filmación de los cuadros. Las escenas se corrigen al instante. Por otra parte, las computadoras personales no tienen una resolución tan alta como la de las máquinas que se usan para el caso anterior, pero son menos costosas.

1.6 APLICACIONES DE LA ANIMACION POR COMPUTADORA

Dada la gran penetración de las microcomputadoras en la vida diaria, se tienen una infinidad de aplicaciones para la animación por computadora.

El hecho de que las imagenes visuales como las del cine y la televisión tengan una gran influencia en las culturas actuales ha producido que la animación por computadora haya tenido un gran desarrollo en nuestros dias.

a) La animación en el cine.

En este punto pueden mencionarse algunas peliculas en las que se han realizado secuencias completas en una computadora con un gran realismo como por ejemplo la Guerra de las galaxias.

Otro ejemplo es la produccion cinematográfica "TRON", producida por Walt Disney. En esta pelicula se utilizaron una gran cantidad de gráficos por computadora.

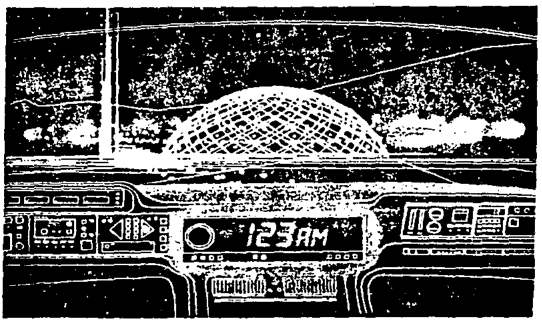


Fig 1.11 Aplicaciones en el cine

Una de las peliculas que tiene efectos muy sofisticados es "Star Trek II". La escena que muestra el Génesis, y que dura un minuto fue producida en cinco meses por una computadora.

b) En el espacio.

Podemos mencionar en este punto las naves Pioner y Voyager de la NASA como apoyo a los viajes espaciales.

Mediante programas, los cientificos de la NASA puden observar las trayectorias que se siguen como si fueran dirigiendo un vehiculo espacial ellos mismos.
Los programas contienen diversas leyes fisicas que regulan el movimiento de la nave.
Con el uso de las mismas tècnicas de simulaciòn se observan la entrada a la atmòsfera y las òrbitas a seguir. Con este tipo de proyectos, se evitan muchos errores en el vuelo real y se observa el comportamiento de la nave en diversas situaciones.

c) En medicina.

En este campo se utiliza principalmente para la visualizaciòn de algunos òrganos y estructuras de los enfermos.

El mèdico observara las figuras desde muchos puntos de vista, de varios ángulos y en algunos casos en secciònes transversales. Los pacientes tambièn pueden observar estas figuras y seguir màs de cerca su curaciòn.

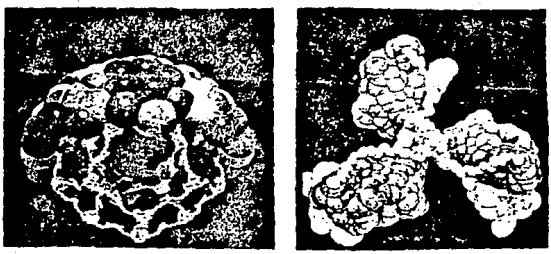


Fig 1.12 Aplicaciones en la medicina

d) En los deportes.

En esta rama la animaciòn se ha utilizado para mostrar a los atletas como mejorar su desempeño.
En la pantalla aparecen varias posiciones de un atleta en un determinado deporte. Estas posiciones son las òptimas para cada movimiento.

También puede hacerse que el computador digitalice el movimiento del atleta y lo muestre en imágenes, lo cual ayuda mucho al entrenamiento de los deportistas.

e) En la educación

En este campo se puede tener un futuro muy promisorio, aunque actualmente no se ha utilizado mucho la animación.

Pueden crearse lecciones por computadora con dibujos animados que serían muy motivantes para los alumnos.
En algunos casos, principalmente en el area científica pueden utilizarse para simular fenómenos o experimentos con cuerpos en movimiento que involucren leyes de la física, como tiro parabólico entre otros.

Si se aplican las técnicas de animación y simulación a la enseñanza , los alumnos podrán ver los efectos de diversos experimentos en la pantalla sin tener que hacerlos en un laboratorio, variando igualmente las condiciones del experimento, lo cual los llevaría a una mejor comprensión de los fenómenos.

e) En Ingeniería

Se aplica primeramente en el diseño de estructuras e instrumentos como pueden ser autos y aviones, simulando a la vez, los fenómenos que pueden afectarlos.

La estructura se diseña en la pantalla y se visualiza desde diferentes puntos de vista y ángulos con el uso de técnicas para efectos tridimensionales .

El diseño de construcciones anterior a su edificación puede prevenir muchos errores, que serían irremediables después.

Con programas más complejos, puede someterse a estas estructuras a condiciones ambientales prefijadas y ver los efectos que provocan.

f) CAD (Diseño auxiliado por computadora) /CAM (Manufactura auxiliada por computadora).

En este campo se utiliza como auxiliar en el diseño y simulación principalmente en la Ingeniería Industrial, donde la información

acerca de la forma de los objetos es muy importante.

Los sistemas de esta area deben tener una gran capacidad para el modelado y diseño de productos y manufactura. Aqui resulta mas importante que el sistema provea precisión y definición clara de las figuras, donde no haya ambiguedades en el despliegue de los objetos.

Existe un sistema llamado Geomap III para modelado de sólidos, el cual integra procesos CAD/CAM. Se usa para todo tipo de diseños industriales como puede ser el de máquinas-herramienta, tornillos, tuercas, ejes etc...

También se utiliza para la simulación de robots.

g) En el arte

En este campo no se ha utilizado mucho porque los artistas no confian en las computadoras como medio de expresión, aunque la máquina ofrece una paleta de 16 millones de combinaciones de colores y permite la creación de efectos y figuras realmente impresionantes que bien pueden llevar al nacimiento de un nuevo arte.

En la Universidad de Columbia se desarrolló un sistema llamado CARTOS. Su objetivo original era apoyar a los biólogos en el análisis de elementos microscópicos complejos en tres dimensiones ya que puede reconstruir y simular objetos tridimensionales a partir de datos obtenidos de microscopios electrónicos.

Este sistema también se ha utilizado para producir imágenes artisticas por medio de una pantalla tipo rastreador y maneja algoritmos de lineas ocultas, edición y rutinas para color y sombreado.

En este ámbito se cuenta con programas para enseñanza del ballet en los cuales aparecen los bailarines ejecutando el ballet como lo harian en el estrado. También pueden hacerse cosas similares a las mencionadas en el inciso de deporte.

h) En publicidad.

En esta area se ha invertido mucho dinero puesto que la

publicidad hecha en la computadora resulta muy novedosa para la mayor parte del pùblico que no está familiarizado con las màquinas, creando comerciales y anuncios deslumbrantes que generan grandes volùmenes de ventas. Junto con el rayo laser la computadora ha revolucionado la publicidad.

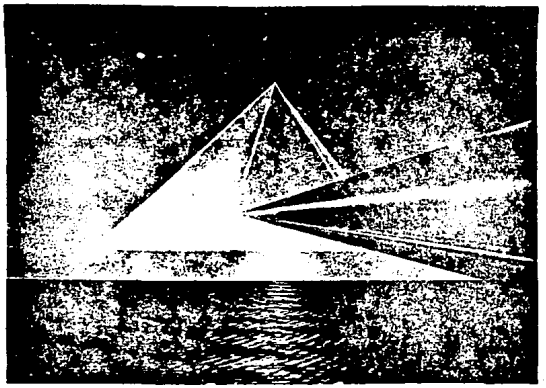
i) En Biologia.

Se ha utilizado para la simulacion de la formaciòn de moléculas desde su fase de desarrollo, hasta que la molècula està completa. Puede rotàrse y observarse desde diferentes puntos de vista, cortes y ángulos.

1.13 Aplicaciones en el arte

j) Juegos.

Aqui podriamos mencionar un sin fin de nombres de juegos construidos con técnicas de animación. Estos juegos han llegado hasta nuestros hogares y fuera de ellos los encontramos en las máquinas tragamonedas. Este tipo de animación que se hace en microcomputadoras y no es muy costosa, ha proporcionado gran entretenimiento a chicos y grandes.



CAPITULO II
EQUIPO PARA PRODUCIR ANIMACION

2.1 CONCEPTOS GENERALES

Cuando se habla de animación, se tiene que tomar en cuenta el papel tan importante que juega el equipo disponible, ya que de él depende en mucho el éxito del proyecto.

Del equipo físico va a depender en gran parte el hecho de que se pueda hacer animación en tiempo real o animación de alta técnica cuadro por cuadro.
A continuación se presenta un breve análisis de los diferentes dispositivos de apoyo para graficación y en particular para animación de figuras.

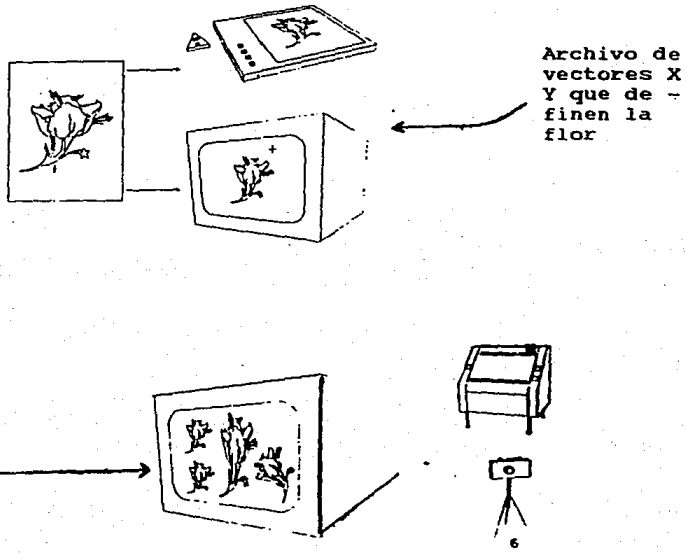


FIG. 2.1 Dispositivos de graficación

2.2 EQUIPO NECESARIO EN UN SISTEMA DE GRAFICACION

Todo sistema de graficación, va a tener tres diferentes dispositivos:

a) Dispositivos de entrada. Son aquellos que permiten introducir al computador la figura en cuestión.

b) Dispositivos interactivos. Son aquellos que despliegan la figura y permiten modificarla.

c) Dispositivos de salida. Son aquellos por donde se obtiene la versión final de la figura. Puede ser una pantalla, un graficador, una impresora o un medio fotográfico.

Los pasos a seguir para obtener una figura en la computadora son:

1. Contar con un modelo original de la figura en papel.

2. Introducirlo en la computadora, digitalizándolo o dibujándolo interactivamente

3. Almacenar la figura en la computadora como una serie de definiciones de contorno (polígonos)

4. Contar con los programas necesarios para manipular el archivo de polígonos.

5. Obtener la figura completa en papel, en pantalla o en cualquier otro medio de salida.

A continuación se describen los diferentes dispositivos.

2.2.1 DISPOSITIVOS DE ENTRADA

Son necesarios para introducir la figura a la máquina y para manipular los datos con ayuda de diversos programas.
Algunos de estos dispositivos se presentan a continuación:

a) Tableta gráfica o tabla para digitalización

La tableta está formada por una serie de líneas paralelas a los ejes x-y.

Algunas de estas tabletas pueden contener una red de 1024 x 1024 alambres, cada uno de los cuales transporta una señal digital.

Las coordenadas que se trabajan son absolutas con respecto al

origen de la tableta.

El usuario puede dibujar con una pluma especial, como si fuera lápiz y papel. Al oprimir la pluma contra la tableta, un botón colocado en la pluma, permite indicar la posición X-Y.
Este dispositivo es uno de los que se utilizan comúnmente como medio de entrada.

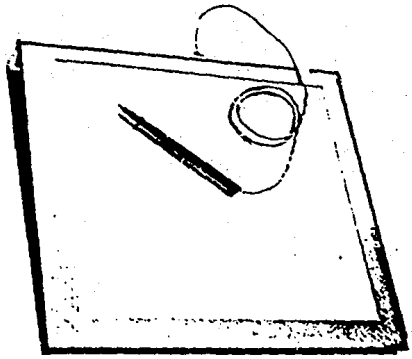


Fig 2.2 La tableta digitalizadora

b). Palancas de control

Son similares a las palancas de mando de las aereonaves. La palanca puede moverse hacia la derecha, arriba, abajo o a la izquierda.

Mediante 2 potenciómetros conectados a la palanca, se convierte el movimiento en voltajes, los cuales a su vez se convertirán en señales digitales . Revisando el ángulo al que se mueve la palanca, se puede deducir la velocidad a la cual se moverá el cursor.

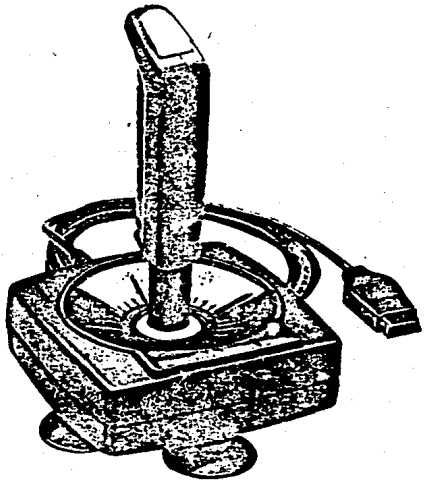
El problema con este dispositivo es que se necesita un converti dor muy caro para el movimiento en pantallas de alta resolución.

Las palancas son muy populares para los programas de juegos de las microcomputadoras.

c). Pluma de luz

Se usa principalmente en monitores de refrescamiento. El dispositivo detecta la presencia de luz en la pantalla a través de una fotocelda que tiene en la punta. Cuando la fotocelda percibe

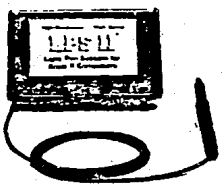
la luz, las coordenadas del punto se proporcionan al usuario del programa.



2.3 Palanca de Control.

La persona señala puntos con la pluma en cualquier lugar de la pantalla y oprime un boton que tiene la pluma para fijar un punto.
Este método resulta un poco cansado ya que hay que mantener todo el tiempo la pluma sobre la pantalla, sin embargo es muy útiles para el manejo de programas con menús.

Fig 2.4 La pluma de luz



d). Ratón

El ratón es una pequeña caja con ruedas que se puede mover facilmente. Cuenta con unos potenciómetros conectados a las ruedas,

los cuales convierten la rotación mecánica en una señal binaria y envían los pulsos digitales al computador.
En la parte superior tienen unos botones que pueden programarse para ejecutar una cierta acción.

No es útil para introducir datos o gráficas, solo para interactuar con un programa.

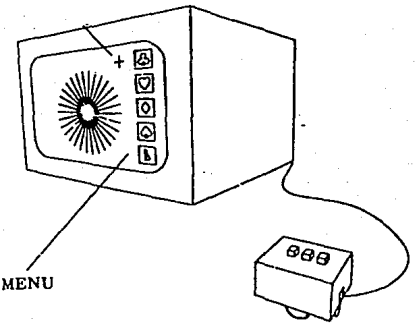


Fig 2.5 Ratón

e).Esfera Giratoria

El dispositivo consta de una esfera que al girar habilita unos potenciómetros produciendo coordenadas en forma digital.
El control del cursor es proporcional a la velocidad a la cual gira.

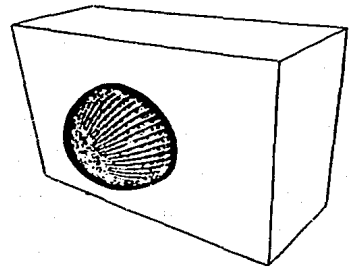


Fig 2.6 Esfera giiratoria

2.2.2 DISPOSITIVOS DE SALIDA

Son aquellos que producen una imagen visual o en papel de la figura.

a). Graficadores de pluma

Existen 3 diferentes tipos de graficadores que se describen a continuación.

a) Graficadores de pluma con tambor o cilindro:

Se utilizan en equipos grandes, pueden realizar grandes figuras a color.

El papel se encuentra montado sobre un tambor o cilindro, el cual gira hacia adelante o hacia atrás . Cuentan también con una barra metáica posicionada encima del papel, la cual sostiene unas plumillas que se mueven en dirección horizontal sobre la barra. Estas plumillas tienen una velocidad aproximada de 40 cm/seg.

Generalmente no pueden manejar mas de 4 plumas.

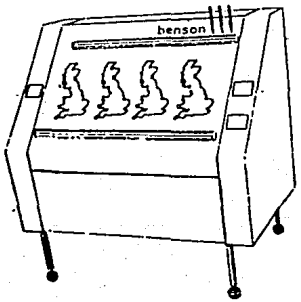


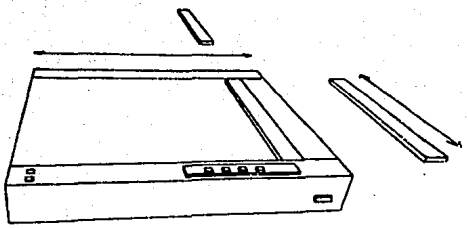
Fig 2.7 Graficador de plumas con tambor

b). Graficador plano

Pueden ser modelos de escritorio o de tamaño grande hasta de 4 x 2 mts.
Se componen de una tabla que tiene una barra transversal la cual se mueve en la dirección X del plano. En esta barra se encuentran suspendidas las plumillas que avanzan sobre la barra en la dirección Y.
Su velocidad aproximada es de 25 cm/s. Tienen la ventaja de que

aceptan cualquier tipo de papel inclusive acetato . Pueden contar hasta con 8 plumillas.

Algunas marcas de graficadores de este tipo son Hewlett Packard, Calcomp 81 y Gould 6120.



Las plumillas se mueven a lo largo de la barra

Fig 2.8 Graficador plano

c). Graficadores electrostáticos

Estos graficadores utilizan principios similares al fotocopiado. El papel entra por un extremo en una sola dirección. Los datos de la figura se trasladan a unos electrodos que se encuentran en una barra , los cuales retendrán un polvo similar al toner y lo colocarán en el papel.

Son muy rápidos pero solo manejan un color y una resolución fija. (80 puntos /cm)

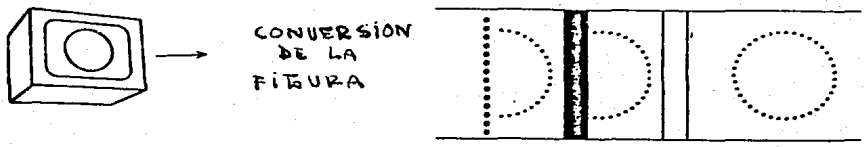


Fig 2.9 Graficador electrostático

2.2.3 DISPOSITIVOS DE SALIDA DE MAS COLORES

a) Sistemas de cámara fotográfica.

La finalidad es tomar una fcto de la imágen que se presenta en la pantalla.

En estos dispositivos se presenta el problema de la captación de los colores y la distorsión por la curvatura de la pantalla.
La cámara puede conectarse a la pantalla y utilizarse filtros para color. Pueden obtenerse fotos de mejor calidad que la imagen que tiene la pantalla.

b) Impresoras de matriz de punto

Es una impresora de matriz convencional que utiliza varias cintas, una de cada color. Generalmente se recorre tres veces la misma linea o la misma hoja para imprimir cada vez con un color diferente.

Manejan una cierta resolución y son rápidas . Tienen baja calidad si se utiliza para el dibujo de areas de un sólo color.

c) Graficadores que inyectan tinta

Forman la imagen depositando finas gotas de tinta sobre el papel. Constan de 3 elementos principales :

1. Una cabeza que contiene 3 tipos de tinta generalmente magenta, amarillo y cyan.
2. Un cilindro giratorio (para papel o acetato)
3. Un microprocesador para controlar la caida de las gotas al papel.

Dependiendo de la marca pueden dibujar desde 1 a 5 puntos por mm, algunos como el Applicon hasta 50 puntos por cm.
Son muy rápidos y producen sombreados de alta calidad.

d) Grabadoras de peliculas.

En este caso se puede fotografiar la imagen o utilizar rayos de electrones o laser para escribir directo sobre la película.

Son muy caros ,se utilizan para animación de alta técnica. El diseñador construye la figura en un rastreador de resolución media y de ahi se envia al grabador de peliculas.

Ejemplos de estos dispositivos son el FR80 (4000 X 6000), CAL-

COMP 1670 (16284 X 16384) con salidas para pelicula de 16 MM o de 35 MM.

2.2.4 DISPOSITIVOS INTERACTIVOS

Existen 3 tipos principales de dispositivos para el despliegue directo de imágenes.

1. Tubo de visión directa de memoria o "DUST".
2. Pantallas de refrescamiento o caligráficas.
3. Sistemas de rastreo.

Se describirán primero los principios básicos de funcionamiento de una pantalla de video.

Toda pantalla está basada en un dispositivo de rayos catódicos. Este dispositivo consta de un cañón de electrones, los que al estrellarse sobre una pantalla recubierta de fósforo, emiten luz.

Un cátodo cargado negativamente se calienta hasta que produce una lluvia de electrónes. Estos son atraidos hacia la superficie de la pantalla.
Si se mantiene una corriente ininterrumpida de electrones, éstos harán brillar la pantalla. El rayo de electrónes se desvia hacia la derecha, izquierda, centro ,arriba o abajo de la pantalla mediante unos amplificadores.

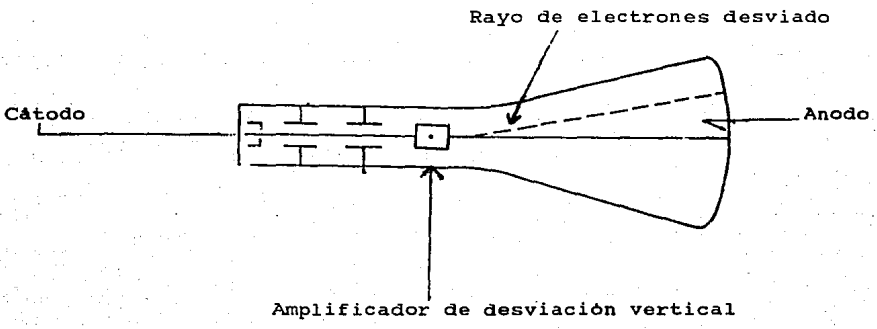


Fig 2.10 Tubo de rayos catódicos

A continuación se describen los principales dispositivos interactivos, señalando en cada caso su posibilidad de producir animación.

a) Tubo de visión directa de memoria "DUST"

Fue diseñado originalmente para imágenes que no requieren una rápida actualización.

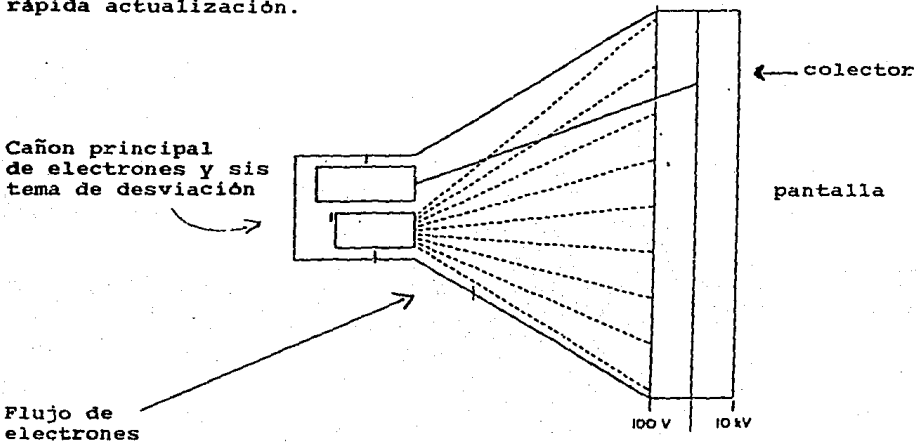


Fig 2.11 Tubo de visión directa de memoria "DUST"

Utiliza un tubo de rayos catódicos controlado electrostáticamente. El cañón principal emite una corriente de electrones hacia una fina malla de alambre situada atrás de la pantalla y un segundo cañón emite un rayo de electrones de baja energia.

Las areas cargadas positivamente aceleran el flujo de electrones hacia la red, causando la iluminación del fósforo de alta persistencia en la pantalla, en correspondencia a los patrones de carga de la malla.
La pantalla se encuentra a alto voltaje (10kv), para poder acelerar los electrones a través de la malla.
Una linea o caracter puede permancer visible hasta una hora si no se le borra.

La pantalla se borra enviando a todo el tubo un voltaje que produce luz negra. Los monitores de este tipo son llamados también monitores de búsqueda aleatoria, ya que una linea o vector puede dibujarse directamente de un punto a otro.

Ventajas

1. Despliega una imágen firme y libre de parpadeo, aun cuando

despliegue muchos datos.
2. Ofrece alta resolución. La típica es 1024 x 1024 en monitor de 8 x 8 pulgadas cuadradas, o 4096 x 4096 en monitor de 14 x 14 pulgadas cuadradas .
3. Es mas económico.

Desventajas

1. Es difícil borrar una parte de la imágen. Para cambiar un fragmento de la figura, hay que limpiar toda la pantalla y redibujar.
2. Tiene bajo brillo
3. No tiene posibilidad de diferentes niveles de tonos o sombreados
4. No tiene color
5. No se puede utilizar una pluma de luz.
6. No permite animación.

b). Pantalla de refrescamiento

Al contrario de la anterior, no utiliza fósforo de larga persistencia .
Funciona de la siguiente manera :

a) La computadora produce una señal digital en coordenadas X-Y
b) Esta señal entra al CDA (convertidor digital a analógico)
c) Los amplificadores convierten los voltajes analógicos en corrientes de magnitud apropiada.
d) El rayo se direcciona en la pantalla al punto proporcional al indicado en X-Y.
El brillo o intensidad se controla con un tercer CDA.

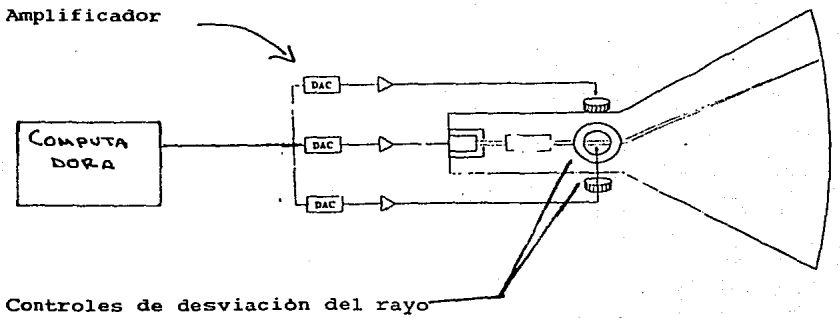


Fig 2.12 Tubo de rayos catódicos

Al choque del rayo, el fósforo brilla por un corto tiempo. Para que la imágen se mantenga, deberá dibujarse antes de que el fósforo deje de emitir luz,por lo tanto hay que volver a enviar la información a la pantalla rápidamente para que no se perciba el parpadeo.

Este tipo de pantallas necesita "refrescar" la imagen entre 30 y 50 veces por segundo. Estas pantallas contienen adicionalmente un procesador de despliegue entre la computadora y la pantalla, el cual lee un archivo en donde se encuentra la descripción de la figura.

Si la figura es muy compleja, se percibirá un leve parpadeo en la imagen, por lo que normalmente el archivo de descripción se guarda en una memoria local.

En las pantallas modernas, es posible dibujar cuando mucho 100,000 vectores dentro de la velocidad requerida para que no haya parpadeo.
Un problema que se presenta en este caso, es la velocidad a la que se pueden efectuar las transformaciones sobre la figura.
El archivo de descripción de la figura contiene :

1. INSTRUCCIONES PARA DIBUJAR LA FIGURA

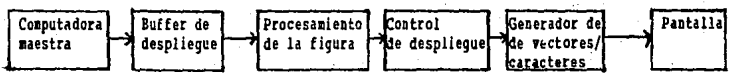
- Puntos definidos en coordenadas absolutas
- Definición de vectores.
- Generación de texto.
- Brillantez

2. INSTRUCCIONES DE CONTROL

- Instrucciones de salto, las cuales permiten el uso de sub-rutinas.
- Transformaciones de matrices.

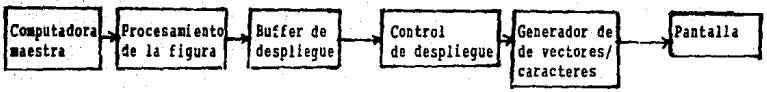
Cuando la figura está completa, una instrucción de salto devuelve el control al principio del archivo para reiniciar el ciclo de refrescamiento.

Para este tipo de pantallas podemos tener dos posibles configuraciones como se muestra:



En la primera el procesador de la imagen es más lento que la velocidad de refrescamiento de la pantalla y debe utilizarse un doble buffer.

Mientras que una figura se procesa, la otra se despliega en la pantalla y asi sucesivamente (Apple).



En la segunda el procesador de la imagen es mas rápido que el de refrescamiento de la pantalla. En este caso la imagen se congela momentáneamente en el "buffer". Los vectores se guardan generalmente en coordenadas del usuario como números de punto flotante. El procesador de la pantalla lee la información del buffer y la pasa a través del generador de vectores en un ciclo de refrescamiento.

Estas pantallas permiten segmentación de la figura lo cual contribuye a tener movimiento dinámico o animación.

Otro factor importante es la velocidad de comunicación entre el procesador central y el dispositivo de despliegue.

Si se considera la actualización de una curva con 250 segmentos o puntos que la describen, la velocidad de comunicacion necesaria debe de ser mayor que un megabit por segundo. Para figuras tridimensionales complicadas pude llegar a los 10 megabits por gundo. En algunos casos será necesaria una interfase de acceso paralelo o directo a memoria entre el procesador central y el dispositivo gráfico.

VENTAJAS

1. Manejan alta resolución
2. Tienen un alto grado de interacción
3. Pueden obtenerse varias intensidades
4. Tienen borrado selectivo
5. Alta capacidad de transformaciones como rotación, escalamiento, traslación y perspectiva.

DESVENTAJAS

1. Son muy caras
2. Es difícil evitar el parpadeo en figuras complejas
3. Tienen poca capacidad de color
4. La elaboracion de imágenes realistas es difícil

Se utilizan en aplicaciones interactivas como en el diseño por computadora. Su problema radica en la manipulación de imágenes complejas.

Algunos de estos monitores son a color y se les llama "pantallas de rayo de penetración para color". El fósforo que usan generalmente tiene dos cubiertas una roja y una verde. El rayo puede tener dos penetraciones en el fósforo y producir diferentes colores, pueden desplegar pocos colores como rojo, verde, naranja y amarillo, el azul es difícil de desplegar.

c). Pantallas tipo rastreador

Una pantalla tipo rastreador es una matriz de puntos donde cada punto se conoce como "pixel".

En contraste con los dispositivos anteriores en los cuales pueden dibujarse lineas dando las coordenadas de dos puntos distantes, una pantalla tipo rastreador se considera como una matriz de celdas cada una de las cuales puede brillar. Una linea se define como una serie aproximada de puntos.

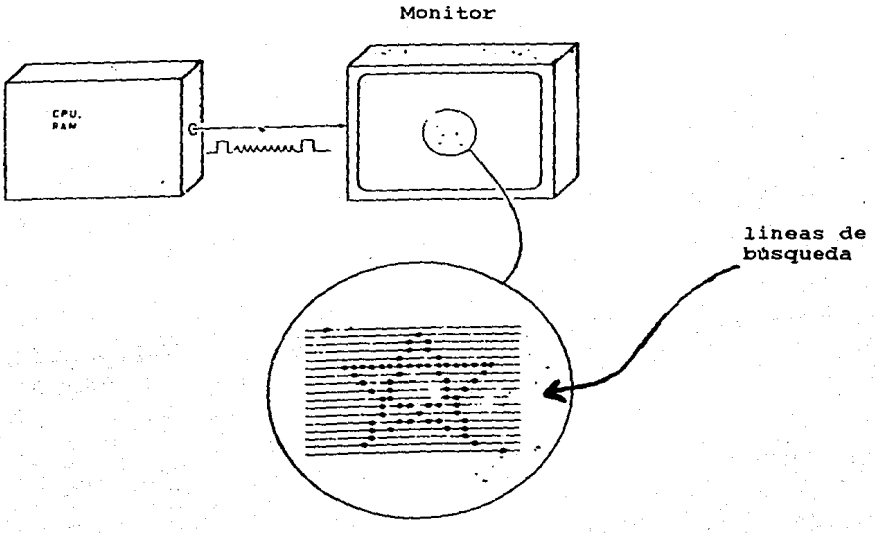


Fig 2.13 Pantalla tipo rastreador

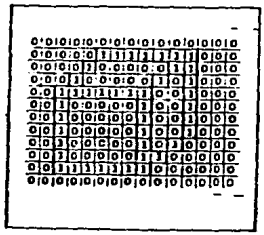
El rayo de electrones puede desviarse y dirigirse hacia el patrón de líneas . A cada una de las líneas que forman el patrón de rastreo se les llama línea de búsqueda.

El rayo se desvia zigzageando en el patrón, donde sube y baja muchas veces por segundo. Este rayo se hace incidir sobre alguna línea de búsqueda la cual aparece como un punto, A este punto se le denomina "pixel".

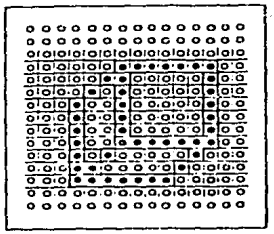
Los "pixels" se alojan en un area de memoria especial llamada memoria para imágen en la cual los puntos se representan como diferentes niveles de voltaje, el 1 es encendido y el 0 apagado.

Existen circuitos especiales entre la memoria y la pantalla que indican cual "bit" esta encendido o apagado y mandan el rayo al lugar adecuado.

Imaginese la memoria para imágen como un plano bidimensional de "bits" donde cada "bit" corresponde a un "pixel". Esto es lo que se llamama un "plano de bits". En este plano hay una correspondencia uno a uno entre "bits y pixels".



Representación en dos dimensiones



"pixels en la pantalla"

Fig 2.14 Plano de bits

En vista de que el plano de bits es un dispositivo digital y la pantalla tipo rastreador es un dispositivo analógico, se requiere una conversión de la representación digital a la analógica. Esto se hace por medio de un convertidor analógico a digital (DAC).

En una pantalla de resolución de 320 x 200 "pixels" se tienen 64,000 "pixels" que se alojan en aproximadamente 8 K (64000/8 = 8000 "bytes").

Si se desea variar la intensidad de cada "pixel" en una pantalla blanco y negro, se deben agregar más bits por "pixel". Por ejemplo si se desean 8 intensidades distintas hay que repre-

sentar 8 estados en donde son suficientes 3 "bits" por "pixel", donde cada combinación de 3 bits representa una intensidad diferente.

Los monitores de color también llamados monitores "RGB= red, green,blue" utilizan 3 rayos de electrones, uno para cada color. El monitor contiene 3 diferentes combinaciones de fosforo en triadas de puntos o en bandas de 3 colores de fósforo).

Los colores están colocados en forma de triángulo . Para estar seguro que los cañones excitarán el fósforo correcto, se coloca una placa de metal entre el cañón de electrónes y la pantalla. Las perforaciones en la placa están colocadas como los puntos en forma triangular.

Para pantallas de alta resolución usualmente se tienen 2 o 3 triadas de colores por cada "pixel", en este caso se debe controlar la intensidad de estos colores y la mezcla de los mismos.

Cada "pixel" tiene la posibilidad de adquirir cualquiera de los 3 colores por lo tanto se usan 3 bits para cada color el cual podrá verse en 8 intensidades distintas. Esto producirá que se tengan 9 "bits" por "pixel" (3 x 3 =9 para los tres colores).

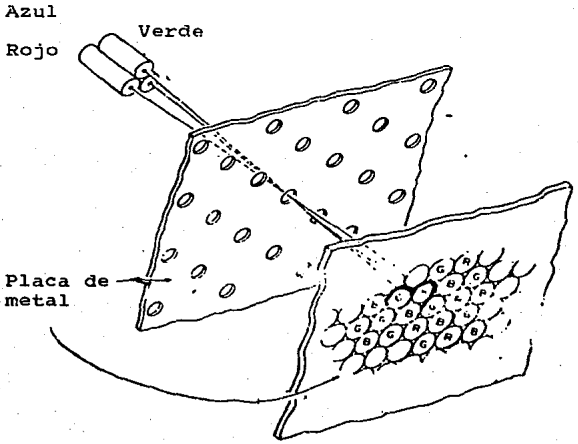


Fig 2.15 Funcionamiento del monitor de color "RGB"

Como cada "pixel" puede tener de 1 a 3 colores y 8 intensidades diferentes, (8 x 8 x 8 = 256) puede obtenerse una gama de 512 colores distintos.

Ventajas

1. Pueden rellenarse polígonos a color rápidamente.
2. Puede evitarse el parpadeo en imágenes complejas.
3. Es posible lograr realismo fotográfico usando polígonos sombreados y efectos para textura e iluminación.

Desventajas

1. Poca resolución por los grandes requerimientos de memoria y la complejidad de los circuitos
2. Las transformaciones geométricas por hardware son posibles pero muy difíciles
3. Se permite borrado selectivo con algo de dificultad. Normalmente, redibujar una línea o punto constituye un borrado.
4. Para la animación, sólo los sistemas avanzados, pueden cargar la memoria del video lo suficientemente rápido.

Aplicaciones

1. Simuladores de vuelo
2. Animación utilizada por Walt Disney. También en animación científica.
3. Diseño e ilustración para partes mecánicas y circuitos integrados (CAD/CAM).

2.3 EQUIPO PARA ANIMACION DE ALTA TECNICA

El problema de muchas computadoras radica en su velocidad de proceso, la cual no les permite generar imágenes sucesivas suficientemente rápido para crear la ilusión del movimiento continuo.

El problema se presenta cuando se quiere producir animación , "en tiempo real", ya que en términos de resolución muchas computadoras pueden generar imágenes que muestran mucho más detalle que un televisor, aunque no puedan generar un nuevo cuadro en un rango de 1/24 a 1/60 de segundo como el equipo de cine o televisión.

Ante este problema, muchas personas han decidido usar la máquina solamente para generar cada uno de los cuadros de la película, los cuales se integrarán y se animarán mediante otros medios cinematográficos.

Agruparemos a todas las técnicas que existen con el fin de utilizar al computador como una herramienta para la producción de películas animadas profesionalmente bajo el nombre de

animación de alta técnica o animación cuadro por cuadro.

Para este tipo de animación es común utilizar pantallas tipo rastreador, en conjunto con una larga memoria digital.

Una larga memoria digital es un conjunto de planos de "bits" colocados uno encima de otro y considerados como una unidad.

El número de planos de "bits" que se usan determinarán las intensidades de los "pixels". Entre mas planos se tengan habrá una mayor gama de colores.

El número de "bits" horizontales y verticales en cada plano determina la resolución de la pantalla. Las casas de animacion artistica utilizan memorias largas llamadas "frame buffers" hasta de 1024 x 1024 pixels y 24 planos de bits, de los cuales se usan 8 bits para cada color.
En 8 bits se pueden representar 256 diferentes intensidades para cada color y por lo tanto, tenemos 16,777,216 colores diferentes (256 x 256 x 256 = 16,777,216) lo cual ocuparia aproximadamente 3072 KB de memoria (128 k por plano de bits x 24 = 3072 k de memoria).

Como se observa se necesita una gran cantidad de memoria para almacenar toda esa información y esto es muy costoso.

Actualmente las memorias largas que hay en el mercado representan las intensidades con 1, 2, 4, 8 o más bits de memoria. 1 "bit" es suficiente para gráficas simples; 2 y 4 para colores sólidos o sombras y 8 bits para detalles finales.

2.3.1 MAPA DE COLOR

La forma en que se codifique la imágen para cada "pixel" depende de cuantos bits se tengan por "pixel". Por ejemplo si se cuenta con 8 bits por "pixel", podemos tener 3 para el rojo, 3 para el verde y 2 para el azul (la razón de esto es que el ojo es menos sensible a la región azul del espectro).

Si usamos sólo 8 "bits" por "pixel" nuestro rango de colores es limitado. Con 3-3-2 podemos tener 8 x 8 x 4 = 256 diferentes matices, pero el ojo humano puede distinguir mas matices.

Hay una forma de obtener más matices sin usar más memoria. Este método se llama mapeo de color y se usa en varios equipos de alta técnica y en algunas microcomputadoras como la Atari.

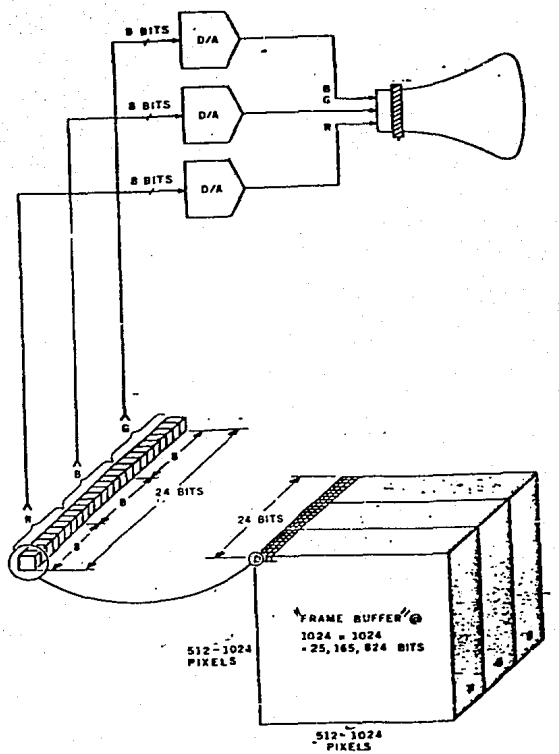


Fig 2.16 Memoria larga ("Frame buffer") de 24 "bits" por "pixel"

En este caso, los valores de los bits se interpretan como direcciones o apuntadores hacia una tabla de colores que se encuentra en memoria "RAM" o en una colección especial de registros .

Esto significa que los 8 bits de un cierto "pixel" se direccionan a una tabla la cual contiene tres valores individuales de color, uno para cada color primario.

En la tabla se muestran las combinaciones de 3 planos de "bits".

	Rojo	Verde	Azul
Negro	0	0	0
Rojo	1	0	0
Verde	0	1	0
Azul	0	0	1
Amarillo	1	1	0
Cyan	0	1	1
Magenta	1	0	1
Blanco	1	1	1

Con este enfoque una memoria larga "frame buffer" de 8 bits por "pixel" puede direccionar a una tabla de colores con un máximo de 256 valores en ella. Esto significa que la pantalla puede desplegar 256 diferentes colores al mismo tiempo.

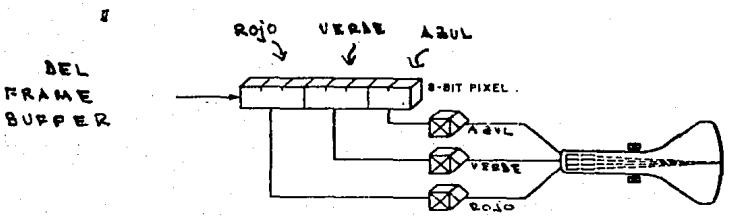
Cada uno de estos colores puede definirse con un alto grado de presición ya que en la tabla se pueden usar mas de 8 "bits" por "pixel" de ser necesario.

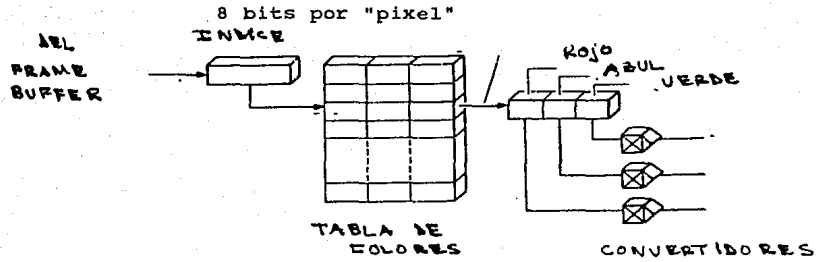
Por ejemplo; la tabla puede tener un ancho de 24 bits permitiendo 8 bits para cada color primario o 256 matices para cada color o sea un maximo de 16 millones de matices por "pixel". Hay que tener en mente que sólo se requieren 8 bits por "pixel" para direccionar a la tabla.

Otra ventaja de este método es que al cambiar los colores en la tabla. Automáticamente se cambian en la pantalla, permitiéndonos variar el colorido de la pantalla e intercambiar colores de manera muy sencilla.

Ventajas

1. Permite mas colores sin incrementar la memoria por "pixel".
2. Los colores pueden alterarse facilmente.
3. Los colores pueden mezclarse
4. Los planos pueden bloquarse y manejarse de forma independiente
5. Permite una animación limitada





8 bits por "pixel" que apuntan a una mapa de colores de 24 bits

Fig. 2.17 Valores de los "pixels" para indexamiento en un mapa de color

2.3.2 MEZCLA DE PLANOS DE BITS EN EL VIDEO .

Tratando al frame buffer como un conjunto de planos en lugar de tratarlo como unidad, podemos tener una serie de imágenes separadas. Por ejemplo un frame buffer de 8 bits por "pixel" puede ser tratado como 2 imágenes de 4 bits, ó 4 imágenes de 2 bits ó como 8 de blanco y negro.

Esto significa que puede variarse el conjunto apagando o encendiendo alguno de los planos, permitiendo, por ejemplo mantener un plano estático como fondo mientras otros planos se ven a través de él logrando efectos especiales.

2.3.3 OTROS METODOS

El método de la memoria larga "frame buffer" es muy caro pues requiere demasiada memoria, ademas resulta muy lento en su tiempo respuesta, aunque produce imágenes con alto grado de detalle.

Finalmente cuando se transfiere la imágen al disco se requiere tiempo y espacio. Una solución a este problema se basa en la

compactación de la imágen, por medio de técnicas de codificación.

Otro método es la conversión geométrica. En este método la imágen se guarda como una descripción geométrica en lugar de ser un mapa de intensidades de "pixel" en el frame buffer.

La conversión geométrica usa un archivo especial de despliegue, el cual es otra area de memoria, para congelar los puntos finales de la imagen.

Un circuito especial lee el archivo varias veces , para generar la imagen y determinar matemáticamente si un segmento de línea intersecta la línea que va a ser dibujada o no. La imagen puede cambiarse modificando la descripción del archivo de despliegue.

El problema en este enfoque es que se necesita hardware especial para realizar la conversión geométrica a una velocidad de 30 cuadros por segundo.

Otro método de codificación se basa en el hecho de que una línea de búsqueda tiene varios "pixels" que tienen la misma intensidad de color.

Si se codifica la longitud e intensidad de cada secuencia de "pixels" idénticos, reduciremos el monto de memoria y espacio en disco requeridos para guardar la imágen. Cada línea codificada consisitirá en una o más instrucciones, cada una de las cuales define una longitud y una intensidad.

2.3.4 TRASLADO DE LA IMAGEN DEL FRAME BUFFER A LA PELICULA

Un camino para resolver el problema de la velocidad es animar fotográficamente la imagén que despliega la computadora.

Esta técnica permite a la computadora tomar el tiempo necesario para generar cada uno de las cuadros. Cuando la máquina termina de dibujar un cuadro, una cámara de cine que está coordinada electrónicamente con la computadora, transfiere la imágen a un cuadro de filmación, entonces se avanza la pelicula, la computadora dibuja el cuadro siguiente y asi sucesivamente. Cuando la pelicula se ha filmado cuadro por cuadro, se muestra a una velocidad conveniente por medio de la cámara.

Trataremos a continuación algunos de los factores que afectan la toma de la fotografia de las imágenes en la pantalla, como pueden ser el tiempo de exposición, la luz ambiental y la alineación de la cámara con la pantalla.

La imágen se crea en la pantalla mediante un rayo de electrónes de intensidad variable que inciden a alta velocidad sobre la pantalla recubierta de fósforo.
El fósforo brilla cuando el rayo incide sobre él. El brillo depende de la intensidad del rayo de electrones .

El rayo incide sobre una linea de búsqueda cada 60 microsegundos o sea que genera un cuadro completo en 1/60 de segundo.

Una foto de una pantalla de televisión expuesta por menos de 1/60 de segundo aparecerá con unas partes más oscuras que otras. Estas partes no son completamente negras porque el fosforo continúa brillando un tiempo después de que el rayo choca con él.

Con un tiempo de exposición mayor que 1/60 de segundo, se obtendrá una foto con unas partes más brillantes que otras, ya que las partes brillantes de la foto se expusieron el doble que el resto.

Cualquier tiempo de exposición que sea un múltiplo exacto de 1/60 de segundo puede proveer de una excelente foto, pero la precisión de la cámara debe ser un 1% mejor que el múltiplo más bajo, por ejemplo 2/60 o 3/60 segundo.

Un forma correcta para fotografiar la imágen es con un tiempo largo de exposición, ya que cada punto se genera entre n o n+1 veces durante la exposición.

Se recomienda 1/2 segundo en el cual se involucran cerca de 30 busquedas en el video. Un tiempo menor a 1/15 de segundo nos lleva a una notable degradación de la imágen.

Estos problemas de tiempo, resultan menos serios si se usa un monitor de fósforo con alta persistencia. La persistencia del fósforo es una medida de qué tanto tiempo permance excitado el fósforo después del choque con los electrones. El fósforo usado en la mayoría de los monitores verdes tiene una persistencia más alta que un televisor.

Otra problema es la pérdida de luz. La figura de la foto tiene su propia luz y cualquier otra luz del exterior sólo destiñe la foto. Este problema se manifiesta en dos formas: la superficie de la pantalla puede reflejar alguna luz como si fuera un espejo, otra parte de la luz puede dispersarse hacia atrás de la pantalla de fóforo.

Otro de los problemas que surgen es cuando la cámara no está alineadae con la pantalla y ocurren distorsiones. Por ejemplo si se despliegan dos líneas paralelas de igual longitud y una de ellas se encuentra más cercana a la cámara que la otra, ésta aparecerá más larga que la otra.

Para solucionar todos estos problemas es muy conveniente que la cámara esté bajo el control del programa que genera la imagen.

Para fotos a color se utiliza un dispositivo llamado grabador de pelicula para fotografiar la imagen.

Este dispositivo contiene un monitor blanco y negro de alta calidad y 3 filtros de color.

Cuando se usa el filtro rojo, los colores azul y verde no se ven, sólo se observa lo relativo a todas las intesidades del rojo. Entonces se toma una foto, sin mover la pelicula se toman dos mas, con los filtros verde y azul, con lo cual se obtiene con la mezcla una foto de alta calidad.

Pero aun es posible obtener imágenes de más alta resolución. Esto se hace enviando a la grabadora las lineas de búsqueda una a la vez. La computadora despliega una linea en la pantalla la cual se expone a la pelicula. Este proceso se repite para cada una de las lineas de cada cuadro produciendo resoluciones de 6000 x 4000 "pixels" con 9 bits para cada color.

Desafortunadamente este enfoque tiene el inconveniente del tiempo, ya que lleva de 5 a 10 minutos el grabar cada cuadro en alta resolución.

Para resolver este y otros problemas, Lucasfilm desarrollo el sistema de animación llamado "Pixar". Este sistema es una computadora de propósito general para imágenes con un procesador , mucha memoria y lasers para los dispositivos de entrada y salida. Este instrumento puede llevar las imágenes a la pelicula por medio de los lasers.

Se usan los lasers porque son la fuente de luz mas controlable de que se dispone y producen colores muy vivos.

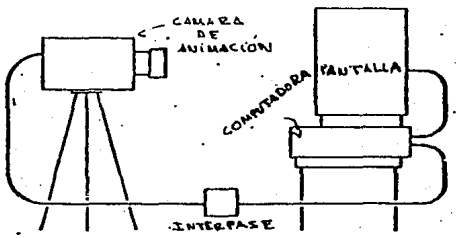


Fig 2.18 Equipo de animación de alta técnica

2.4 ANIMACION EN TIEMPO REAL

En la sección anterior se describen los dispositivos que se utilizan para la animación de alta técnica. Como puede deducirse, este tipo de dispositivos difícilmente pueden realizar animación en tiempo real, pues están hechos para la elaboración de figuras muy complejas y de una gran calidad, pero que consumen muchos recursos.

Para obtener animación en tiempo real se tiene que pensar en una menor calidad de las figuras y menos complejidad en las secuencias de animación.

Si se desea realizar animación en una microcomputadora el problema es más grave aun dado su reducido tamaño y sus características físicas en general. Estas máquinas tienen un bajo costo, pero diversas restricciones como son:

1. La resolución de sus pantallas

2. Su capacidad de memoria

3. Su velocidad de procesamiento

Esto hace más complicada aun la producción de figuras animadas en microcomputadoras, principalmente para animación a color. Se analizan a continuación algunos de estos problemas planteando sus posibles soluciones.

Un sistema típico de graficación en microcomputadora pasa por las siguientes etapas:

1. Creación de la imágen en tres dimensiones

2. Transformaciones geométricas a dos dimensiones

3. Recorte de la imágen

4. Rastreo de la imágen

5. Vaciado a la pantalla

La primera etapa consiste en crear una descripción de la imágen como una lista de primitivos gráficos (líneas, polígonos y cubos) descritos en coordenadas tridimensionales.
En la etapa de transformaciones geométricas los primitivos gráficos se convierten del modelo tridimensional, a su posición propia en la pantalla (X,Y) en un modelo bidimensional.
En estas transformaciones se efectuan operaciones de escalamiento, traslación, rotación y perspectiva, para lo cual se

requiere de 20 operaciones de punto flotante para cada vértice de 3 dimensiones (9 sumas, 9 multiplicaciones y 2 divisiones).

Después de que los primitivos gráficos se convierten a 2 dimensiones se efectúa un recorte dentro de los límites de la pantalla, es decir los objetos que estan más allá de esos límites no se despliegan.

En esta etapa la imágen original de 3 dimensiones se ha convertido en un conjunto de instrucciones para 2 dimensiones en el sistema coordenado de la pantalla.

En la cuarta etapa los polígonos y las líneas se convierten en "pixels" de colores, que se guardarán en un area denominada memoria para rastreo de la imágen. A este paso se le llama rastreo y es el más simple de todo el proceso.

En primer lugar se clasifican los vértices del polígono en orden ascendente y se generan cada una de las líneas de búsqueda de la pantalla. El sistema calcula entonces la intersección de los lados del polígono con la línea para localizar el primer y último "pixel" que se encuentran dentro del polígono, modificando los "pixels" intermedios con un nuevo color.

Después de hecha esta conversión, los primitivos gráficos se representan como un arreglo de "pixels" de colores que pueden desplegarse en un monitor de TV o imprimirse.

Si además se requiere dibujar imágenes 30 veces por segundo se deben usar 2 "buffers", uno de los cuales despliega la imágen mientras que en el otro se borra y se dibuja la siguiente (Ej. Apple)

A continuación se describen a detalle cada una de las etapas anteriores para determinar los problemas de cada una de ellas.

2.4.1 TRANSFORMACIONES Y RECORTES.

Existen tres maneras de calcular las fórmulas de transformación:

1. Por medio de un programa que se ejecute en el procesador central. Ej. Intel 8086.

2. Con el uso del programa anterior y de un coprocesador de punto flotante. Ej. Intel 8087.

3. Mediante "chips" de propósito especial, para operaciones de punto flotante. Ej. AMD29325 y WTL1032.

Las transformaciones de 2000 vèrtices 30 veces por segundo con 20 operaciones de punto flotante por transformación, requiere de 1'200,000 operaciones de punto flotante por segundo ("flops") o sea .8 microsegundos por operación. Desafortunadamente los métodos basados en la programación del microprocesador principal son 2000 veces mas lentos y aun con el uso del coprocesador el desempeño es 20 veces más lento.

En los últimos años las compañias de semiconductores han empezado a ofrecer "chips" de hasta 5'000,000 "flops" a un precio razonable.

Los cálculos de recorte son más complejos, pero generalmente los primitivos gráficos se encuentran completamente dentro o fuera de la pantalla y por lo tanto no requieren procesamiento. En consecuencia el tiempo utilizado es pequeño con respecto al de las transformaciones.

2.4.2 RASTREO.

El rastreo es un proceso más difícil que las transformaciones y el recorte. Una simple imagen consta de cientos de miles de puntos cada uno de los cuales debe ser utilizado al menos una vez durante el proceso de rastreo.

Por ejemplo asumamos que la imagen a desplegar consiste de 1000 polígonos con una longitud promedio de 100 x 100 "pixels" cada uno. Esto se traduce en 10 millones de "pixels" por imagen, los cuales deben dibujarse 30 veces por segundo, por lo que deben dibujarse 300 millones de "pixels" en un segundo.

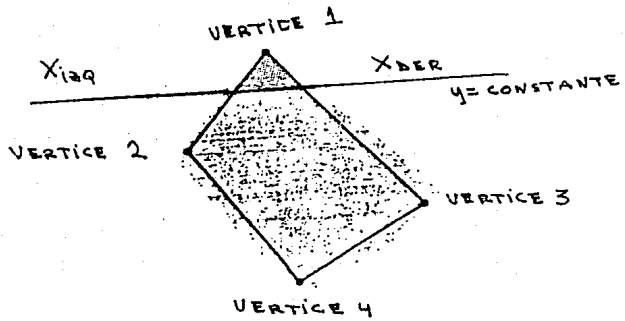


Fig 2.19 Para el rastreo de un polígono de cuatro lados se debe determinar donde se intersectan los lados del polígono con la

linea de búsqueda
Esto significa que un sistema de rastreo debe trabajar a velocidades menores de 3.3 nanosegundos por "pixel" (1/300,000,000) en promedio para tiempo real. Desafortunadamente el promedio de ejecución de un microprocesador es de 2000 nanosegundos por instrucción.

La mayoria de las tarjetas de gráficas disponibles para la IBM PC y otras marcas representan cada "pixel" como un "byte" de memoria mediante el cual se indica el color.

Haciendo una estimación optimista de que el programam en el microprocesador puede modificar un "pixel "por instrucción, se llevaria 20 segundos para rastrear todos los polígonos (10 millones de "pixel" a 2 microsegundos por "pixel"), esto es 600 veces más lento que lo necesario.

Para aumentar la velocidad se han introducido procesadores especiales para gráficos. El mejor conocido hasta ahora es el NEC7220. En este caso el procesador central envia las transformaciones geométricas de los primitivos gráficos al procesador de gráficas el cual utiliza cada "pixel" uno a la vez, pero a una velocidad menor de 800 ns. Rastrear los 1000 polígonos le llevaria 8 segundos que sigue siendo 240 veces más lento que el tiempo necesario para el desempeño en tiempo real.

El problema es que los "pixels" deben modificarse uno a la vez. Además la mayoria de estos procesadores tienen severas restricciones acerca del tipc de primitivos gráficos que pueden dibujar. Por ejemplo el NEC7220 sólo rellena rectángulos alineados con el eje de las x's.

En la tabla siguiente se muestran los tiempos estimados para el rastreo.

	"Pixels" por Ciclo de memoria	Tiempo promedio de acceso a un "pixel"	Tiempo promedio de rastreo de 10 millones de "pixels"	Tiempo promedio de rastreo de 300 millones de "pixels"	Imagenes obtenibles por segundo
Solo Programas "1 pixel/byte"	1	2000 ns	20 sec	600 sec	0.050
Procesador de Graficas "1 pixel/byte"	1	800 ns	8 sec	240 sec	0.125
Solo Programas "Bit Map (16 "pixel"s/16 bit world)"	16	125 ns	1.25 sec	37.5 sec	0.800

Procesador de Graficas "16 pixels/palabra"	16	20 ns	0.20 sec	6. sec	5.
Procesador de Graficas y "SLAM Chip"	1 a 1024 tipicamente 100	0.2 ns a 200 ns (tipicamente 2 ns)	0.02 sec	0.6 sec	50.

Fig 2.20 Tiempos de realizacion mediate varios métodos de rastreo de una imagen con 1000 poligonos de 100 por 100 "pixels" (10 millones de "pixels" en total).

Cuando el despliegue es en un monitor blanco y negro se necesita solo un "bit" por "pixel". Esto implica que se pueden almacenar 16 "pixels" en una palabra de 16 "bits" y utilizarlos todos simultáneamente con una instrucción. Este método se conoce como mapa de bits, y es usado por la Apple Macintosh.

Cuando se rellenan poligonos largos se permite una mejoria de 16 veces sobre el "frame buffer" de 8 "bits" por "pixel" para color. De todas maneras como la tabla indica, sigue siendo 37 veces más lento que el tiempo requerido. Esta estimación es muy optimista ya que los sistemas de mapeo de "bits" gastan tiempo en las orillas del polígono para determinar cuales de los 16 "bits" están afuera y cuales adentro.

Para poligonos pequeños predomina este proceso durante todo el tiempo de rastreo.

Recientemente se anunció un procesador gráfico que tiene la ventaja de manejar 16 "pixels" simultaneamente para despliegue de color, a este "chip" se le llama "QPDM (AM95C60 quad pixel data flow manager)". Este procesador puede manejar en la memoria de despliegue, hasta 16 "pixels" de 4 "bits" cada uno en un tiempo de 300 ns. Aun este dispositivo sigue siendo lento y de hecho es 6 veces más lento que lo necesario, para dibujar 1000 poligonos en tiempo real (30 imagenes de .02 segundos por imagen).

Como se observa el problema de llevar y traer los "pixels" a memoria es muy dificil. Existen sistemas que logran la velocidad necesaria para trabajo en tiempo real y son extensiones de las arquitecturas antes descritas. Estos sistemas usan procesadores gráficos que manejan muchos "pixels" a la vez, por ejemplo 64. Son muy caros y cuestan cientos de miles de dolares por lo que no son accesibles para el usuario promedio.

2.4.3 MANEJO DE LAS LINEAS DE BUSQUEDA EN MEMORIA.

El problema del rastreo puede resumirse como sigue: los sistemas de graficación necesitan manejar muchos "pixels" simultáneamente para rastrear poligonos y llenar las areas rápidamente, pero esto es muy caro.

El mejor lugar para manejar los "pixels" es en la memoria que los guarda. Considérese por el momento la estructura interna de un "chip RAM" convencional de 64 "kbits". Externamente la memoria aparece como 64k localidades de 1 "bit" cada una. Cada consulta a memoria se refiere solo a un "bit", pero la memoria en realidad es una matriz de celdas de 256 renglones y 256 columnas. Los primeros 8 "bits" de la dirección indican el renglón a ser utilizado y los otros 8 la columna correspondiente.

Cuando la memoria RAM se utiliza para guardar imágenes, cada renglón representa una linea de búsqueda y cada bit del renglón un "pixel". De esta manera cada "chip RAM" representa una porción de 256 lineas y 256 "pixels" de la imágen en cuestión . Se utilizarán muchos chips para representar una imagen larga con muchos bits por "pixel".

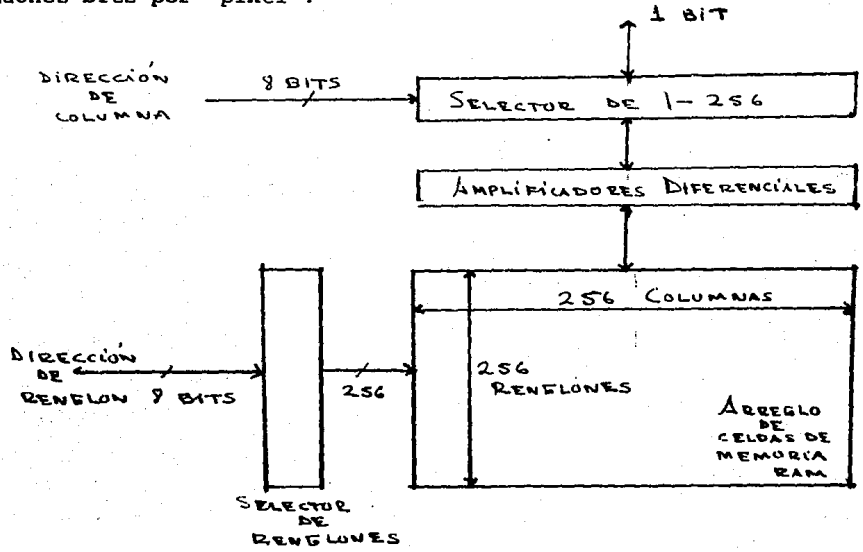


Fig 2.21 Estructura interna de un "RAM" convencional de 64 K-bit

Normalmente, cada vez que un "pixel" se modifica, la línea de búsqueda correspondiente se extrae de la memoria, pero sólo uno de los 256 "pixel"s puede salir del chip a la vez. Sería imprâctico permitir que los 256, bits de la memoria salieran simultaneamente, ya que se necesitaría un chip de 256 puntos de contacto.

De todas formas la línea de búsqueda está disponible adentro de la memoria. Se puede adicionar un procesador de propósito especial, el cual puede modificar partes de la línea de búsqueda, sin tener que mover los "pixels" grandes hacia afuera del chip.

En este concepto se basa una investigación realizada en la Universidad de Stanford por Stefan Demetrescu el cual desarrolló el llamado chip de memoria inteligente o "chip SLAM" ("scan line access memory chip").

El "chip SLAM" ejecuta los comandos que recibe de un dispositivo "bus" de 19 "bits". Los SLAMs son capaces de ejecutar directamente los comandos de llenado de línea horizontal que son la base de la función de llenado de polígono.

La operación de llenado se completa a través del uso de cuatro comandos de SLAM.

1. El primer comando indica cual es la línea de búsqueda a llenar.

2. El segundo comando especifica el patrón de 16 "bits" que se usara para iluminar todos los "pixels" que se modificarán.

3. El tercer comando especifica la coordenada X(derecha) e indica al SLAM que lea la línea de búsqueda seleccionada del areglo de memoria.

4. Finalmente el cuarto comando especifica la coordenada X(izquierda) e indica al SLAM que modifique todos los "pixel" dentro del rango de las coordenadas anteriores y escriba de nuevo la línea de búsqueda en el arreglo de memoria correspondiente.

Un sistema SLAM puede modificar de uno a muchos miles de "pixels" simultâneamente ya que maneja la línea de búsqueda completa y cualquier parte de ella puede modificarse en un ciclo de memoria.

En la figura de la página siguente se muestran las principales partes de un "chip SLAM" de 16 "kbit".

El "chip SLAM" consta de seis secciones primordiales.

1. La matriz convencional de memoria RAM.

2. La unidad aritmética y lógica "UAL Halftone", la cual realiza

las operaciones booleanas a la entrada del patrón.

3. El comparador paralelo que acepta los valores X(der) y X(izq), genera un bit 1 para cada una de las 256 posiciones horizontales que existen en el rango de las coordenadas anteriores y un bit 0 para todas las otras.

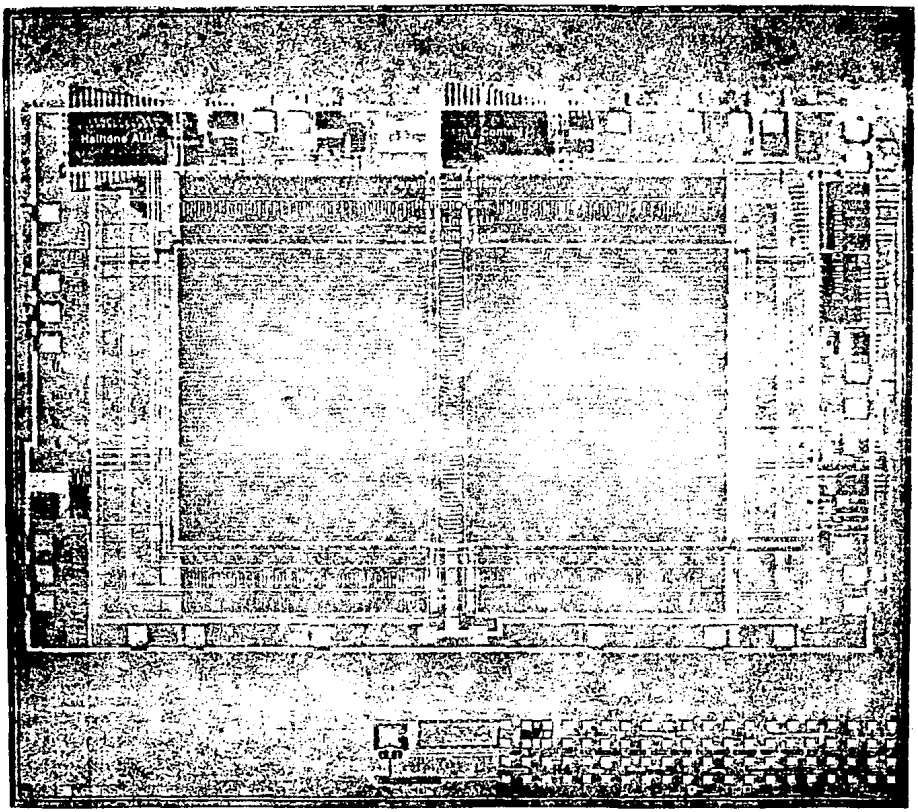


Fig 2.22 Fotografía de un chip SLAM

4. La UAL para lineas de búsqueda determina qué valores serán guardados nuevamente en la memoria de acuerdo al comparador paralelo y el "halftone bus".

5. El control Y y el selector de renglón habilitan el renglón adecuado de la memoria cuando se les proporciona el número de línea de búsqueda.

6. El registro de selección de despliegue bloquea la línea de búsqueda seleccionada para que se desplliegue independientemente del resto del "SLAM".

El "chip SLAM" puede modificar cualquier parte de una línea de búsqueda en un ciclo de memoria (200ns). Puede rastrear un polígono de 100x100 "pixels" en 100 ciclos de memoria (1 para cada línea de búsqueda) haciendo un total de 20 micro seg. (200 ns en 100 ciclos).

Como resultado, el sistema "SLAM" puede rastrear 1000 polígonos en 20 milisegundos y 50 imágenes por segundo, lo cual es más de lo necesario para animación en tiempo real.

Los chips SLAM no se encuentran disponibles en el mercado de los semiconductores comerciales aún.

2.4.4 EL SISTEMA DE DESPLIEGUE.

Hasta el momento se ha asumido que la memoria para imagenes puede ser usada exclusivamente para rastreo de imagenes primitivas. Esto es correcto si se usa un doble "buffer". Así una imagen se despliega mientras la otra se actualiza. El uso del doble "buffer" es escencial para que el observador no perciba el cambio de las imágenes.
El problema con el doble "buffer" es que requiere doble memoria. Esto puede ser significativo para el despliegue de 1000 x 1000 "pixels" de 8 bits cada uno (1 megabyte para cada uno de los "buffers".

En los sistemas de bajo rendimiento donde las imagenes no cambian rápidamente, una imagen se despliega y se modifica simultáneamente. Esto introduce otro problema, porque tanto el sistema de despliegue como el rastreador trabajan con la misma area memoria y esto lleva a una severa degradación de la función de rastreo.

Por ejemplo, la memoria de despliege de la Macintosh está ocupada en actualizar la pantalla el 50% del tiempo, y sólo el otro 50% esta disponible para otros procesos en el "RAM". Esto reduce el rendimiento en un factor de 2.

Como otro ejemplo el procesador de despliegue NEC7220 sólo permite cambios al "buffer" de la imagen durante el tiempo de trazado horizontal y vertical en la pantalla (50% del tiempo total).

En general se puede asumir que al menos el 50% del "buffer" para imagen se requiere para el sistema de despliegue si no se usa doble "buffer".

Recientemente los productores de "chips" de memoria han empezado a ofrecer "chips" especiales para resolver este problema, los cuales se les denomina "VIDEORAMS" o "Dual Ported RAMS" y están disponibles para Texas Instruments, NEC y AMD.

Como se muestra en la siguiente figura los "VIDEORAMS" son "RAMS" convencionales con un registro de transferencia de 256 "bits" adicionado.

Cuando un renglón de la memoria se despliega, el "VIDEORAM" localiza el renglón completo y guarda la información en el registro de transferencia. Los datos pueden transferirse independientemente de las operaciones de rastreo en la memoria. El registro de transferencia es un segundo puerto fuera de la memoria. Esto reduce efectivamente el número de consultas, de 256 a 1 para cada línea de búsqueda.

Estos "VIDEORAM" son formas primitivas de los "SLAMs", ya que pueden utilizar líneas de búsqueda pero sólo con el propósito de desplegar el contenido de la memoria.

Los "VIDEORAM" no pueden efectuar ninguna operación útil en el llenado de polígonos. Se ha dicho que los "VIDEORAM" solucionan el problema del rastreo, pero como se ha mencionado tienen ciertas limitaciones. Si el sistema tiene doble "buffer" estos "chips" ofrecen una pequeña o casi ninguna ventaja ya que la imagen desplegada está en una memoria para imagenes, diferente del lugar donde se va a llenar.

Si el sistema no es de doble "buffer" tiene en promedio el doble de rendimiento, pero no hacen nada absolutamente para resolver el problema del rastreo.

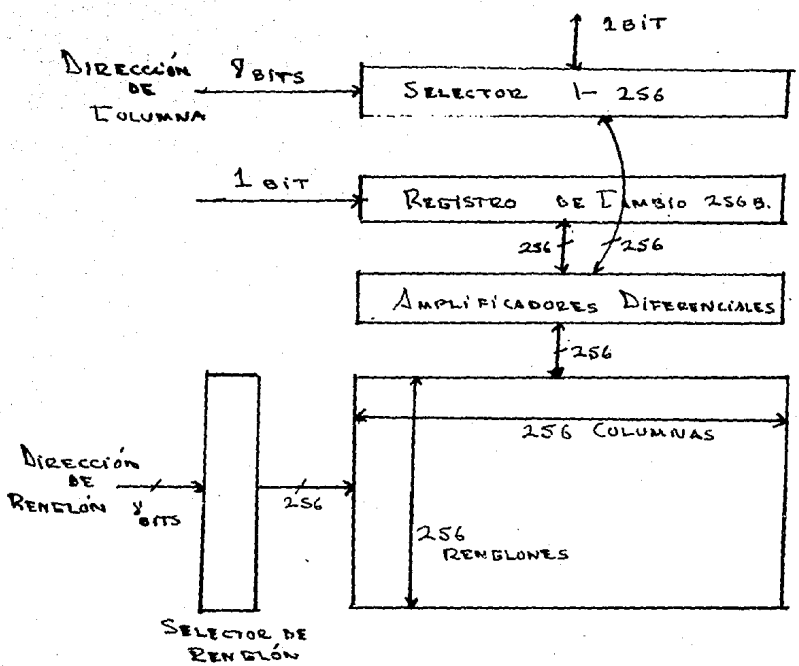


Fig 2.23 Estructura interna de un "video RAM" de 64 "K-bit", similar al RAM convencional con la adición de un registro de 256-"bit"

Mientras una mejora en el rendimiento de un factor de 2 no es spreciable, el mayor cuello de botella ocurre al ir llenando la memoria. Los sistemas pueden incrementar su rendimiento en el orden de 100 a 1000 veces en lugar de 2, incrementando la posibilidad de consulta a la memoria a través del uso de "SLAMs" por ejemplo.

Hasta aquí se ha planteado el grave problema que existe para obtener animación en tiempo real, manipulando figuras complejas.

Actualmente este tipo de animación se realiza a niveles mas simples con figuras sencillas y en algunos casos fondos fijos y poco color.
Esta animación se observa en los video juegos. La forma en que se realizan estos trabajos, será tratada con mayor detalle en el capitulo 4 .

Por ahora se tendrá que esperar nuevos adelantos técnicos que mejoren los tiempos de procesamiento y consulta a memoria para poder realizar animación de mejor calidad en tiempo real.

CAPITULO III
SISTEMAS DE ANIMACION DE ALTA TECNICA

3.1 Introducción

El objetivo de este capitulo es presentar la descripción de los principales sistemas de animación que existen, haciendo notar de que manera se han resuelto los problemas que se han presentado.

A pesar de que la animación por computadora se encuentra aun en sus primeras etapas de desarrollo, se han tratado de superar las barreras que representan la técnica y los lenguajes de programación disponibles.

En general puede decirse que las necesidades de programación para poder producir animación son:

1. Un lenguaje para construir y modificar figuras estáticas.

2. Un lenguaje para representar y especificar el cambio en las figuras y su dinámica.

3. Un conjunto de programas que transformen las especificaciones de la estructura de la figura y su dinámica en secuencias de imágenes visibles.

4. Un conjunto de programas que guarden y traigan de una memoria auxiliar esta secuencia de imágenes y faciliten su despliegue en tiempo real para observarlas en forma inmediata y transmitirlas a un medio de grabación permanente.

A lo largo del capitulo se podrán identificar estos elementos en los diferentes sistemas.

Como ya se mencionó los primeros intentos para usar las computadoras en este campo fueron en 1960, la animación por computadora se utilizó en principio para peliculas abstractas de tipo artistico. A principios de 1960 John y James Whitney produjeron secuencias de una pelicula realizada de manera muy sencilla, con la ayuda de una computadora analógica.

En 1964 Kenneth Knowlton de los Laboratorios "Bell", introdujo Beflex el primer lenguaje completo para dirigir animación, en blanco y negro. Beflex es una abreviatura de "Bell flicks for Bell Telephone Laboratories and Flickers".

"Beflex" llegó a ser sinónimo de animación por computadora en los años 60's.
Mas recientemente Tom DeFantis realizó el video abstracto a color "Espiral" y Larry Cuba's la pelicula "Dos espacios" de efectos geométricos en blanco y negro.

Se ha concluido que de todas las etapas de producción de una pelicula (entintado, opacado , pintado) la computadora es mas útil en la producción de cuadros intermedios. Muchas de las caricaturas producidas por Hanna Barbera en Hollywood, utilizan una computadora para este fin.

La producción de cuadros intermedios por computadora fue introducida a fines de 1960 en el sistema "Genesys", desarrollado por Ron Baecker en el Instituto Tecnológico de Massachusetts (MIT) en Cambridge.
Este es uno de los primeros sistemas en los que las imágenes y la animación se definen gráficamente, en lugar de usar un lenguaje de animación.

En Genesys el animador utiliza una tableta gráfica para introducir los datos y definir la ruta de movimiento (p-curva). El método de la p-curva es una técnica natural para especificar la ruta y la dinámica del movimiento.

Al usar la computadora para la producción de cuadros intermedios, es necesario que los dos cuadros de los extremos sean isomorfos (es decir deben ser la misma figura con diferente posición y escala asi se pueden generar los cuadros intermedios mediante rotaciones y escalamientos de la primera imágen.

Si los dos extremos no están basados en el mismo dibujo, el sistema requiere mas información.

Este tipo de procesos puede realizarlos el sistema desarrollado por Marceli Wein y Nestor Burtynk de Canada, el sistema fue utilizado en 1974 para la pelicula "hambre". En este sistema el animador introduce las lineas de la figura en un cierto orden. Cuando se introduce el siguiente cuadro, se compara el orden en el que las lineas se tracen para determinar la correspondencia entre los dos extremos, la primer linea será transformada gradualmente en la primer linea y asi sucesivamente, ambas figuras deben tener exactamente el mismo número de lineas.

En un sistema de cuadros clave la computadora representa cada linea como un conjunto de puntos, la computadora asocia cada punto en el cuadro fuente con el correspondiente en el cuadro destino, y asi podrá calcular los puntos intermedios.

Coloreado es el proceso de "pintar" la imágen en la memoria de la máquina y verla materializada en la pantalla.

Los programas de coloreado se usan normalmente en la creación de fondos. En la mayoria de los casos los fondos se hacen con mucho detalle ya que aparecerán durante toda la secuencia. Como el fondo no se mueve, se construye por separado del resto de las secuencias de animación.

Un sistema completo de animación debe contener un sistema caligráfico. Un ejemplo de esto es el Sistema para figuras de Evans and Sutherland en el cual el animador selecciona una brocha virtual (un pequeño patrón de "pixels") y un color de un menú en la pantalla. La tableta y la pluma se utilizarán como la brocha y el lienzo.

El animador puede utilizar brochas de diferentes medidas y formas simulando infinidad de efectos.

En la Universidad Cornell en Ithaca N.Y. se ha desarrollado un sistema en cooperación con los estuidios Hanna Barbera. El sistema provee la automatización parcial para las fases de entintado, opacado, dibujo de fondos y otras etapas de producción de una caricatura tradicional. Las técnicas manuales comunes se usan ra la creación de dibujos en papel, se transfieren a la computadora y se graban con una cámara de video

El artista ve la imágen en una pantalla de color. Mediante una tableta electrónica y una plumilla selecciona un color y lo utiliza para rellenar un área en la pantalla. Con sólo señalar un punto dentro de la región, el algoritmo la llena hasta sus bordes.

Comparado con las técnicas manuales, este algoritmo es muy rápido y reduce el tiempo de opacado por acetato en un factor de 5 o mas.

Un sistema similar desarrollado por Garland Stern en el Instituto de Tecnologia de Nueva York se usa en producciones como "Medida por medida ", que es una explicación animada de 22 minutos del sistema métrico.

Cuando se han pintado todos los acetatos se pasa al ensamblamiento o mezcla de todas las imágenes separadas, y a la aplicación de efectos de cámara.

Muchas de las restricciones en la producción tradicional de caricaturas son consecuencia del equipo y la técnica utilizados para el ensamblamiento. Normalmente el equipo para grabación es un cámara montada en una tabla con rotación y movimiento en las

cuatro direcciones.

Como todos los acetatos están en el mismo plano es muy difícil mantener la ilusión de profundidad cuando la cámara se mueve, ya que el fondo y el frente están a igual escala.

Este problema fue enfrentado por Walt Disney en 1936, quien inventó la cámara multiplano. En esta cámara se colocan los dibujos en planos de vidrio individuales a diferentes alturas arriba de la cámara. Cada nivel se puede manipular independientemente y se crea una convincente ilusión de espacio tridimensional.

Para la producción de modelos espectaculares en 3D se fotografían objetos tridimensionales.

El problema se divide en dos etapas: la creación del modelo en 3D y la animación del mismo.

Las secuencias demandan un alto grado de realismo con modelos muy complejos. Una vez que se ha construido el modelo, se coloca en la posición deseada y se fotografía un cuadro. Entonces se cambia la posición del modelo por fracciones de una pulgada y se toma otro cuadro. En algunos casos se produce menos de un segundo de película en un día completo.

Algunas partes de "Tron" fueron modelados con "MAGI INC's Synthavision systems". Este sistema usa modelos construidos a partir de sólidos (esferas, elipses, cilindros, conos y polígonos).

La animación del modelo se hace de diferentes formas. Las técnicas interactivas se usan para crear las posiciones claves del modelo, y los cuadros restantes se calculan por procesos de interpolación en tres dimensiones. Un enfoque diferente es tomado en el "Actor/Scriptor Animation System". El sistema de animación actor/escritor, también utilizado en Tron toma un enfoque diferente.
Este sistema controla la animación mediante un libreto escrito en una extensión gráfica especial de Lisp, la cual incluye tipos de datos geométricos como vectores , polígonos, sólidos, color, luz y puntos de visión.

En suma, el sistema está provisto de un conjunto de operaciones geométricas (mover, rotar) para posicionar y orientar los objetos y los tipos de datos números llamados Newtons.

El Newton se usa para describir parámetros dinámicos como el ángulo de rotación de un objeto durante una secuencia.

En el grupo de investigación del area de graficación por computadora de la Universidad de Ohio, se usan procedimientos tipo

"modelo", para describir las escenas tridimensionales.

Por ejemplo para crear un bosque se tiene un procedimiento que llama repetidamente al procedimiento árbol, el cual en turno, llama al procedimiento ramas y asi sucesivamente. El nivel mas bajo de procedimientos, es un simple generador de poligonos. Cambiando unos cuantos párametros, el animador puede utilizar los mismos procedimientos para crear una gran variedad de bosques.

La generación de animación en tres dimensiones se produce más facilmente sin tomar en cuenta las leyes físicas.

La falta de restricciones físicas puede sin embargo, crear algunos problemas, ya que si no se toman precauciones dos objetos pueden pasar uno a través de otro.

Algunas de estas dificultades se superan con un sistema restringido de animación es decir mediante un conjunto de leyes físicas artificiales.

Otra posibilidad es el uso de una cámara virtual en tres dimensiones. En un programa de cámara virtual la computadora despliega los datos como si se vieran desde cualquier punto del espacio. Muchos de los movimientos de la cámara pueden simularse cambiando el punto de visión y el ángulo.

Una técnica desarrollada por Michael Potmesil e Indranil Chakravarty en el Instituto Politécnico Rensselaer simula la cámara virtual como un dispositivo óptico con lentes y apertura. Esta técnica permite enfocar desde un plano arbitrario, incluyendo nube de movimiento y la simulación de efectos especiales.

Un gran problema es la memoria necesaria para almacenar todo esto. También existe el problema del escalonamiento en las figuras, clásico de una pantalla tipo rastreador. Existen algoritmos eficientes para procesar solo la porción visible de una escena 3-D, tales como el algoritmo z-buffer el cual toma en cuenta la distancia a la que se encuentra cada punto del observador y permite la eliminación del escalonamiento.

En la Universidad del estado de Ohio, se desarrollan las técnicas de antiescalonamiento. Estos algoritmos efectuan operaciones de promedio equivalentes a las aplicadas en un filtro digital para remover los componentes de alta frecuencia de la figura.

El grupo de Cornell a atacado este problema incluyendo en sus algoritmos de mezcla, una técnica especial que realice el antiescalonamiento mientras se ensamblan varias imágenes.

El sistema es utilizado por los estudios Hanna Barbera en Hollywood. Se simula el efecto de alta resolución sin el alto costo

asociado con los algoritmos mas generales.

Los problemas de antiescalonamiento, sombreado, iluminación, textura y movimiento de los cuerpos, siguen explorándose.

Un ejemplo de como se puede aplicar la animación en tres dimensiones es el enfoque tomado para la película "Los oficios " del Instituto de Tecnología de Nueva York.

La construcción del modelo en tres dimensiones empieza cuando el modelador, prepara una serie de diseños los cuales se mejoran hasta que permiten la creación de vistas detalladas para dibujos de ingeniería. El programador puede crear una base de datos computacional para cada componente construyendo archivos que contengan las superficies primitivas y los parámetros de transformación (traslación, escalamiento, rotación).

Una vez que la estructura física de los componentes se ha descrito satisfactoriamente, se adiciona un alto grado de realismo mediante las técnicas de modelado de superficies. Estas técnicas incluyen textura (mapeo de textura a 2-D en una superficie de 3-D) y mapeo de textura en 3-D en una superficie de 3-D, color e iluminación donde se describe las reflexion y brillantez necesaria.

Para movimientos mas complejos la animación por cuadros clave no es factible y deben usarse las técnicas matemáticas.

Un ejemplo es la secuencia de animación de "Los oficios" en la cual la hormiga constructora tiene que caminar a través de un terreno irregular. Puede ser muy complejo posicionar cada uno de los siete segmentos de las seis patas de la hormiga manualmente para cada cuadro. En lugar de esto, el animador especifica el camino a través del terreno y una rutina de postproceso, matemáticamente determina la posición correcta de los 42 segmentos de las patas.

Algunas de las cosas mas espectaculares se ven por la televisión de E.U. Este mercado está dominado por un pequeño número de casas especializadas notándose Abel y asociados, Acme, Creaciones por computadora, Efectos digitales , Información Internacional INC. , MAGI, y Marks y Marks.

El costo de la producción por segundo puede ser mucho mas alto que en la animación tradicional pero los resultados que se ofrecen son incomparables. En muchos casos, la apariencia final se logra mediante la combinación , de efectos de computadora, acciones de la vida real, efectos ópticos y otros.

La animación por computadora es mas que una combinación de equipo y programas en una forma de arte y destreza y la destreza se adquiere a través de entrenamiento artistico y experiencia. Los sistemas mas exitosos deben desarrollarse con la colaboración de artistas con experiencia . En el futuro se deben enfocar esfuerzos hacia el diseño de sistemas para artistas que son completamente ajenos a la tecnologia y a la programación.

A continuación se presenta una descripción detallada de varios sistemas de animación ya implementados que utilizan diferentes técnicas. El orden en el que aparecen es cronológico, todos los que se presentan son sistemas de animación cuadro por cuadro. Los sistemas de animación en microcomputadora, se presentan en el capitulo siguiente.

3.2. ANIMACION DIRIGIDA. GENESYS 1969

3.2.1 UN EJEMPLO EN GENESYS

GENESYS es un sistema implementado en el laboratorio Linconln del MIT en una computadora TX-2 .

Para ilustrar su uso, se muestra una secuencia dinámica de un perro corriendo hacia su comida. El perro corre hacia un recipiente, mueve la cola, baja la cabeza y empieza a tomar la leche. Se mostrarán varios languetazos antes de cortar a la siguiente escena. El proceso se desarrollará como sigue:

ANIMADOR (A): CALL GENESYS;

GENESYS (G) : HELLO.GENESYS WAITS YOUR CREATION;

A: FORMMOVIE DINNERTIME;

Va a crear una secuencia llamada "dinnertime"

G: FRESH

Indica que el nombre "dinnertime" se usa por primera vez

A: FORMBACKGROUND;

Se va a definir una figura (un fondo) que va a ser visible durante toda la secuencia

G: SKETCK IT, MAN;

A:El animador esboza el tazòn con la plumilla en la tableta.

G:OK;

A:FORMCELL No.1 in CLASSBODY

Se va a dibujar una versiòn inicial del cuerpo del perro

A:BIND BODY, LEGS, TAIL, HEAD, TONGUE;

Garantiza que cualquier movimiento aplicado al perro moverá el cuerpo, las patas la cola y la lengua juntos, asi el perro no se desintegra al moverse.

G:OK;

A:SKETCHPCURVE BODY;

El animador indica mediante la plumilla la ruta que se desea para el movimiento (El acto de imitar un movimiento continuo se llama p-curva).

A:PLAYBACK;

Indica que desea ver la versión actual del movimiento. En la pantalla se desliza un perro rígido hacia el tazón

A:FORMCEL No.2 in CLASSLEGS:

El animador dibuja las patas en otra posición y las define en una segunda clase de celdas llamadas "LEGS". Esto puede hacerse para varias posiciones

A:TYPESELECTIONS from LEGS;

Escribe una sucesión de opciones de las posiciones de las patas. Para cada opción se indica en cual celda se va a desplegar el perro con las patas en esa posición.

A:PLAYBACK;

Se muestra la versión actual, ahora las patas se mueven cuando el perro va al tazón.

A:TYPESELCIONS from TONGUE;

Ahora se diseñan las posiciones de la lengua para que el perro pueda comer la leche.

A:TAPRHTYHM SLURPINTERVALS;

Se puede intuir el ritmo al cual se desea que el perro coma y se irá indicando por medio de los botones.

A:REPEATPATTERN FORM frame 59 TRHOUGH frame 64 of SELECTIONS from TONGUE at INTERVALS of SLURPINTERVALS;

Asume que los languetazos ocurren en los cuadros 59 a 64. El patrón de lenguas seleccionado se repite a intervalos determinados por el ritmo especificado.

A:PLAYBAKC

Ahora el perro va hacia el tazón y languetea.

A:EDIT X WAVEFORM of BODY

Se requiere una mayor aceleración en el movimiento del perro . El animador despliega la coordinación del perro contra el

tiempo y rediseña parte de la forma de la onda.

A:EDIT FRAME 44;

Se asume que el perro llega al tazón en el cuadro 44 y entonces hay que ver disminuir la velocidad.

A:FIX X and Y of BODY AFTER frame 44;

Hay que detener al perro una vez que llega al tazón

A:PLAYBACK

Se despliega la versión actual

A: SAVE DINNERTIME
Se graba la secuencia

G:DINNERTIME IS SAVED . GOOD BYE

En el escenario anterior:

1.Se generan aproximadamente 100 cuadros. Se usaron herramientas simples como son los programas que permiten el diseño directo y la correción de las partes de la figura

2.Puede verse la secuencia inmediatamente permitiendo ensayo de prueba y error.

3.Por medio de imágenes estáticas se representa la relación entre el tiempo y los parámetros del movimiento de la figura.

4.El animador puede imitar aspectos dinámicos en tiempo real. El ritmo y movimiento se graban para su aplicación posterior.

3.2.2 GENERACION DE SECUENCIAS

En este sistema se utilizan varias descripciones para la generación de una secuencia de cuadros. Estas descripciones son algoritmicas y se presentan a continuación.

Descripción dinámica global

Una descripción dinámica global es una secuencia de datos cuyos elementos sucesivos determinan los parámetros criticos en los cuadros sucesivos del movimiento. El animador define las representaciones de las descripciones dinámicas, estas secuencias de datos, dirigen el algoritmo para generar una despliegue de

animación.

Una descripción dinámica global es una descripción de movimiento la cual es una descripción de movimiento continuo= descripción de ruta o una descripción de movimiento discreto= una selección de descripción o una descripción de ritmo

Descripción de ruta

Las modificaciones a figuras estáticas consisten en el cambio de sus parámetros como localización ó medida e intensidad. La figura estática se animará especificando el comportamiento temporal de esos parámetros.
Una representación del comportamiento temporal de parámetros que varian continumente se denomina descripción de ruta.

Las descripciones de ruta se hacen en cualquiera de las siguientes formas :

1. Diseñando una nueva representación de figura.

2. Modificando una representación existente.

3. Especificando directamente el algoritmo de la secuencia de datos

4. Especificando indirectamente el algoritmo en términos de una secuencia de datos existente.

5. Combinando procesos físicos reales que se trasmiten por un medio de entrada a la computadora

Considérese el movimiento de una figura que va de una esquina de un cuadro a la esquina diagonalmente opuesta, caminando a lo largo de dos paredes adyacentes. Ignórese el movimiento vertical y considérese solamente el movimiento del centro del cuerpo. Primero camina en la direccion de la coordenada X positiva y luego en la dirección de la coordenada Y negativa, se asume que parte del reposo, luego acelera y desacelera hacia la primera esquina se hace un breve intervalo mientras toma su lugar y finalmente acelera y desacelera hacia su destino.

Una descripción completa del movimiento en el plano consiste en las funciones de las coordenadas X y Y contra el tiempo. La representación de los parámetros de cambio de la figura se denomina "formas de onda". El tiempo es "pintado" en la onda a lo largo

del espacio dimensional. La construcción de la onda se hace mediante el movimiento de la plumilla. La pantalla registra y hace tangible este movimimento.

En resumen una descripción de ruta define la actividad dinamicamente. Una descripción de ruta puede determinar el cambio en la localización , intensidad, densidad, grosor y textura.

Descripciones de ritmo.

Es una sucesión de tiempos de despliegue o intervalos entre cuadros. Las descripciones de ritmo facilitan la coordinación y sincronización entre actividades dinámicas.

Una descripción de ritmo no define por si misma el cambio en la figura, define el compás. Es un conjunto de señales respecto a las cuales se organiza temporalmente el cambio en la figura.

3.2.3 RESUMEN

Este sistema se implementó en el laboratorio Lincoln del MIT en una computadora TX-2 .

Consta de un modo de construcción o edición, un modo de despliegue y un modo de filmación.
Las imágenes son enviadas al programa de despliegue el cual simula una velocidad variable y una grabadora de video.

El sistema hace los cuadros visibles a un 1/24 de segundo. Cuando la secuencia está lista se pasa a filmación y se conecta una cámara a la computadora. El programa de filmación despliega la imagen por aproximadamente 1/5 de segundo, la cámara se avanza mediante una señal de la computadora.

El animador "GENESYS" permite diseño, borrado, copiado, traslación, rotación, escalamiento. Permite el diseño de P-curvas y la descripción del ritmo. Algunas personas relacionadas con el arte y la animación han construido secuencias de caricaturas con el apoyo de GENESYS.

3.3 ANIMATOR : UN SISTEMA BIDIMENSIONAL DE ANIMACION DE PELICULAS

Este sistema se desarrolló en 1971, los autores expresan que los sistemas de animación, tienen limitaciones significativas que restringen sus aplicaciones diarias. Animator elimina dos de las deficiencias mas serias:

1)Provee retroalimentación del sistema al animador humano
2)Elimina muchas de restricciones de los lenguajes de programación convencionales.

Muchos sistemas de animación convencionales, tienen una gran falta de retrolimentación durante el proceso de especificación de movimiento. La razón para esta falta de retroalimentación es que la definición del movimiento siempre se traslada a un forma intermedia o final antes de que pueda verse en su forma original.

Como resultado se tiene un retraso significativo entre el tiempo de especificación del movimiento y el tiempo en donde se obtiene información de retroalimentación y esto decrementa la efectividad del programador.

Por otra parte generalmente el animador no cuenta con todas las facilidades para la animación (una computadora digital y una grabadora de peliculas) .
Tipicamente la grabadora está en un lugar diferente al lugar donde está la computadora y el proceso de animación se retrasa considerablemente. Una cinta magnética contiene las instrucciones de máquina para la grabadora y debe enviarse al lugar en donde ésta se encuentra.

Como es necesario efectuar corridas de prueba antes de lograr el resultados final, este traslado se repite varias veces para cada escena lo cual retraza aun mas la producción.

Para resolver este problema muchas instalaciones que no tienen una grabadora de peliculas, cuentan con programas que producen salidas gráficas en un graficador o en una impresora utilizando los mismos comandos que para producir la pelicula de esta manera sólo se envia la versión final.

Generalmente se tienen que usar lenguajes de programación convencionales como Fortran los que comunmente son extraños a los artistas o animadores.

Lo ideal es que se cuente con un sistema de tiempo compartido que permita la entrada de datos mediante una pluma de luz ó de alguna forma mas familiar al artista.

El equipo donde se implementó este sistema es una DEC-338 con un disco DF32 y un teletipo ASR 33, un controlador de cinta TC01 DEC y una interfase a una IBM 360/75, esta interfase se lleva a cabo mediante una linea telefónica "duplex". Como en la Universidad de Pensylvania no se cuenta con la grabadora, la cinta se envia a Brooklin para su procesamiento en una Stromberg Datagraphix 4020.

3.3.1 DESCRIPCION DEL SISTEMA

La DEC-338 sirve como terminal de entrada y permite un diseño de prueba y error de las secuencias de figuras en modo conversacional. Durante todas las etapas de especificación del movimiento, se lleva un registro de las acciones del usuario en los elementos de entrada. Este registro puede enviarse a la IBM 360/75 donde el S-D 4020 da las instrucciones necesarias para generar la misma secuencia de figuras.

La implementación actual de Animator consta de 3 programas. Uno de ellos permite el uso de la DEC-338 como terminal de entrada y es el único con el que el usuario interactua. Otro programa transmite el registro de las acciones del usuario en los elementos de entrada, de la DEC 338 a la IBM 360. El tercer programa está implementado totalmente en la 360. Toma el registro y utiliza el paquete SCORS que produce las instrucciones S-D 4020 necesarias para generar el movimiento que el usuario prescribió.

El papel del programa de la 338 es:
1.Monitorear las acciones del usuario en los elementos de entrada asociados con el 338.
2.Desplegar las definiciones del usuario en el 338.
3.Mantener un registro de la cola de entrada de usuarios.

Para esto último se desarrolló un lenguaje llamado "lenguaje de transmisión" en el cual todas las acciones del usuario se trasladan inmediatamente antes de ser reconocidas por el sistema. El archivo de acciones del usuario se compila en el "lenguaje de transmisión" y se genera el archivo de transmisión. La función del programa de la 360 es entonces, traducir el archivo de transmisión en instrucciones S-D 4020.

En muchos casos los programas implementados en la 338 y en la 360 se usan para efectuar cálculos similares. Por ejemplo se pueden tener rutinas que dado un centro y un radio, generen el código de máquina necesario para desplegar un circulo.

3.3.2 ELEMENTOS DE UNA PELICULA

Una pelicula animada en este sistema consta de 4 elementos:

figuras, movimientos, escenas y segmentos de movimiento.

Una figura se compone de elementos de un conjunto de primitivos gráficos que provee el sistema o de figuras previamente definidas. En cualquier momento durante el proceso de especificación del movimiento puede definirse una nueva figura y adicionarse o cambiarse cualquiera de las ya existentes.

Un movimiento es un operador aplicado a una figura y puede definirse en términos de primitivos de movimiento provistos por el sistema o por otros previamente definidos. Un movimiento puede definirse como paralelo o secuencial. Un movimiento paralelo es aquél en el que se aplica mas de un operador de movimiento a una figura en el mismo tiempo Por ejemplo un movimiento paralelo definido como una rotación y una traslación causará un giro y un movimiento a través de la pantalla simultaneamente.

Un movimiento secuencial es aquél en el que no sólo se indican las acciones, sino la duración de estas. Un movimiento secuencial describe los primitivos o definiciones de movimiento que serán aplicadas en secuencia a la figura.

Un movimiento secuencial definido en términos de rotación y traslación aplicado a una figura causará que ésta rote un cierto número de cuadros y se traslade a través de la pantalla en otro cierto número de cuadros.

En cualquier momento en las especificación del movimiento pueden definirse nuevos movimientos o modificarse los ya existentes.

Una escena consiste de una o mas figuras y el movimiento asociado con ellas.

Un segmento de movimiento se compone de al menos una escena y/o un segmento de movimiento definido previamente y describe como se integrarán las escenas en orden secuencial.

Una vez que el usuario construye las definiciones de las guras y de las secuencias, puede especificar las escenas o segmentos que quiere producir y enviarlas a la IBM/360 para producción.

3.3.3 MODOS DE OPERACION

ANIMATOR se opera en 6 modos diferentes:

1. El modo de definición de figura

2. El modo de definición de movimiento
3. El modo de definición de escena
4. El modo de definición de segmento de movimiento
5. El modo producción
6. El modo transmisión

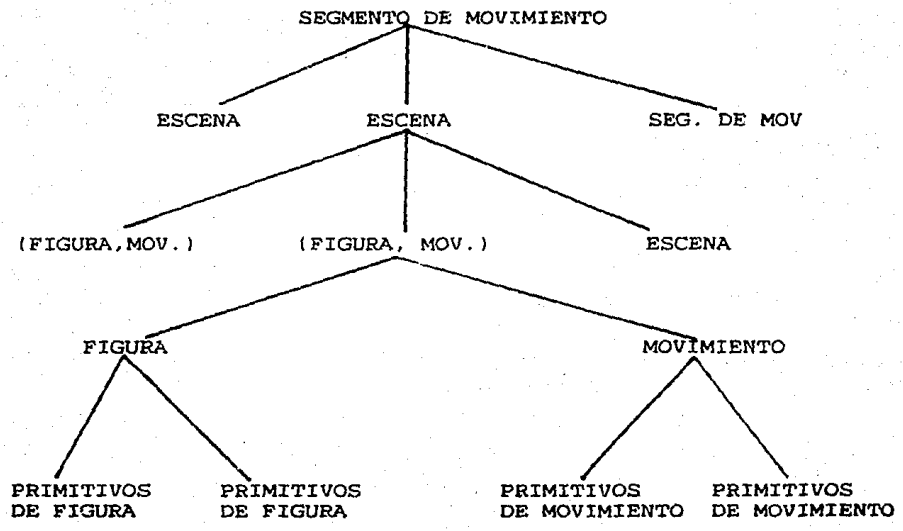


FIG. 3.1 Representación estructural de una película.

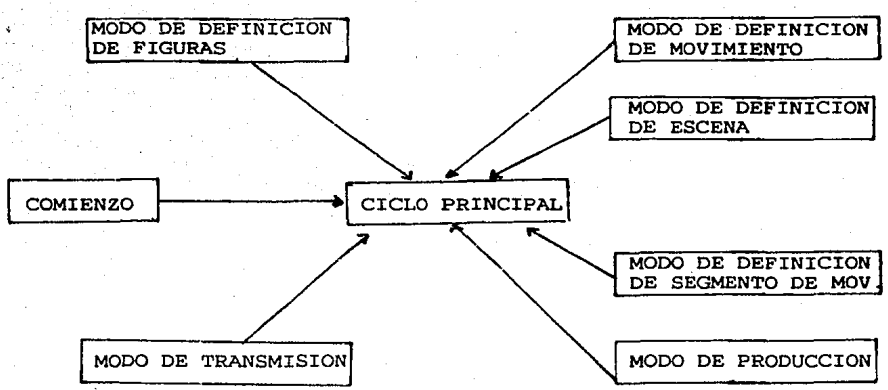


FIG. 3.2 Estructura del programa

Al principio del programa el usuario puede escoger el modo en el que desea trabajar.

Modo de definición de figuras

Este modo se usa para la construcción de objetos gráficos estáticos.

Una definición de figura consiste de un nombre asociado con una sucesión de primitivos graficos y/o figuras previamente definidas.(No se permite recursividad en las definiciones).
La sintaxis de la definición de figuras puede describirse en forma normal de Backus como sigue :

⟨figura⟩::= ⟨nombre⟩⟨descripción de figura⟩

⟨descripción de figura⟩::=⟨primitivos de figura⟩ ⟨figura⟩⟨descripción de figura⟩

⟨primitivos de figura⟩::=CLEAR SCREEN | FIX INTENSIFIED HORIZONTAL OR VERTICAL LINE
FIX UNINTENSIFIED HORIZONTAL OR VERTICAL LINE | FIX INTENSIFIED LINE | FIX UNINTENSIFIED LINE
DRAW FOLLOWER LINE | CIRCLE ENTER COORDINATES | TEXT ERASE LINE | ERASE LINES | REMOVE SUBPICTURE
POINT | SPECIFY LENGTH | DUPLICATE LENGTH

⟨nombre⟩::= ⟨caracter⟩ | ⟨caracter⟩ | ⟨caracter⟩ | ⟨caracter⟩ | ⟨caracter⟩|⟨caracter⟩

⟨caracter⟩::= A | B | ....Z | 0 ...9 | * | + | * | ( )

Modo de definición del movimiento

Se utiliza para la construcción de los operadores que se aplican a las figuras. Para permitir la máxima flexibilidad, pueden definirse como paralelos o secuenciales.

El número de cuadros durante los cuales se aplica un operador a una figura se limita solo por el tiempo de uso del operador en la definición de movimiento secuencial. Los operadores usados en movimiento paralelo no tienen ninguna prescripción de tiempo de duración .

Un movimiento paralelo consiste de un nombre asociado con una secuencia de primitivos de movimiento o definiciones previas. Una definición de movimiento secuencial consiste de un nombre asociado con un primitivo de movimiento secuencial o definiciones previas. (No se permite recursividad).

La sintaxis de definición de movimiento es :

⟨movimiento paralelo⟩::= ⟨nombre⟩⟨descripción de movimiento paralelo⟩ | ⟨primitivos de movimiento⟩

⟨movimiento secuencial⟩::=⟨nombre⟩⟨descripción de movimiento secuencial⟩

⟨descripción de movimiento paralelo⟩::=⟨primitivos de movimiento⟩ | ⟨movimiento paralelo⟩ | ⟨descripción de movimiento paralelo⟩⟨descripción de movimiento paralelo⟩

⟨descripción de movimiento secuencial⟩::=⟨movimiento paralelo⟩ | ⟨contador de cuadros⟩ | ⟨movimiento secuencial⟩ ⟨descripción de movimiento secuencial⟩⟨descripción de movimiento secuencial⟩

⟨primitivos de movimiento⟩::=ERASE DEFINITION | HOLD | TRANSLATE | ROTATE ABSOLUTE ROTATE RELATIVE ZOOM ABSOLUTE | ZOOM RELATIVE | FILL | BLINK

⟨contador de cuadros⟩::= Un número decimal entre 1 y 2000

Modo de definición de escena

Se utiliza para especificar las figuras que aparecen simultáneamente en una porción del movimiento, la posición absoluta inicial de estas figuras y su movimiento secuencial. La duración de la escena se controla por medio del contador de cuadros asociado con el movimiento secuencial. Una definición de escena consiste de un nombre asociado con una sucesión de descripciones de escena básicas y/o otras existentes.

La sintaxis es :

⟨escena⟩::=⟨nombre⟩⟨descripción de escena⟩

⟨descripción de escena⟩::=⟨descripción básica de escena⟩ | ⟨escena⟩ | ⟨descripción de escena⟩ ⟨descripción de escena⟩

⟨descripción básica de escena⟩::= ⟨X⟩⟨Y⟩ ⟨figura⟩ ⟨movimiento secuencial⟩

⟨X⟩::= un número decimal entre 0 y 1023

⟨Y⟩::= ⟨X⟩

Modo de definición de segmentos de movimiento

Se utiliza para describir una pelicula o un segmento de la misma especificando las relaciones temporales dentro de la escena . La definición de un segmento de movimiento consiste de un nombre asociado con una secuecia de escenas y/o definiciones previas.

La sintaxis es la siguiente:

⟨segmento de movimiento⟩::=⟨nombre⟩⟨descripción de segmento de movimiento⟩

⟨descripción de segmento de movimiento⟩::=⟨escena⟩ | ⟨segmento de movimiento⟩ | ⟨descripción de segmento de movimiento⟩ ⟨descripción de segmento de movimiento⟩

Modo producción

Se utiliza para especificar los nombres de las escenas ó segmentos de pelicula que se desean producir en la 360.

Modo Transmisión.

Se usa para transmitir las acciones relacionadas con las escenas y segmentos de movimiento que se van a producir. Aunque el sistema fue diseñado orginalmente para la producción de peliculas de 16 o 35mm o para salida a la S-D 4020, el usuario puede producir gráficos en el Calcomp 565 o en una impresora.

3.3.4 UN EJEMPLO DEL USO DE ANIMATOR

En este ejemplo se muestra como usar ANIMATOR para generar la secuencia que se observa en la figura de la página siguiente.

Aparece una U para las acciones del usuario y una A para las del sistema ANIMATOR.

1.U. Teclea Animator

2.A. Aparece el menú principal en la pantalla 338.

3.U. Selecciona "PICTURE DEFINITION" con la pluma de luz

4.A. Entra a modo "definición de figura"

5.U. Asigna el nombre OLAS a la figura y mediante la pluma de luz y los botones dibuja en la pantalla las olas

6.U. Teclea BARCO y lo dibuja de la misma forma

7.U. Oprime los botones necesario para regresar al menú principal

8.A. Aparece el menú principal

9.U. Selecciona la opción "MOTION DEFINITION"

10.U. Entra a modo "definición de movimiento"

11.U. Teclea el nombre ROCK1 y utiliza una combinación del teletipo, pluma de luz, y los botones para construir la siguiente definición :

ROCK1= TRANSLATE con ΔX/cuadro=88 & ΔY/cuadro=0 y ROTATE RELATIVE con ΔX=0 & ΔY=0 & Δθ/cuadro=-25

12.U. Teclea ROCK2 y construye la siguiente definición:
ROCK2= TRANSLATE con ΔX/cuadro=128 & ΔY/cuadro=-64 y ROTATE RELATIVE con ΔX=-0 & ΔY= 0 & Δθ/cuadro= 50

13.U. Teclea ROCK3 y contruye la siguiente definición:

ROCK3=TRANSLATE ΔX/cuadro=128 & ΔY/cuadro=64 y ROTATE RELATIVE con
ΔX=0 & ΔY=0 & Δθ/cuadro= -50

14.U. Oprime los botones necesarios para iniciar una definición de un movimiento secuencial , teclea STILL y construye la siguiente definición

STILL=HOLD con un contador de cuadros = 6

15.U. Teclea ROCK23 con la siguiente definición

ROCK23=ROCK2 con contador de cuadros =1 seguido por ROCK3 con contador de cuadros =1

16.U. Teclea un nueva definición de movimiento secuencial. Teclea ROCK con la siguiente definición

ROCK=ROCK1 con contador=1 seguido por ROCK23 seguido por ROCK23 con contador =1

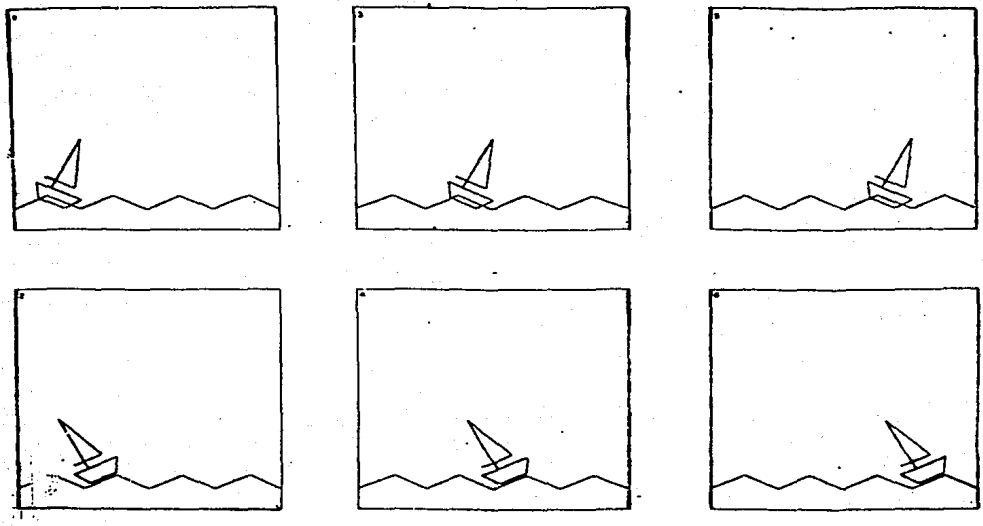


FIG. 3.3 Secuencia de cuadros creados por ANIMATOR

17.U. Regresa a menú principal

18.A. Despliega menú principal

19.U. Selecciona "SCENE DEFINITION"

20.A. Entra a modo "definición de escena"

21.U. Define una escena.
Teclea SCENE. Utilizando los botones teletipo y pluma de luz construye la definición de la escena en términos de la figura OLAS centrada en el punto (512,544) con movimiento asociado STILL

y la figura BARCO centrada en el punto (0,244) con movimiento asociado ROCK.

22.U. Regresa a menú principal

23.A. Despliega menú principal

24.U. Selecciona "PRODUCTION MODE"

25.A. Entra a "modo producción"

26.U. Utilizando el teletipo y los botones indica que la escena SCENE será producida

27.U. Regresa a menú principal

28.A. Despliega el menú principal

29.U. Selecciona "TRANSMISSION MODE"

30.A. Entra a "modo transmisión"

31.U. Selecciona el Calcomp con la pluma de luz e inicia la transmisión.

3.4 ANIMA II SISTEMA DE ANIMACION A COLOR EN 3-D.

Anima II es un sistema diseñado para la producción de peliculas tridimensionales a color. Está orientado al animador, al educador y al artista, Anima II provee un ambiente propicio para la creación, animación y despliegue en tiempo real de poliedros sombreados, la salida de la pantalla está directamente conectada a un equipo de grabación de video y a un televisor. El sistema fue diseñado por la compañia "Computer Graphics Reseach".

Este sistema fue desarrollado en una PDP 11/45 con 64K de memoria principal (32 de los cuales contienen el RSX11-D) y 32K de memoria MOS. Los periféricos usados fueron una pantalla de refes-camiento de 4096 x 4096 con palanca de control, un disco fijo de 44 "Megawords", un decodificador de propósito especial de color de rastreo que sirve como interfase de video en tiempo real. Los programas se desarrollaron en lenguaje de ensamblador.

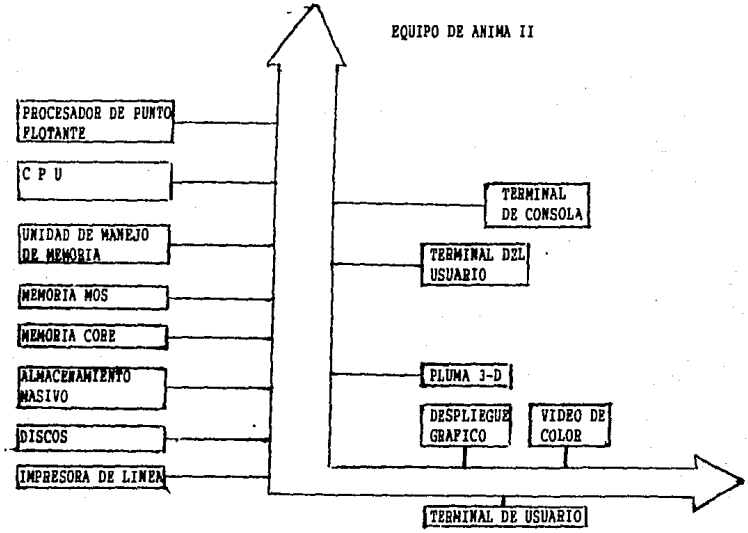


FIG. 3.4 Equipo fisico que utiliza ANIMA-II

El sistema se creó de tal manera que un usuario ajeno al area de computación sea capaz de:

- Crear poliedros complejos con modelado geomètrico interactivo.

- Escribir un libreto de animación para definir el movimiento de los objetos.

- Animar el libreto utilizando un procesador del lenguaje de animación, en el cual el algoritmo de Myers es lo mas importante.

- Desplegar y grabar directamente en tiempo real la secuencia en color que el lenguaje de animación calculó y grabó en el disco.

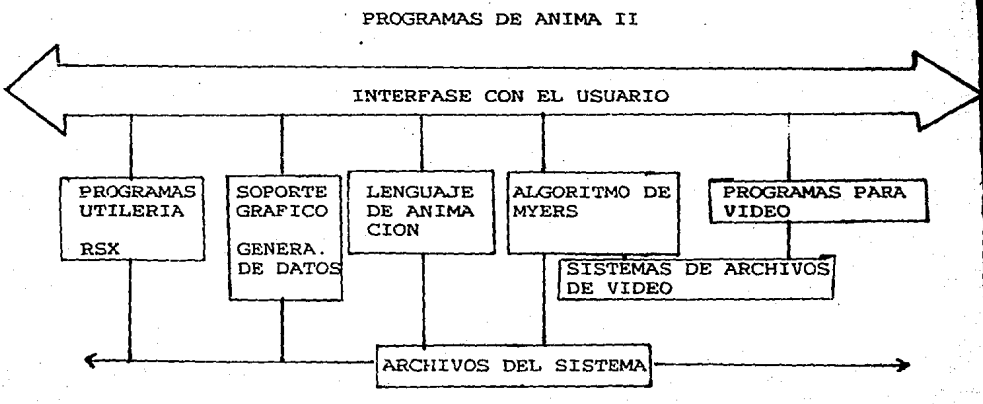


FIG. 3.5 Programas del sistema ANIMA II

3.4.1 GENERACION DE DATOS.

Los objetos se crean mediante un programa interactivo.

Los poliedros pueden unirse e interceptarse para formar figuras complejas. Para el usuario es transparente la estructura de datos de los objetos, la cual consiste de polígonos cerrados que forman superficies cóncavas o convexas cerradas. El usuario solo se preocupa por la posición de los objetos, por oprimir botones y por la mezcla de las superficies.

La rutina para la generación de datos utiliza 32K de memoria MOS y 20k de memoria principal para el manejo de dispositivos que

guardan la lista de despliegue y actualizan la pantalla. Aproximadamente se llevará de 5 a 15 minutos para la construcción de un bloque de letras, de 2 a 3 horas para un pato y una rana y mas de 5 para figuras mas complejas.

3.4.2 LIBRETO

Una vez que se han creado los objetos tridimensionales el usuario controla el proceso mediante un libreto el cual se escribe en un nguaje especial. El lenguaje de ANIMA II ofrece un medio de imitar los movimientos del mundo real dividiéndolos en movimientos mas simples pero controlados en el espacio y en el tiempo. Cada instrucción estará activa durante un cierto tiempo. Cuando éste haya trascurrido la instrucción se guardará para su uso posterior. El animador sólo especifica las posiciones iniciales y finales de los cuadros y el sistema aplica las transformaciones necesarias para producir los cuadros intermedios. Por ejemplo:

SET POSITION name , X, Y, Z AT frame1

CHANGE POSITION name TO X,Y,Z FROM frame1, TO frame 100

La habilidad de guardar las instrucciones del lenguaje permite al usuario animar múltiples objetos sin necesidad de involucrarse con el flujo de control del programa.

Cuando el animador está satisfecho con las acciones de los objetos puede controlar la escena completa. Otras facilidades del lenguaje de animacion son: color, brillo, iluminación, posición y rotación de las fuente de luz.

3.4.3 LENGUAJE DE ANIMACION.

Cuando se han descrito tanto los objetos como el movimiento el animador necesita invocar al lenguaje para calcular la secuencia final. Se compila el archivo de animación que contiene todos los objetos y parámetros de control necesarios para las rutinas de visualización de superficies. Si por ejemplo el libreto describe una secuencia de animación de 13 bloques de letras de varios lores que dura 20 segundos (600 cuadros), el archivo compilado se verá como una lista de 600 cambios dinámicos de estructuras de datos cada uno de los cuales define los parámetros de espacio y despliegue de una colección de superficies de colores para cada cuadro.

El programa se divide en cuatro rutinas: preprocesador, "scheduler", intérprete y compilador.

Preprocesador.

Esta rutina está relacionada con la construcción de las estructuras de datos pero también con la revisión de la sintaxis del libreto. Las estructuras de datos incluyen:

- Información de caras y vértices que describen la superficie poligonal tridimensional de cada objeto.

- Las diferentes formas posibles que puede tomar cada objto.

- Los grupos de apuntadores de cada objeto.

- Los subgrupos de apuntadores de cada objeto.

- Varias fuentes de luz flotantes.

- Diversas rutas tridimensionales compartidas por todos los objetos.

-Un color para el interior y otro para el exterior de cada cara del objeto.

El lenguaje de animación puede controlar hasta 128 objetos, grupos de objetos o posibles formas de objetos. La limitación que existe son los 32K de espacio direccionable en la máquina. El lenguaje de animacion utiliza 32K de memoria MOS y 32K de memoria principal, esto libera 20K de espacio para datos de objeto (4000 a 5000 lados) y en otra parte 16K para la definición de vértices y rutas (alrededor de 5300 puntos de 3 palabras cada uno). Si existe espacio en memoria el lenguaje puede controlar 128 objetos, 128 formas, 64 grupos, 32 rutas y hasta 500 instrucciones. La rutina de preprocesador revisa sintaxis, interpreta y ejecuta las instrucciones en el libreto.

"Scheduler".

Sus funciones son :

- Juzga acerca de cuándo el bloque de comandos se activa y se desactiva después de comparar la información de los cuadros clave en el bloque de comandos, con el contador de cuadros del sistema.

- Actualiza los parámetros de movimiento en el area de trabajo, si el comando está activo para el cuadro.

Intérprete.

Después de que el bloque de comandos pasa por el "scheduler", el intérprete busca cada comando activo, determina el tipo de

parámetro (rotación, posición, medida, color, forma, ruta, etc) y efectúa los movimientos y transformaciones necesarias a la estructura de datos.

Compilador.

Esta rutina compila el archivo de datos y lo convierte en código ejecutable, calcula el color de cada cara, hace las transformaciones de perspectiva y recorte y construye el archivo de animación que contiene la descripción completa de la escena de cada cuadro en el libreto.

El color de cada cara se obtiene de las relaciones entre la posición de las fuentes de luz y el plano de la cara. El sistema cuenta con tres fuentes de luz que pueden ser trasladadas y rotadas. La distancia de la fuente a la cara determina el brillo. De esta relación se obtiene un valor entre 1 y 224 que corresponde a un color en la paleta de colores.

Cuando se ha compilado el último cuadro se obtiene un archivo de datos que se convertirá en la secuencia final de colores mediante el algoritmo de superficies y la rutina de conversión del rastreador.
El lenguaje calcula tipicamente una secuencia de 300 cuadros (10 seg) en menos de 5 minutos.

3.4.5 DESPLIEGUE Y REGISTRO.

Normalmente se utiliza un mecanismo estandar de transmisión de televisión. La transmisión al video no se almacena en formato NTSC, sino en combinaciones de código de intensidad de color, que serán convertidas a formato NTSC en tiempo real.

ANIMA II fué diseñado e implementado en la Universidad de Ohio.

Algunos integrantes del grupo que lo realizó son: Charles C'suri, Allan Myers, Richards Parent, Timothy Van Hook, Diana Rainwater y Ronald Hackthorn.

3.5 ANIMENGINE

Animengine es un sistema diseñado especificamente para el area de ingenieria. Auxilia a los diseñadoras a encontrar y resolver los problemas relacionados con el diseño de procesos a través de una interfase hombre-máquina.

La animación en ingenieria tiene los siguientes requierimientos:

1. Despliegue exacto de objetos y sin ambiguedades.

Los objetos desplegados en ingenieria son partes mecánicas, módulos, unidades de producción y máquinas de ensamblaje.

Debe ser posible distinguir los diferentes objetos en la pantalla de una forma exacta. También es muy importante la apariencia sólida.

2. Producción a alta velocidad y automáticamente.

La animación en ingenieria es un camino para la comunicación entre diseñadores, ingenieros y trabajadores, por lo que es necesario producir la animación tan rápido como sea posible.
La animación en ingenieria es producida normalmente por diseñadores los cuales no son animadores profesionales, por esta razón un sistema de animación en ingenieria debe trabajar automáticamente.

3. Baja dependencia de la computadora y bajo costo.

El sistema debe estar disponible constantemente durante el proceso de diseño. Tipicamente varias estaciones comparten un mismo computador pero no deben depender demasiado del mismo, además las terminales deben ser baratas.

Basado en los tres requerimientos anteriores, se tomaron las siguientes decisiones para el diseño de ANIMENGINE.

1. Una pantalla caligráfica o vectorial trabaja más rápido que un rastreador, pero se seleccionó éste último por la necesidad de despliegue de áreas sólidas de color y patrones para hacer mas identificables los componentes de las figuras.

2. Para tener una velocidad alta de animación se necesita una pantalla de alta velocidad. Un sistema en tiempo real es el ideal, pero el equipo para tiempo real en un rastreador es muy caro. En su lugar se utiliza una pantalla interactiva combinada con una grabadora de video. El diseñador puede ver cada marco de animación uno a la vez, grabarlo y después desplegarlo.
Esta configuración del sistema produce varias escenas por minuto.

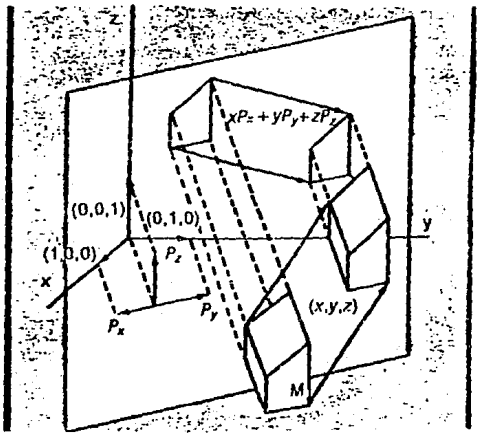
Se prefieren las videocintas a la pelicula porque se puede ver la animación inmediatamente ya que no se necesita calidad de película, ni resolución muy fina. Otra de las ventajas de utilizar señales de video es que la animación se puede guardar facilmente y transferirse a través de una red de sistemas digital y analógica.

3. Para lograr la baja dependencia del computador se necesita un dispositivo gráfico que sea inteligente. Si el dispositivo tiene un "buffer" segmentado permite trasladar, rotar y escalonar figuras hacia abajo y arriba con comandos simples.

3.5.1 DESPLIEGUE EN DOS DIMENSIONES

Para este sistema basta un dispositivo gráfico bidimensional, esto viene del hecho de que gran parte del movimiento que se va a desplegar es movimiento paralelo combinado con rotación simple y puede ser hecho en una pantalla bidimensional.

Si el movimiento de los objetos es paralelo y la proyección es paralela u oblicua, podemos tratar la traslación de un objeto en el espacio tridimensional, como su traslación en el plano bidimensional proyectado. El caso se muestra en la figura:



Plano proyectado

FIG. 3.6 Movimiento paralelo en el espacio del objeto y movimiento paralelo en el plano proyectado

La proyección del objeto M, el cual es trasladado por (x,y,z) en el espacio del objeto, es la misma que la proyección de M trasladado xPx+ YpY + zPz en el plano de proyección donde Px, Py y Pz son proyecciones de los vectores unitarios en el espacio del objeto.

Esto significa que para un movimiento paralelo, se necesita transmitir solamente la proyección de la forma del objeto a la estación de trabajo y la animación puede producirse localmente. En el caso de rotación de objetos, se tiene que transmitir una proyección diferente para cada cuadro.

3.5.2 EQUIPO

El equipo utilizado se muestra en la figura:

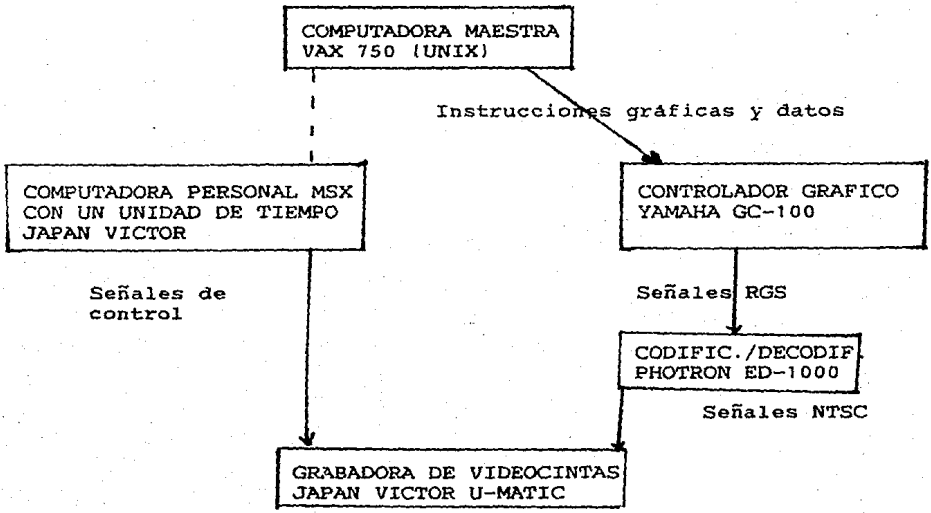


FIG. 3.7 Equipo físico que utiliza ANIMENGINE

Se utiliza una computadora VAX 750 con UNIX. El dispositivo gráfico es una Yamaha GC-100 con 725 x 480 "pixels". Puede emitir señales RGB cuyo tiempo coincida con las señales NTSC RS-170 y con un Photron encoder/decoder ED-100, se pueden obtener señales de video NTSC directamente de el GC-100.

Se usa una grabadora de videocinta Victor U-matic.
Esta grabadora se controla por medio de una computadora personal MSX con una unidad Japan Victor.

3.5.3 PROGRAMAS DEL SISTEMA

El sistema consta de tres unidades llamadas Controlador Global, Manejador de la pantalla y Manejador de la grabadora.

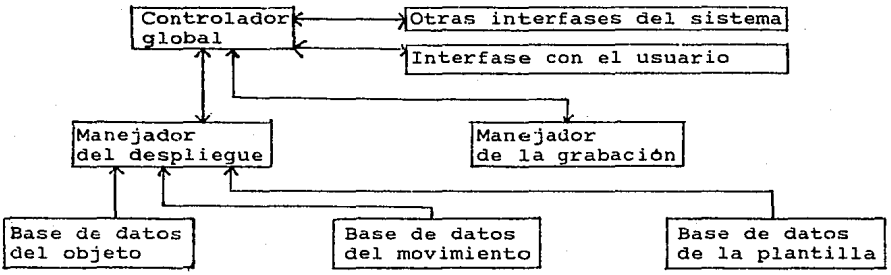


FIG. 3.8 Programas del sistema

El controlador global maneja el sistema completo así como sus interfases con otros sistemas y usuarios. El manejador de la pantalla controla el despliegue de gráficas, que es la interfase con la base de datos del objeto, del movimiento y de la pantalla.

La base de datos del objeto contiene los datos de la forma del objeto. La base de datos del movimiento contiene los datos que describen su posición y su movimiento. Estas bases de datos son compartidas por otros sistemas como son el sistema de ayuda de diseño y los sistemas de simulación dinámica.

3.5.4 PRINCIPIOS DE DESPLIEGUE.

Como ya se mencionó, el sistema debe mostrar una apariencia sólida para ayudar a los diseñadores a reconocer la forma, posición y dirección de los objetos desplegados.

Algunos de los métodos para hacer esto son el trazado de rayo, sombreado, oscurecimiento de los objetos distantes y textura.

No se utiliza trazado de rayo por la cantidad de cálculos requeridos. Ya que los objetos distantes también son importantes, no se pueden oscurecer por lo que se utiliza sombreado y textura para dar la apariencia sólida.

Sombreado.

Para el sombrado se utiliza un modelo de reflexión difusa con iluminación paralela por lo que no se necesita cambiar los colores de los objetos mientras estos se mueven en paralelo en el espacio del objeto.

Técnicas de despliegue cartográfico.

Los colores de cada objeto no necesitan ser realistas, pero si ser únicos. El color para cada cara varia dentro de un cierto límite dependiendo del efecto de sombreado. Esto permite la fácil identificación de cada objeto. Las caracteristicas de textura se usan también para el mismo propósito.

Exactitud y no ambiguedad son contradictorios, por ejemplo un hoyo pequeño dibujado exactamente puede ser dificil de ver, es mejor poner un simbolo en lugar del hoyo. El sistema usa marcas simbólicas para expresar las posiciones de cada uno de los pequeños hoyos o puntos invisibles. Mediante lineas adicionales se pueden representar direcciones del movimientos. A estas técnicas se les denomina técnicas de despliegue cartografico.

Anti-escalonamiento.

ANIMENGINE no contempla el antiescalonamiento, esto se debe a que las figuras generadas por señales NTSC son menos claras que las señales de un RGB y el antiescalonamiento es innecesario.

DIBUJADO DE LAS ORILLAS.

En animación para ingeniería es importante distinguir las orillas de los objetos desplegados. Estas son identificables por el corte de color, pero algunas veces no pueden distinguirse como en el caso de dos superficies pintadas con el mismo color.

Por ejemplo, dos caras que tienen el mismo vector normal son pintadas con el mismo color, consecuentemente la orilla que separa las dos caras no es identificable. Considerando casos similares los diseñadores prefieren que las orillas sean lineas distintas del resto del objeto como en la figura.

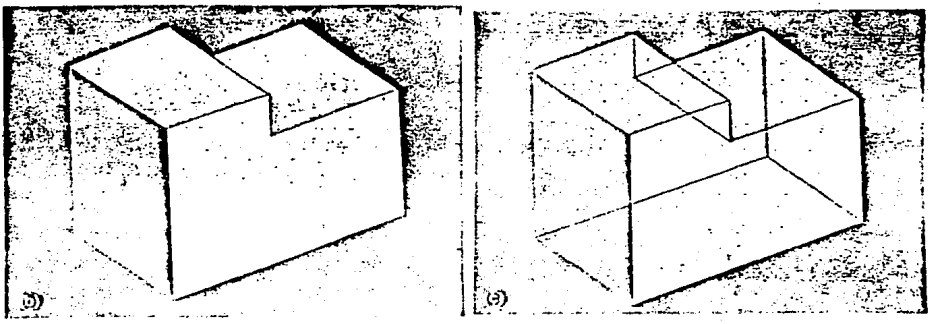


FIG. 3.9 Efectos sobre las orillas

Efectos de transparencia.

Los modelos geométricos de ANIMENGINE están basados en poliedros y todas las superficies son planas.

En general un dibujo de áreas sólidas es mejor que un dibujo de primitivo pero este último tiene la ventaja de que hace reconocibles las superfices ocultas. Desplegando las orillas de la superficies ocultas junto con las superficies visibles puede tenerse una figura transparente y reconocer la forma completa del objeto dado.

Visibilidad de las orillas

Las orillas de superficies visibles no son necesariamente orillas visibles. Por ejemplo en la siguiente figura, es claro que el número de rectángulos representa la curva de la superficie, aunque no se tienen que dibujar las orillas entre cada rectángulo.

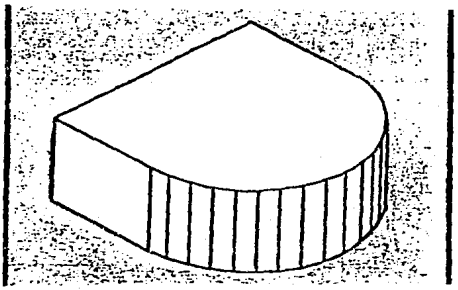


FIG. 3.10 Sólido con una superficie aproximada por curvas

Se decidió que la visibilidad de las orillas depende del ángulo entre las dos superficies que comparten la orilla.

El algoritmo que se usa está basado en el algoritmo Painter's por lo que pueden utilizarse las funciones mas inteligentes del dispositivo.

Se dibuja cada escena de atrás para adelante por lo que las líneas ocultas vienen a ser el elemento para decidir el orden en el que se dibuja cada cara. El punto mas importante de este algoritmo es determinar cuando se intersectan las proyecciones de dos caras en el plano de proyección en un periodo dado. El algoritmo contempla dos casos : el primero cuando una cara es estacionaria y en movimiento paralelo y el segundo cuando las caras son ralelas y están en movimiento uniforme.

El determinar cuando la proyección de dos caras se intersectan una a la otra en el plano de proyección en un tiempo dado es equivalente a detectar la interferencia entre dos prismas oblicuos en el espacio tridimensional, compuesto por dos ejes en el plano de proyección y un eje de tiempo.

En la figura dos prismas oblicuos representan el movimiento de dos caras en el tiempo.

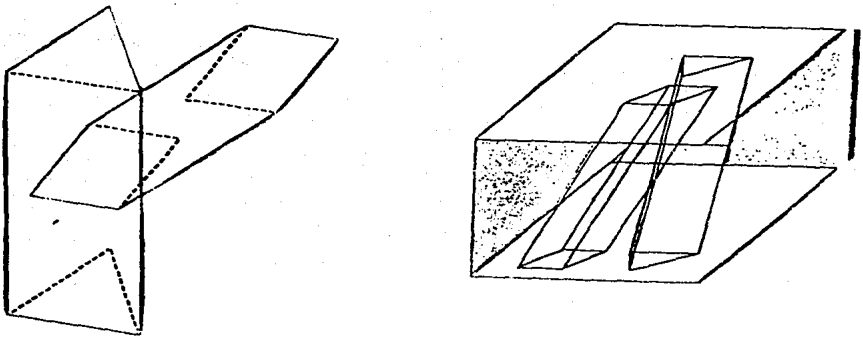


FIG. 3.11 Detección de la interferencia de dos prismas oblicuos

Si los dos prismas oblicuos se intersectan, las proyecciones de las dos caras se intersectan durante el tiempo. Si se hace el cálculo en el espacio tridimensional se puede decidir cuando se intersectan y cuando no, pero el cálculo se requiere en el plano proyectado.

Uno de los caminos alternativos es examinar el lugar geométrico de las proyecciones de las dos caras, si éstos no se intersectan las proyecciones no se intersectan en el tiempo. Lo contrario a esta prueba es necesario pero no suficiente para decidir cuándo la proyección de las caras se intersecta.

Los objetos en ingeniería son rígidos y sus caras no cambian. Si se toma un nuevo origen de coordenadas en un punto fijo de una de las proyecciones de los objetos, el movimiento de la proyección del otro objeto en las nuevas coordenadas será la diferencia de los movimientos originales. La proyección del primer objeto, intersecta el lugar geométrico de la proyección del segundo objeto en las nuevas coordenadas, si y sólo si las proyecciones originales se intersectan.

3.5.6 MODELO DE PLANTILLA

En ingeniería aparecen frecuentemente figuras como flechas, anillos, pernos, tuercas, y tornillos. Estos objetos tienen la mismo forma dentro de un tipo particular de parte y sólo difieren en medida.

Un modelo de plantilla consiste de un nombre de parte y una lista de números que especifican su medida. Cuando se dibuja el modelo de plantilla se utiliza la información de la base de datos de plantilla. Generalmente esta información en la base de datos, no incluye modelos sólidos o especificaciones para el llenado de la figura en el espacio tridimensional, solamente provee las reglas para el dibujado.

Estos modelos tienen otra ventaja, el manejo de la rotación de objetos. Por ejemplo al especificar una flecha como un modelo de plantilla que rota alrededor de su línea de centro, no es necesario redibujarla porque la rotación no cambia el modo en el que se dibuja.

3.5.6 CONCLUSIONES Y TRABAJO FUTURO

ANIMENGINE satisface los requerimientos para animación en ingeniería, despliegue exacto y sin ambiguedades, alta velocidad, producción automática, dependencia baja del computador y costo modesto. La presentación de áreas sombreadas necesarias para identificación fácil de los componentes de la figura es un requerimiento que entra en conflicto con el despliegue rápido. Se tiene que usar un método nuevo para superficies ocultas el cual permita producir despliegues con suficiente calidad y en forma rápida.

3.6 SISTEMA DE LA COMPAÑIA PACIFIC DATA IMAGE

Pacific Data Images es una empresa que produce animación comercial para transmitirse en televisión. Mas adelante se explicará paso a paso el proceso que se sigue , discutiendo las herramientas con que se cuenta, así como los programas necesarios para sus producciones.

3.6.1 PANORAMA DEL SISTEMA

El sistema PDI fue desarrollado especialmente para la producción de animación comercial. Produce animación en 3 dimensiones en un rastreador, cuenta con un lenguaje de animación, y con herramientas para diseño de animación en tiempo real. Su salida es hacia película de 16 mm, 35 mm o videocinta de 1 pulgada. En la siguiente figura se presenta el esquema general del sistema.

La animación se define mediante: un libreto escrito en el lenguaje de animación del sistema, un archivo de coordenadas universales de polígonos y un conjunto de datos del movimiento.
Estos datos son procesados por el programa Libreto, el cual interpreta el libreto de animación y crea varios tipos de archivos de polígonos.
El archivo de polígonos en coordenadas universales también puede desplegarse en forma de modelo primitivo y puede definir modelos que se utilicen en otros libretos.

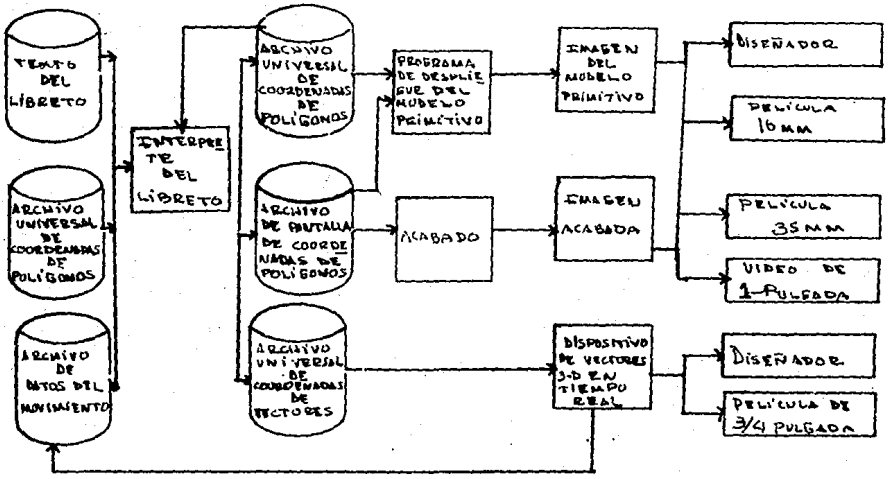


FIG. 3.12 Diagrama del sistema

Las imágenes se despliegan en monitores de color o en pantallas caligráficas. También pueden enviarse directamente a una o dos grabadoras de películas o a un codificador NTSC para grabación en videocinta.

"Libreto"

Es un lenguaje gráfico de propósito especial que permite realizar animación de alto nivel. El sistema libreto está escrito en lenguaje C y comparte muchas de sus características y facilidades. Está diseñado con una sintaxis sencilla, maneja valores de omisión para el ambiente gráfico en general, permite modelado, transformaciones y programación modular.
Las estructuras de datos y tipos de archivos que utiliza son compartidos con otras herramientas de diseño y esto permite la comunicación de todos los programas que se diseñan.

Se le llamó "libreto" porque se utiliza para procesos de producción. En las primeras etapas del trabajo, el modelo básico y el movimiento para la animación se describen en libreto, este guión orginal incluye tiempos elementales, iluminación, vistas e información del modelado que mas tarde será refinada y utilizada en la construcción del producto final.

En cada etapa de la producción, el libreto se actualiza y refleja los cambios en la producción para la incorporación de nuevos modelos y movimiento de datos para otras partes del sistema. Realmente, el libreto es mucho mas que un guión de película ya que describe de manera concisa, qué está pasando, cuándo y donde en toda la producción, definiendo también la animación.

El libreto fue diseñado para ser un lenguaje prototipo en la construcción de imágenes y pruebas de movimiento. Típicamente el libreto preeliminar se construye en la primera sesión con el cliente.

El libreto maneja directamente los tipos de datos y las operaciones de modelado para iluminación, visualización y transformaciones. En esta parte se concentra la atención en qué es lo que se debe hacer, más que en cómo se va a implementar.
Muchos de los valores de omisión no necesitan especificaciones, pudiendo describir objetos rápida y facilmente. Por ejemplo si se requiere un prisma de 5 lados el comando en libreto sería :

Prism 5.

Si no se especifican los atributos del prisma como el color, el radio, la altura y el tipo de superficie, se asumen los estándares o los valores previamente establecidos. Adicionalmente otros parámetros como la posición de la cámara, el foco del lente, luces, ventanas y puertos de visión, se establecen en el

libreto o se toman por omisión.

Como la secuencia de animación se redefine y se corrige en el smo libreto, la posibilidad de error y de repeticiones innecesarias se minimiza.

3.6.2 DISEÑO DEL MOVIMIENTO

Ya que las principales tareas de un animador y de un director en animación de 3-D, es la "coreografía" de los objetos y el control de la cámara, se ha desarrollado un sistema de diseño de movimiento.
El sistema consiste de:

1) Diversas herramientas para diseño de movimiento interactivo y no interactivo que pueden utilizarse en cada escena
2) Métodos para la visualización de cada secuencia de movimiento.
La siguiente figura ilustra el flujo de los datos durante el proceso de diseño del movimiento.

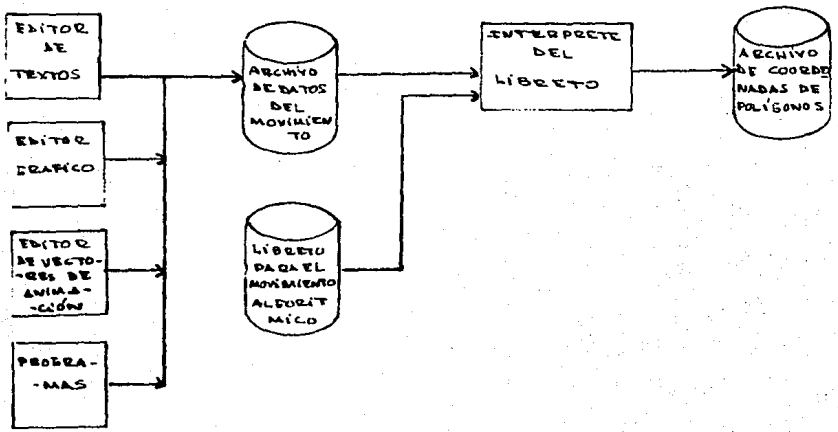


FIG. 3.13 Programas para el diseño del movimiento

Para el diseño interactivo del movimiento se utiliza una pantalla caligráfica o vectorial y se establece la posición de los cuadros para la cámara y los actores. El animador especifica interactivamente la visión de la cámara, el foco de la lente, la dirección de visualización, y la posición y orientación de cada

actor.
Entonces se colocan los cuadros en una secuencia de animación. En seguida la computadora genera el número de posiciones requeridas para los cuadros intermedios.

Pueden utilizarse métodos matemáticos para el control del espacio y del tiempo asi como para modificar la transición del movimiento entre cuadros.

Vista previa del Movimiento.

Se graba el modelo primitivo de una secuencia de animación. Se produce un despliegue de la coreografia actual, a través del uso de modelos simples. Pueden efectuarse pruebas de movimiento antes de tener a los actores en cada una de las escenas.

Finalmente el movimiento puede verse en un resolución baja de 256 x 234. Aqui la coreografia, iluminación y color pueden verse juntos antes del acabado final.

3.6.3 DISEÑO DE MODELOS

La siguiente figura ilustra las diferentes herramientas, para la creación del modelo final.

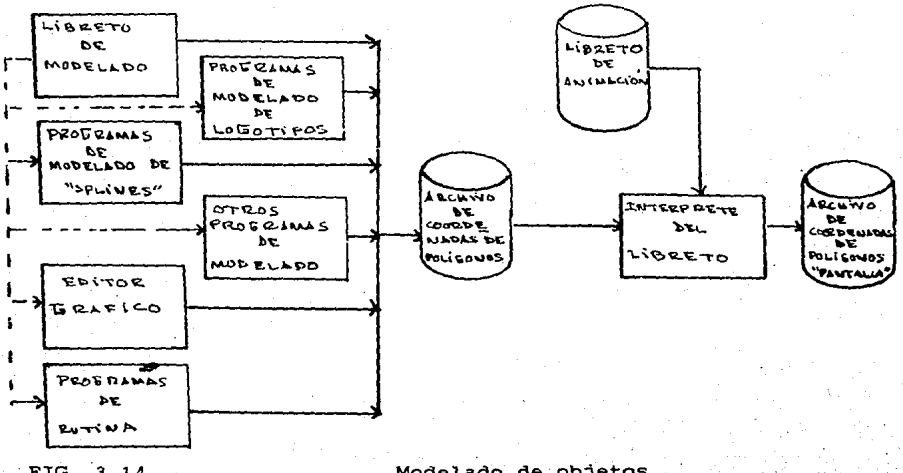


FIG. 3.14 Modelado de objetos

En el lenguaje libreto, los programas de modelado y el editor gráfico, producen un formato común para el archivo de coordenadas de polígonos. Este archivo, será procesado por otros programas de modelado.

Una vèz que el archivo de polígonos está completo puede accesarse mediante el libreto de animación para la conversión a coordenadas de pantalla e incorporarlo asi a la escena final. El lenguaje libreto se usará para crear modelos basados en primitivos geométricos, pueden crearse objetos simples con estos primitivos.

Existe un programa especial para la creación de logotipos tridimensionales en base a contornos bidimensionales. Los "splines" y contornos usados por este programa pueden ser introducidos de tres maneras:

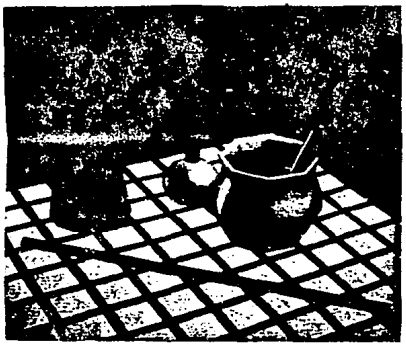
1) Usando un editor interactivo
2) En base a datos creados en un editor de texto
3) Por medio de otros programas.

Se utilizan también programas de modelado basados en el concepto de filtros. Un programa de filtro es aquel que acepta un conjunto de datos, efectua una serie de transformaciones sobre ellos, y emite los datos en su forma original. Las tareas complejas se realizan combinando los programas de filtro de tal forma que la salida de un programa es la entrada del siguiente, este tipo de combinaciones, proveen capacidades gráficas extensas con programas relativamente simple.

Los programas aceptan como entrada archivos de polígonos en coordenadas universales, los procesan (cambian su color, aplican transformaciones, etc..) y emiten un nuevo archivo de polignos.

Cuando no es suficiente un modelado geométrico, se tienen disponible un editor gráfico interactivo. El artista dibuja el modelo con una tableta de datos y un monitor interactivo. La computadora guarda el dibujo en un archivo y lo convierte en polígonos los cuales pueden utilizarse como modelos o como entrada para otros programas de modelado.

Darle "acabado" a un modelo gráfico significa desplegarlo en una pantalla de color como un objeto sólido eliminando las superficies ocultas y con un sombreado adecuado para tres dimensiones, cuando el modelo no ha sido "acabado" aun se le denomina modelo primitivo.



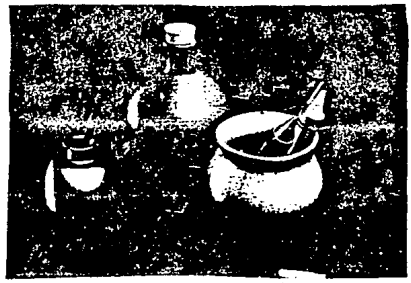


FIG. 3.15 Detalles del acabado de un objeto

Es difícil diseñar un objeto complejo por lo que es conveniente contar con un método rápido para visualizar los objetos en su etapa de diseño. En el sistema, cualquier modelo puede previsualizarse como un modelo primitivo de color. Los modelos primitivos tienen la ventaja de desplegarse rapidamente y mostrar la estructura completa de la escena. Los dibujos de modelos primitivos pueden también graficarse en papel para la planificación detallada y la documentación del sistema.

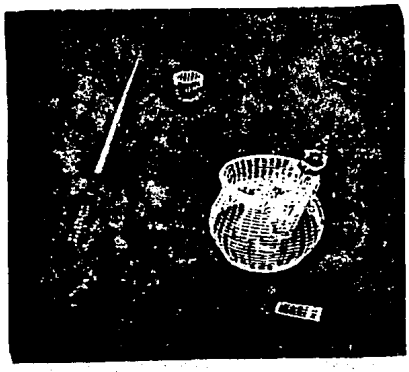


FIG. 3.16 Modelos primitivos de un objeto

A cada modelo se le da un nombre y se guarda en una biblioteca de modelos. Si una escena se forma de varios modelos , el libreto puede recuperarlos de la biblioteca.

Una vez que el modelo ha sido "acabado", puede colocarse en cualquier lugar de la pantalla solo, o en combinación con otros modelos. Adicionalmente la cámara puede posicionarse para ver cualquier localidad de la pantalla, también se permiten varias fuentes de luz, en cualquier posición y de cualquier color. De hecho en esta etapa del diseño la mayor parte del tiempo se invierte en el ajuste de las luces y tonalidades finales del modelo.

3.6.4 PRUEBAS DE CUADRO

Los modelos y los movimientos pueden usarse cualquier número de veces, en un variedad de formas. Esto es particularmente ventajoso para la preparación de pruebas de una escena, un libreto puede refinarse paulatinamente y observarse hasta que se esté seguro de que se tiene lo que se quiere.

Un modelo primitivo puede cambiarse en diferentes órdenes de magnitud mas rápidamente que una imágen totalmente acabada, mostrando el movimiento exactamente como aparecerá al final del trabajo, ya que se usará el mismo libreto y los mismos modelos en la pieza final. También se efectuan pruebas para el acabado en varias etapas de la animación, por lo que el diseñador puede ver y modificar los modelos, el colorido, la luz y en general toda la escena.

Normalmente la última parte de la fase de pruebas, es el despliegue de la secuencia "acabada" en baja resolución. En esta imágen se han eliminado las superficies ocultas, el escalonamiento y se ha dado el sombreado adecuado para 3-D.

Las pruebas de "acabado" final necesitan desplegarse a la velocidad final de la película. Viendo cada una de las escenas finales el diseñador debe estar seguro de que todos los datos están correctos, que se actualizan y funcionan bien.
Para estar seguro de esto, todos los datos del movimiento deben reunirse y actualizarse antes de las pruebas de cuadro.

En la siguiente figura se muestra como se coordinan las piezas de información para formar la imágen final.

Lo importante de todas estas pruebas es la facilidad con la que se producen, en algunos casos están disponibles es unos cuantos minutos y pueden repetirse. Los diseñadores de animación tienen

el control preciso sobre lo que debe ser cambiado y lo que permance igual.

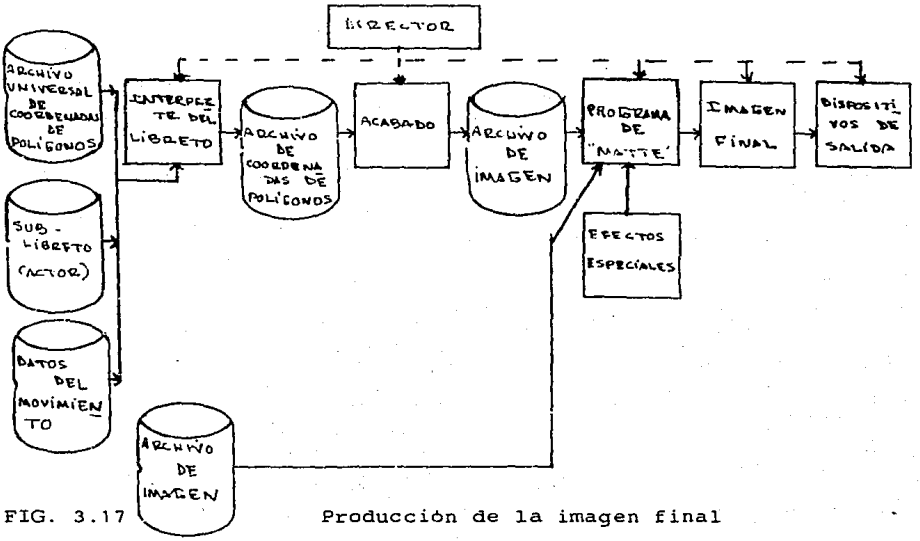


FIG. 3.17 Producción de la imagen final

3.6.5 EL CUADRO O ESCENA FINAL

Después de que se han terminado la pruebas se crea la secuencia final. El producto final se graba en 16 mm o 35mm a través del uso de una computadora controlada por una grabadora de peliculas.

Generalmente se prefiere producir secuencias de animación finales directamente en videocinta de una pulgada. Para hacer esto se usa un lenguaje para controlar el "acabado" de la secuencia final, para el libreto de animación y para el control automático de la grabadora de peliculas.
Si se utilizan varios libretos o si la escena necesita imágenes de fuentes externas, el lenguage también lo controla.

El trabajo final se le entrega al cliente junto con los libretos y los modelos. La selección de imagenes intermedias para la animación se archiva en una base de datos.

El estudio se abrió comercialmente desde Mayo de 1983 y desde entonces se ha hecho producción de comerciales. Cerca de la mitad

de los trabajos se han realizado sin el uso de programas costosos.

3.7 MIRA. UN SISTEMA DE ANIMACION EN TRES DIMENSIONES

MIRA es uno de los sistemas de animación mas completos. El primer articulo sobre el sistema aparece en 1983 y se cuenta con un total de cuatro articulos que describen las ampliaciones que se le han hecho hasta el presente año.

El sistema se desarrolló en la Universidad de Montreal Canadá.
MIRA se desarrolló como una extensión del lenguaje Pascal ya que éste no tiene la capacidad de definición de tipos de datos abstractos. En esta extensión se define un nuevo tipo de datos llamado figura. La extensión se llama MIRA-2D.
Posteriormente se adicionaron tipos gráficos en 3-D creando Mira-3D. Este lenguaje permite al usuario la construcción de figuras tridimensionales.

MIRA-3D incluye :

- Vectores artimèticos tridimensionales
- Instrucciones gráficas
- Transformaciones de imágenes
- Transformaciones visuales (perspectiva y proyecciones paralelas)

Lo mas importante en todo esto es el tipo figura. Para definir un tipo figura se deben definir :

1. Las caracteristicas de la figura y sus parámetros
2. El algoritmo que permita al usuario construir la figura con ayuda de los parámetros

```
type PYRAMID = figure (A,B,C,D:VECTOR);
               begin
                  moveabs A;lineabs B,C,A,D,C;
                  moveabs B; lineabs D
               end;
```

La figura puede tener atributos como estilo de lineas, intensidad, ancho de linea y color.

Se cuenta con varias figuras estandar que se usan comunmente en tres dimensiones, como son :

1. El tipo line (linea)
2. El tipo figura plana : triangle,square, circle y ellipse (triángulo, cuadrado, circulo y elipse)
3. El tipo plane (plano) definido por tres vectores
4. El tipo box (caja), definido por cuatro vectores
5. El tipo sphere (esfera) definido por un centro C y un radio

R (aproximada por polígonos)

6. Los 5 tipos de poliedros regulares : thetrahedron, cube, octahedron, dodecahedron e icosahedron (tetraedro, cubo, octaedro, dodecaedro e icosaedro)
7. Los tipos superficie : cylinder, cone, parabolic cone, parametric surface, revolution surface, bezier surface, coons surface y b-spline surface (cilindro, cono, cono parabólico, superficie paramètrica, superficie de revolución, superficie de bezier, superficie de coons y superficie b-spline).

El MIRA 3-D se implementó desarrollando un preprocesador que cuenta con 600 lineas de programa fuente en Pascal desarrollado en una CDC CYBER 173 y una DEC VAX-780 . La salida es un Programa estandar en Pascal. La biblioteca consta de 10,000 líneas de programa Pascal y es independiente de la máquina. La parte dependiente puede reescribirse y adaptarse facilmente para una HP2648A, Tektronix 4027, DEC GIGI, Norpak, terminales AED, graficadores Hewlett Packard y varias impresoras. Sólo los procedimientos visuales se toman del GSPC CORE SYSTEM (paquete de graficación estandard).

En la página siguiente se muestra una jirafa construida utilizando MIRA-3D.

La pelicula "Sueño de vuelo" esta totalmente producida en una computadora. Es la historia de una criatura que vive en otro planeta y sueña que viaja a través del espacio como un pájaro y llega a la tierra.

La pelicula fue desarrollada utilizando el MIRA 3D para crear objetos y movimiento.

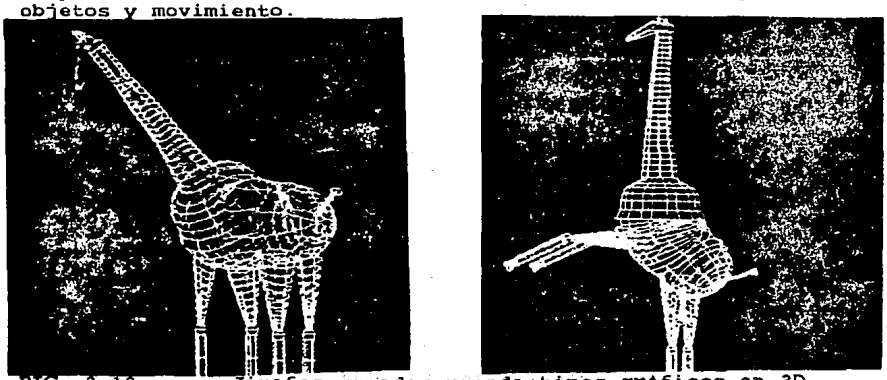


FIG. 3.18 Jirafas creadas usando tipos gráficos en 3D

Mediante un escena se explicará como se construyeron los tipos dinámicos. Un vagabundo está sentado en un bosque en la noche y tira unas piedras a un pequeño estanque, observa un pájaro e imagina que vuela como él. Esta escena involucra objetos dinámicos como el vagabundo, el pájaro y las ondas en el agua. También tiene objetos estáticos como la piedras, los árboles, el estanque, el horizonte y la esfera celeste con sus estrellas.

El pájaro se define como sigue :

```
type BIRD = figure (FRAME: INTEGER; H:HALF-BODY; W:WING;
C,D:VECTOR);
```

donde H es la parte derecha de la mitad del cuerpo, W la ala recha, C el centro de rotación del ala derecha, D la dirección de rotación de esa ala. Como el ala drecha siempre está en su máxima posición vertical, es necesario determinar el ángulo de rotación, este ángulo depende del cuadro. Para determinarlo se utiliza la ley de Catmull de aceleración/desaceleración.

Aqùi se muestra parte del código que se ejecuta cuando se crea la variable bird .

```
type BIRD= figure (FRAME:INTEGER; H:HALFBODY; W:WING;C,D:VECTOR);
var
     RELATIVE : 0..CYCLE;
     FRACTION, BETA: REAL;
     W2 : WING
     RIGHTPART, LEFTPART: FIG;
begin
     RELATIVE:=FRAME mod CYCLE;
     if RELATIVE > (CYCLE div 2) then
          RELATIVE:=CYCLE-RELATIVE;
          FRACTION:=(RELATIVE*2)/CYCLE;
          BETA:=LAW(ACCEDECE, ANGLEMAX, FRACTION);
          ROTATION(W,C,BETA,D,W2);
          UNION(H,W2,RIGHTPART);
          delete H,W2;
          SYMYZ(RIGHTPART,LEFTPART);
          include RIGHTPART,LEFTPART
end
```

En este listado CYCLE es el número de cuadros requeridos para hacer un movimiento del ala hacia arriba y hacia abajo, FRAME es el número del cuadro, BETA es el ángulo de rotación del ala, ANGLEMAX es el ángulo máximo y FRAC es la fracción de fase. Entonces el pájaro se crea y se dibuja de acuerdo a la siguiente sucesión :

```
procedure DRAWBIRD (FRAME:INTEGER);
var FIRSTBIRD: BIRD;
begin
     create FIRSTBIRD (FRAME, RIGHTBODY, RIGHTWING,C,D);
     TRANSLATION (FIRSTBIRD, <<0,0,FRAME*BIRDSTEP>>,FIRSTBIRD);
     draw FIRSTBIRD;
     delete FIRSTBIRD;
end;
```

En este procedimiento la variable FIRSTBIRD se crea dinámicamente en memoria, y se le aplica una traslación. El vector de traslación <<0,0, FRAME*BIRDSTEP>> tiene una componente z, dependiendo del número de cuadro. La instrucción DRAW significa que se despliega el contenido de la variable y finalmente se borra de la memoria.

3.7.1 TIPOS ACTOR Y CAMARA

Los tipos abstractos (como las figuras), son muy útiles, pero no tienen su propia animación, por lo que se han diseñado tipos de datos básicos animados. Estos tipos son: tipo actor y tipo cámara.

El siguiente ejemplo introduce este concepto.

```
type STRANGESQUARE = actor (A,B,C,D:VECTOR);
     time T1...T2;
     const V=.........(*velocidad*)
     type
          TVEC=animated VECTOR (P1,P2:VECTOR);
               val P1...P2;
               time T1...T2;
               law P1+ V * CLOCK
          end;
     var VERTEX: TVEC;
     begin
          init VERTEX (C,(A+C)/2);
          moveabs A;
          lineabs B, VERTEX, D,A
end;
```

Se define una cuadrado que se anima entre los tiempos T1 y T2. Un vértice se mueve en la dirección del centro con una velocidad constante V, este vértice está definido como un vector animado.

Un tipo básico de animación es un tipo básico definido de tal forma que cada variable de este tipo (llamada variable básica de animación) sea animada. Pueden animarse tres tipos básicos: enteros, reales, y vectores. Un tipo animado se define dando el valor inicial y final del número o del vector, los tiempos inicial y final, y la función o ley que describe como varían los valores en el tiempo.

Un tipo actor es un tipo de datos gráfico abstracto que se construye usando figuras que pueden manipularse.

Un actor puede referenciar a otro actor.

El bloque "actor" puede contener cualquier declaración excepto tipo actor y tipo cámara y puede contener cualquier instrucción de MIRA-3D.

El tipo cámara es un tipo abstracto animado. Su sintaxis es exactamente la misma que la sintaxis para el tipo actor en donde la palabra actor se reemplaza por la palabra cámara.
Los límites de tiempo tienen el mismo significado que para un actor.
Dentro de un tipo cámara pueden definirse tipos básicos de animación y variables, pero no pueden usarse tipos actor u otros tipos cámara.

Los tipos básicos de animación , tipo actor y tipo cámara se han introducido en Cinemira, el cual es un lenguaje de alto nivel cuya base gráfica es el MIRA-3D. Un libreto en Cinemira es un subprograma dedicado a la animación y consiste en una secuencia de escenas. Cada escena tiene un nombre y está construida mediante una secuencia de instrucciones que sirven para inicializar actores, cámaras y decorados.

El decorado es una colección de objetos gráficos que no se mueven durante la escena. En Cinemira el decorado se define mediante la instrucción:

```
decor <figure list>
```

donde la <figure list> es una lista de variable "figura". La ventaja de la instrucción decor es que reune a todos los objetos que no tienen animación por ejemplo :

```
create HOUSE (...);
create SKY;
create SUN(...);
decor HOUSE, SKY, SUN;
```

Para 1985 los autores de MIRA presentaron extensiones al sistema. El nuevo sistema está aun mas orientado hacia el artista o animador.

Se diseñó el sistema MIRANIM el cual comprende:

-Modelado de objetos 3-D
-Construcción de cuerpos
-Un sistema orientado al animador llamado ANIMEDIT
-Un sublenguaje para animación CINEMIRA-2

La siguiente figura muestra la organización del MIRANIM.

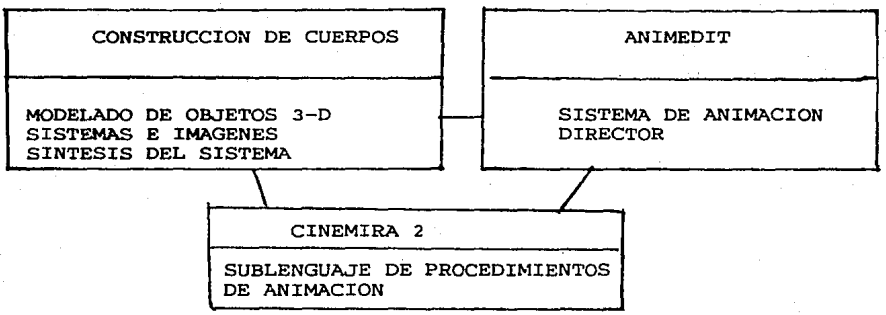


FIG. 3.19 Organización del sistema MIRANIM

3.7.2 MODELADO DE OBJETOS 3-D

Este proceso consiste en el modelado de objetos 3-D y en la síntesis de imágenes que permitan la construcción y combinación de objetos complejos sin necesidad de programar. Las curvas se generan usando entrada gráfica directa o invocando subsistemas que analizan las ecuaciones paramétricas. Estas curvas pueden usarse para la construcción de las siguientes superficies :

- Cilindros: definidos por el desplazamiento de un segmento de línea a lo largo de una curva.

- Conos: definidos por el desplazamiento de un segmento de línea a lo largo de una curva que pasa a través de un punto fijo.

- Conos parabólicos: definidos por el desplazamiento de una curva parabólica a lo largo de una curva pasando a través de un punto fijo

- Superficies de revolución: definidas por la rotación de una curva alrededor de una linea.

En seguida se muestran algunos ejemplos de imágenes producidas de esta manera.

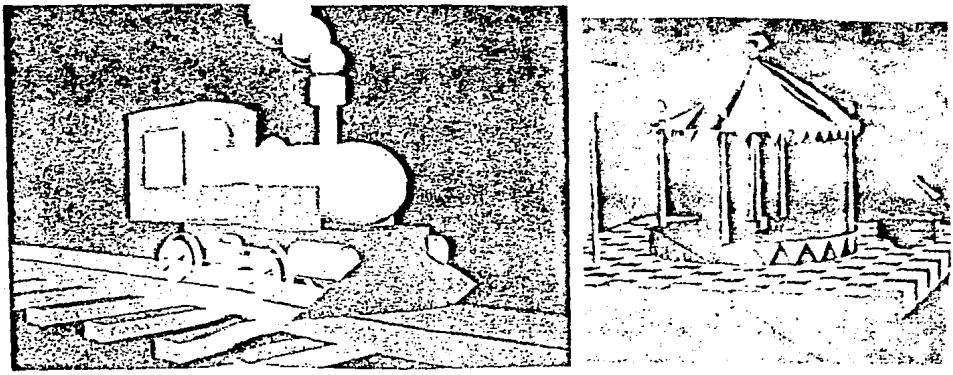


FIG. 3.20 Modelado de objetos 3-D

Las superficies paramètricas y las sub-superficies o parches (pedazos de superficies) pueden generarse en 4 formas :

- De 4 curvas utilizando el método Coons
- De puntos de control usando el método Bezier
- De puntos de control utilizando el método Barsky Beta-splines
- De ecuaciones paramètricas introducidas via un subsistema de análisis de ecuaciones.

También se cuenta con la posibilidad de crear texto tridimensional lo cual es una forma fácil para crear logotipos.

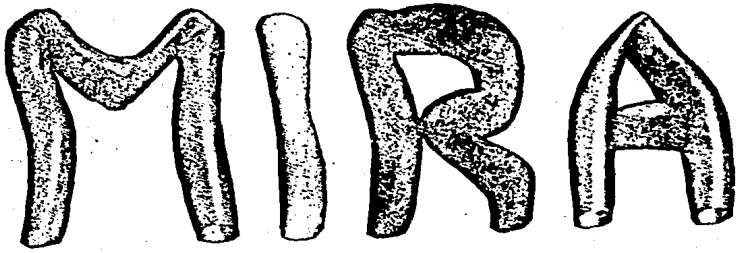


FIG. 3.21 Logotipo realizado en MIRA

Todos los objetos cuentan con un nombre, y se tienen comandos para el movimiento relativo de unos objetos respecto a otros , por ejemplo :

PUT objeto1 relación objeto2

donde la relación puede ser LEFT, RIGHT, ABOVE, UNDER, BEHIND o BEFORE.

Para colocar un florero en una mesa se usaría :

PUT VASE ABOVE CHAIR ALIGN VASE CHAIR XZ

Existen otros comandos para la concatenación de objetos, la definición de cámaras virtuales y el control de iluminación y sombreado. Se tienen 3 modelos de sombreado disponibles : Sombreado constante, Sombreado Gounraud y Sombreado Phong.

Pueden definirse varias fuentes de luz con componentes difusas y especulares.

En seguida se muestra una tabla con los comandos para construcción de objetos.

Comandos de construcción de cuerpos

ALIGN	BETA	COLOR
CONE	COONS	COPY
CREATE	DELETE	END
ERASE	EXECUTE	HELP
LIST	MOVE	PARACONIC
PARACAMERA	PERCAMERA	POSITION
PUT	READFILE	READGRAPH
REDRAW	REFLECTANCE	REVOLUTION
ROTATION	SAVE	SCALE
SHADE	SHOW	SOURCE
TEXT	TRANSLATION	TRANSPARENCY
UNION		

3.7.3 ANIMEDIT

Animedit es un sistema diseñado para artistas que no saben programar.
El animador puede crear actores con movimientos, cámaras virtuales y decorados interactivamente. Adicionalmente es posible definir fuentes de luz múltiples que se moverán alrededor.

El sistema consta de 8 modos de operación que son: MAIN, VARIABLE, DECOR, ACTOR, CAMARA, LIGHT y ANIMATION. Cada modo tiene sus propios comandos como se muestra en la figura de la página siguiente.

Creación de objetos

Se asume que el animador ha creado previamente sus objetos utilizando el sistema Constructor de objetos. El modo OBJECT en Animedit ofrece las posibilidades de un editor limitado de gráficas 3-D. Los objetos pueden rotarse, escalonarse, trasladarse y colorearse.

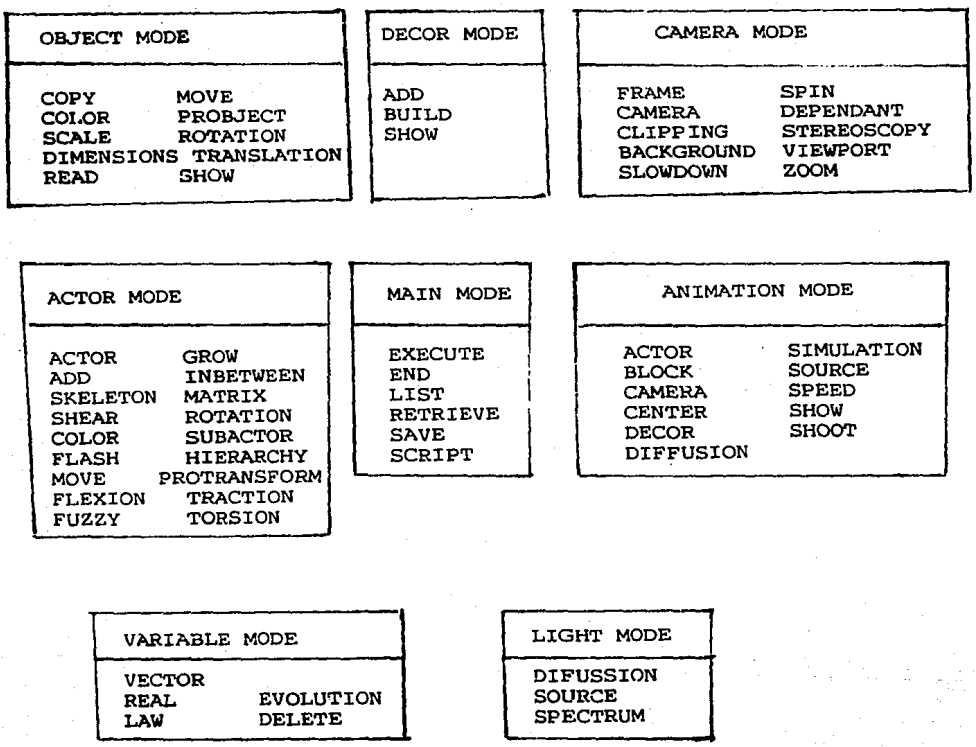


FIG. 3.22 Modos de operación de ANIMEDIT

Creación de decoración

Todos los objetos que no se modificarán en un buen tiempo son parte del decorado. El modo DECOR permite al animador construir objetos de decorado y deplegarlos para juzgar su apariencia. Por ejemplo :

DECOR MYDECOR, PAGODA, TREE1, TREE2

Dibuja lo que se muestra en la figura.

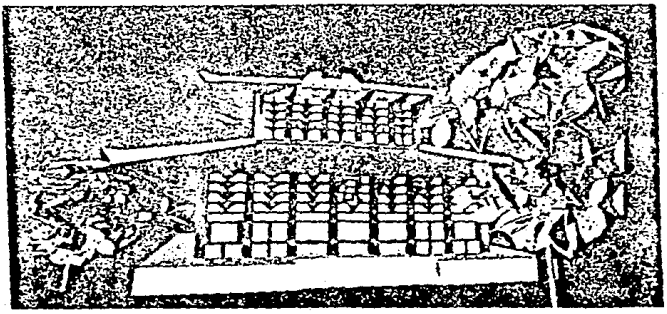


FIG. 3.23 Pagoda

Variables y definición de leyes

En el modo variable el animador puede crear constantes y variables animadas, estas variables son definidas por leyes de evolución que describen como cambian los valores a través del tiempo.
Entre las leyes disponibles están las leyes de Catmull y las principales leyes del movimiento físico.

Estas variables juegan un papel importantes ya que son ellas las que dirigen el movimiento, no sólo de los actores sino de las camaras y las luces.

Definición de actores

El animador define a los actores (objetos animados) y les asocia una lista de transformaciones en el modo ACTOR. Hasta ahora se tienen disponibles 16 tipos de transformaciones, que incluyen rotación, traslación, cambio de medida, torsión, recorte, tracción, flexión, transformaciones estocásticas y cambio de color.

Los parámetros de una transformación deben ser variables de animación, por ejemplo , en una rotación el ángulo puede ser un número real animado y la dirección del eje, un vector animado.

No se limita el número de transformaciones asociadas con un actor.

Definiciones de cámaras

En Animedit, el animador puede definir una o varias cámaras en el modo CAMARA. Cada cámara, tiene un punto de visión y varios puntos de interés. También pueden especificarse recorte, giro, punto de visión y acercamiento para la cámara. El punto de visión o el punto de interés de la cámara pueden seguir el movimiento de un actor, utilizando varias cámaras al mismo tiempo se logran efectos interesantes.

Control de luz

Pueden producirse imágenes realistas si se remueven las superficies ocultas y se adicionan efectos como sombreado, tonalidad, transparencia y textura.
Esto se controla en el modo LIGHT.

Animación

En el modo ANIMATION se decide el tiempo de aparición y duración de los actores, cámaras, luces y decorados.

Libretos

El animador puede en cualquier momento, traer actores, cámaras, decorados o luces. También puede desplegar o imprimir información detallada acerca de la escena. Finalmente se tiene la posibilidad de generar en cualquier momento un libreto óptimo usando comandos SCRIPT, esto significa que cualquier escena puede reconstruirse usando solamente el comando EXECUTE seguido del nombre del libreto.

3.7.4 CINEMIRA-2

El sublenguaje CINEMIRA-2 está limitado a la programación de entidades utilizadas por ANIMEDIT.
Por ejemplo un animador que quiere introducir una transformación EXPLOSION en una escena en la cual el actor CAR corre, se estrella y se destruye en una explosión, puede controlar la EXPLOSION por medio de parámeteros como la velocidad del carro V y la distancia al obstáculo D.

Los parámetros pueden tener uno de los siguientes tipos :

-VECTOR
-REAL
-OBJ para objetos gráficos
-ACT para actores
-CAM para cámaras.

Procedimientos objeto

Los tipos objeto son tipos gráficos de alto nivel que definen la composición de un objeto 3-D. En Cinemira-2 un objeto se modela como una combinación de caras, vértices y orillas, por ejemplo un cubo puede declararse de la siguiente forma :

figure of 8 vertices, 6 faces, 12 edges

Las caracteristicas de la figura se definen en una lista de parámetros formales que especifican el objeto para el usuario. Por ejemplo, para crear una SPHERE el usuario debe dar el centro y el radio

type SPHERE = object (CENTER:VECTOR; RADIUS:REAL)
En el cuadro se presenta la definición del tipo BOX que es un paralelepipedo caracterizado por sus vértices.

```
type BOX=figure (A,B,C,D,COLO:VECTOR);
     var CORI, BORI, DORI: VECTOR
     spec
          name BOX standing CONSTANT;
          figure of 8 vertices, 6 faces;
          begin
               CORI:=C-A;
               BORI:=B-A;
               DORI:=D-A;
               vertices=A,C,B+CORI,B,D,C+DORI,D+BORI+CORI,
               D+BORI;
               createface 1 to 6 with 4 edges;
               face 1:= 1,2,3,4;
               face 2:= 2,6,7,3;
               face 3:= 3,7,8,4;
               face 4:= 5,1,4,8;
               face 5:= 1,5,6,2;
               face 6:= 6,5,8,7;
               COLOR (COLO)
          end;
```

Procedimientos ley

Las leyes son funciones del tiempo (CLOCK). Estas leyes se usan mediante el comando LAW del editor.

En el siguiente cuadro se muestra la definición de una ley aplicable al movimiento de una cámara que realiza un acercamiento lineal.

```
law DEFLAW: VECTOR;

var
     FRACT, ALPHA, BETA, YVAREYE, RVAR:REAL
begin
     FRACT:= (CLOCK-DTAPP)/DIST;
     if CLOCK <= DTAPP then
          DEFLAW:= <0,YEYE,LAWC(4,DAPP-DTRANS,CLOCK/DTAPP)-DTAPP>
     else
          if FRACT <= TTRANS then
               begin
                    BETA:= 4*PI*FRACT;
                    YEYE:= 2*YMAX*(1+COS(BETA/2-PI))/2
                    DEFLAW:=<RI*(1-COS(BETA)),YEYE,-DTRANS+RI*
                         SIN(BETA)>
               end
          else
               begin
                    ALPHA:=2*PI*FRACT;
                    YVAREYE:=2*YMAX*(1+COS(ALPHA-PI))/2;
                    FRACT:=(FRACT-TTRANS)/(1-TTRANS);
                    RVAR:=R1-(R1-R2)*FRACT;
                    DEFLAW:=<RVAR*SIN(ALPHA),YVAREYE,-RVAR*
                         COS(ALPHA)>;
               end
end;
```

3.7.5 MIRANIM

El sistema Miranim se implementó en una computadora Vax 11/780 para varias terminales, las imágenes son producidas en una terminal AED 767.

En octubre de 1985 los autores publican un nuevo articulo en el cual tratan el problema de la animación como un problema de "evolución del movimiento" el cual puede describirse mediante "leyes de evolución".

Para realizar animación es importante el diseño de "leyes de evolución". Los animadores deben aplicar estas leyes a estados variables que dirijan la animación.
Estas leyes no deben ser sólo leyes analiticas, sino que deben basarse en ecuaciones diferenciales de un estado previo. Algunas veces estas leyes se modificarán durante el proceso de animación.

Estas leyes se han introducido en el MIRANIM como se presenta en el cuadro siguiente:

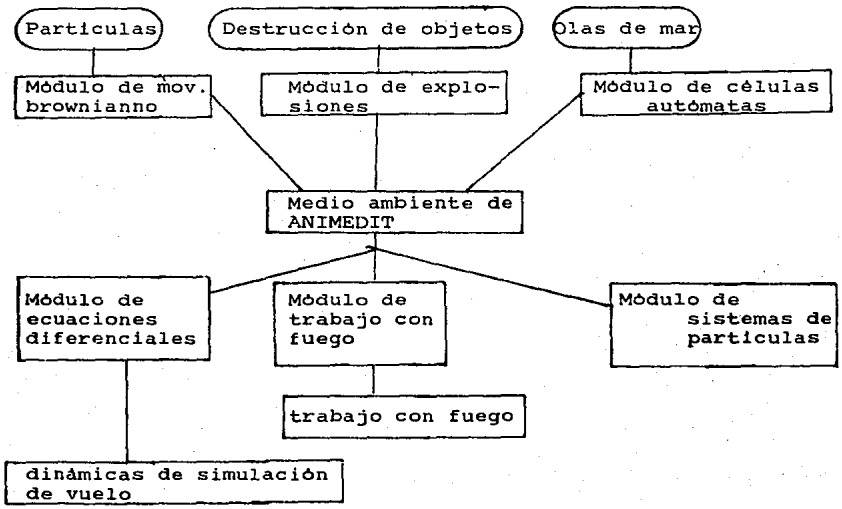


FIG. 3.24 Módulos especiales de ANIMEDIT.

El problema en el diseño de estas leyes, es formularlas analiticamente .

La estrategia en CINEMIRA-2 se basa en lo siguiente: si las leyes pueden expresarse analiticamente, se programa una ley simple, si la ley puede expresarse solamente como una función del estado previo, se guardan los valores como variables globales y la ley de evolución regresa un valor calculado en base a las variables globales.

Módulo de sistemas de particulas

Un sistema de particulas es una colección de particulas que juntas representan un objeto "fuzzy".

Un objeto dinámico "fuzzy" es un objeto bién definido y con superficies brillantes, su forma es irregular y cambia con el tiempo.

En un periodo de tiempo las particulas en el sistema nacen , se mueven , cambian y mueren. Reeves describe como calcular cada cuadro en una secuencia de movimiento mediante los siguientes pasos:

1. Generar una nueva particula por medio del control de procesos estocásticos asignándole atributos individuales

2. Desaparecer a las particulas cuyo tiempo se acabó

3. Mover y transformar las particulas restantes de acuerdo a sus atributos dinámicos

4. Dar acabado a las particulas vivientes.

Los sistemas de particulas pueden implementarse en MIRANIM. Por ejemplo, puede obtenerse la posición del centro de un sistema de particulas y considerarlo como un punto de interés para la cámara.

En modo OBJECT el sistema se inicializa como sigue:

```
PROCEDURAL <nombre del sistema> PARTICLES <número del sistema>
                                          <forma de la particula>
                                          <forma del sistema>
                                          <centro inicial del sistema>
                                          <velocidad del sistema>
                                          <aceleración del sistema>
                                          <ángulo de eyección>
```

Para cada particula del sistema se activa un bloque de animación llamado ATTRIBUTES que controla los atributos iniciales de cada particula nueva. El bloque se activa como sigue:

```
BLOCK ATTRIBUTES <número del sistema><tiempo inicial><duración>
      SIZE-RATE, VELOCITY-RATE,COLOR-RATE
      SIZE-INI, SPEED-INI, LIFETIME, COLOR-AVE, COLOR-STD
```

Las primeras tres variables son las tazas de cambio de la medida, el color y la velocidad. SIZE-INI, VELOCITY-INI y LIFETIME, son vectores variables que definen la velocidad y la desviación estandar de la medida inicial, la velocidad inicial y el tiempo de vida. COLOR-AVE y COLOR-STD son vectores variables que definen la distribución del color de la particula.

El movimiento de la particula se puede controlar en diferentes formas :

1. Utilizando leyes predefinidas (de movimiento lineal, circular, y armónico)

2. Utilizando movimiento Browniano

3. Utilizando leyes basadas en ecuaciones diferenciales simultáneas

Módulo de movimiento browniano simple

El movimiento browniano puede describirse al ver un pequeña particula inmersa en un liquido, la cual describe un movimiento en zigzag.
En este modelo se asume que la particula se mueve a lo largo de tres lineas unidimensionales (a lo largo, de los ejes x, y y z). Para obtener una ley que calcule la posición de la particula en cualquier momento, se tiene que usar una variable de estado global para cada particula. Estas variables son modificadas por un bloque de animación y la ley proporciona solamente la posición actual consultando las variables de estado. Ejemplo:

```
program BRMOTION;
var
     STATE:array 1..MAXPART of
          record
               POSITION:VECTOR:.......
          end;

law BROWNIAN(PARTN:INTEGER):VECTOR;
begin
```

```
     BROWNIAN:STATE PARTN.POSITION
end;
block UPDATE;
begin
(*actualiza la posición de la particula)
end;
begin
(*inicializa la posición de la particula)
end.
```

Módulo de leyes de evolución basadas en ecuaciones diferenciales simultáneas.

La estrategia para introducir las leyes basadas en ecuaciones diferenciales en MIRANIM es:

1. En CINEMIRA-2 se escribe el procedimiento DIFFEQ para definir las ecuaciones diferenciales

2. En CINEMIRA-2 se escriben las leyes para obtener los valores de las variables de estado

3. En ANIMEDIT se definen los valores iniciales de las variables de estado y se aplican las leyes de variables animadas.

Por ejemplo considérese el problema de la expulsión del piloto, en el cual se tiene que determinar la combinación de la velocidad de la aereonave y la densidad del aire que permitan al avión estabilizarse verticalmente. Una velocidad alta y una altitud baja pueden servir para un eyección satisfactoria y las ecuaciones son:

```
dX/dt= V cos(theta)-VA
dY/dt= Vsin(theta)
dV/dt= 0                          para 0<=Y<Y1
     = -D/M-G sin(theta)          para Y>=Y1
d(theta)/dt= 0                    para 0<=Y<=Y1
           = -(G cos(theta))/V     para Y>=Y1
D=(RHO*CD*V)/2
```

donde

VA es la velocidad del avión
M es la masa del piloto + su asiento
G es la constante de gravitación
D es la fuerza de fricción
CD es la constante de fricción
Y1 es la longitud de los carriles de eyección
RHO es la densidad del aire

X,Y son las coordenadas de la trayectoria que se desea obtener y theta es el ángulo con la vertical

Para resolver el problema y obtener la solución en forma de leyes analiticas de evolución se diseñó un programa general en CINEMIRA-2 el cual resuelve ecuaciones diferenciales utilizando tres métodos : Euler, Euler modificado y Runge-Kutta (orden 4). A continuación se muestra la estructura del programa.

```
program CONTINUOS;
var
     Y:array (1..MAXSTATE) of REAL;
     P:array (1..MAXPAR)of REAL;
     G:array (1..MAXDERV) of REAL;
type
     SYSTEM=object(METH:INTEGER; NST:INTEGER);
          begin
               case METH of
               1: METHOD:= EULER;
               2: METHOD:= MODIFIEDEULER;
               3: METHOD:= RUNGEKUTTA
          end;
          if NST<=MAXSTATE then NSTATE:=NST
          else error
          end;
STATE=object(NUM:INTEGER;VAL:REAL);
          begin
               if NUM<=NSTATE then P(NUM):=VAL
               else error
          end;
PARAM=object(NUM:INTEGER; VAL:REAL);
          begin
               if NUM<MAXPAR then P(NUM):=VAL
               else error
          end;

procedure DIFFEQ;
{* Define las ecuaciones diferenciales bajo la forma
G(I):=f(Y(J), P(K)); *)
end;

block SIMULATION;
(*  responsable de la integración de las ecuaciones y del cálculo
de los valores de las variables de estado *)
end;

law RESULT:VECTOR;
(* El usuario debe definir algunas leyes para obtener el valor de
las variables de estado *)
end;
```

Módulo de explosiones

MIRANIM puede representar modelos de explosiones de objetos, una explosión se define como un procedimiento objeto:

```
PROCEDURAL <nombre de la explosión> EXPLOSION
          <objeto gráfico>
          <intensidad de la explosión>
          <número de fragmentos>
          <centro de la explosión>
          <velocidad de la explosión>
```

```
VARIABLES                          Entra en modo variable
LAW POSITION, FRAGMENT, 1          Define una ley POSITION que
                                   regresa la posición del
                                   primer fragmento
VECTOR EYE,A                       Define una variable animada
EVOLUTION EYE,POSITION,0,10        Aplica la ley POSITION a EYE
                                   por 10 segundos
CAMERA                             Entra en modo cámara
CAMERA CAM,EYE, INTEREST           Define una cámara virtual CAM
```

A continuación se presenta el libreto de animación.

```
OBJECT                                  Entra a modo objeto
READ BODY, BODYFILE                     Lee un objeto
PROCEDULAR EXPL, EXPLOSION              Prepara una explosión

VARIABLES                               Define las constantes
VECTOR INT,C,0,60,0
VECTOR EYE,C,0,60,-125
VECTOR SOURCE,C,0,60,-300
VECTOR COL,C,0.9,0.8,1

CAMERA                                  Entra a modo cámara
CAMERA CAM,EYE,INT                      Define una cámara CAM
BACKGROUND CAM,COL                      Define un fondo color COL

LIGHT                                   Entra a modo Luz
SOURCE SOU, POINT,P                     Define una fuente de luz

DIRECTOR                                Entra a modo director
CAMERA CAM,0,1,0                        Activa la cámara CAM
SOURCE SOU,0,1,0                        Activa la fuente SOU
BLOCK TRACE,0,1                         Activa el bloque TRACE
SHOW 0,1,6,A,S                          Lo muestra.
```

La posibilidad de definir cualquier ley de evolución dentro de un sistema de animación provee nuevas posibilidades para el movimiento y rompe con el enfoque tradicional del uso de muchas operaciones matriciales.

Esta fase del sistema aun se encuentra en su etapa de experimentación y se espera poder realizar diseños novedosos con la introducción de este tipo de técnicas.

3.7.6 MANEJO DE CAMARA

En Abril del presente año, los autores presentan un nuevo artículo que trata sobre la implementación de cámaras virtuales. El artículo describe las diferentes técnicas que se pueden usar como el zoom y el recorte entre otras.

Una cámara virtual simple consiste de: un par de vectores característicos , el ojo y el punto de interés. Una cámara virtual se define normalmente como el programa responsable de las transformaciones del espacio 3D a su proyección 2D. Esto consiste de una serie de subprogramas o un conjunto de matrices aplicadas a los objetos
La cámara virtual debe manipularse de la misma forma que un camarógrafo manipula una cámara real.

El papel de una camara virtual es efectuar las transformaciones geométricas necesarias para convertir puntos 3D en puntos en el plano imagen en 2D. La cámara virtual tiene un nombre y se caracteriza mediante dos parametros, el ojo y el punto de interés por ejemplo :

```
CAMERA MYCAMERA, EYE, INT
```

EYE es un punto (o vector) que representa la localización de la cámara, el punto de interés es el punto hacia donde la cámara se direcciona. En el sistema ambos puntos se definen como vectores y si son constantes, se indican sus valores por ejemplo :

```
VECTOR EYE, C, <-10,-20,100>
VECTOR INT, C, <0,0,0>
```

Acercamiento

Una lente de acercamiento permite al camarógrafo enfocar rapidamente el campo a filmar En una cámara virtual el acercamiento se define por el cambio del radio, las dimensiones del espacio de la imágen y el espacio de la pantalla.

```
ZOOM MYCAMERA, ZOOMVALUE
```

Donde ZOOMVALUE se definió previamente como un número real constante. Por ejemplo

REAL ZOOMVALUE, C, 2.5

Giro

El ojo y el punto de interés, definen una línea y una dirección, en donde existe la posibilidad de la rotación en un ángulo dado alrededor del eje. Esta característica de la cámara se llama giro, el valor de omisión es el ángulo cero. En la figura se muestra una camara virtual con punto de interés y giro y se define como

SPIN MYCAMERA, SPIVALUE

donde el SPINVALUE se define como :

REAL SPINVALUE,C,1.5

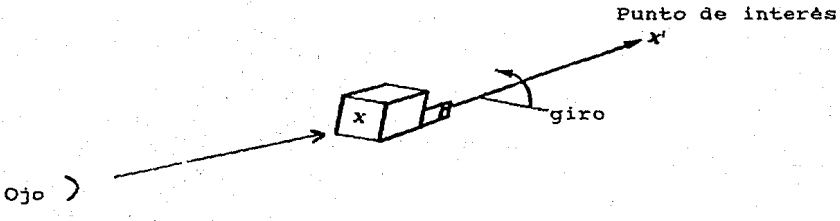


FIG. 3.25 Efecto de giro producido por una cámara virtual

Puertos de visión y recorte.

El puerto de visión se define como la porción o area donde se despliega la gráfica que corresponde a la ventana en el espacio de la imágen y se define como :

VIEWPORT MYCAMERA, V, VIEW1, VIEW2

donde VIEW1 es el punto inferior izquierdo del VIEWPORT y VIEW2 el punto superior derecho. Por ejemplo :

VECTOR MYCAMERA, VIEW1,C, <0,0>
VECTOR MYCAMERA, VIEW2,C, <0.5,0.5>

Generalmente toda la imágen fuera del puerto de visión se recorta, pero en algunos casos puede ser útil suprimir este recorte automático o recortar todo lo que esté dentro del puerto de visión en lugar de lo que esté afuera. Por ejemplo se define :

CLIPPING MYCAMERA,I,CLIP1,CLIP2
ó
CLIPPING MYCAMERA, O,CLIP1,CLIP2

para un recorte dentro o fuera del rectángulo, definido por el punto inferior izquierdo CLIP1 y el punto superior derecho CLIP2.

Filtros de color.

Los filtros de una cámara son lentes de vidrio que se colocan enfrente del lente principal. Estos filtros se usan para corrección de los colores o para efectos especiales. En una cámara virtual los filtros de color se simulan facilmente modificando los colores durante el proceso de despliegue. En el sistema se utilizan cuatro tipos de filtros de color:

1. Filtros de color para reemplazo (REPFILTER): Reemplazan uno o mas componentes RGB con un nuevo valor.

2. Filtros de color aditivos (ADDFILTER) Adicionan valores constantes a uno o mas componentes RGB

3. Filtros de color Substractivos. (SUBFILTERS) Sustraen valores constantes de uno o mas componentes RGB.

4. Filtros de color de límite de color (LIMFILTER): Caracterizados por dos vectores <r1,g1,b1> y <r2,g2,b2> los cuales definen un rango. Estos filtros, reemplazan cualquier componente RGB fuera del rango con el valor límite. Por ejemplo si el componente R es menor que R1, es reemplazado por R1.

Filtros de niebla.

Un filtro de niebla se simula por la combinación de una imágen con el color de fondo. La niebla se espesa cuando la distancia entre el punto considerado y el ojo es grande esto significa que la niebla es una característica de la cámara. Se define un filtro de niebla para la camara CAM como sigue:

FOG CAM, DIST

donde DIST es la distancia a la cual el color se mezcla en un

50%, con el color de fondo.

Camaras estereoscópicas

Las imágenes producidas en una terminal con un modelo de cámara virtual tienen un defecto, sólo se simula un ojo.
En la realidad el cerebro recibe dos imágenes una por cada ojo, como los ojos no están exactamente en la misma posición aparece un efecto de profundidad. Con un solo ojo, se suprime este efecto. Para solucionar este problema se usan cámaras virtuales estereoscópicas.

STEREOSCOPIC MYCAMERA,12

Define una cámara estereoscópica con una separación de 12 unidades entre los dos ojos. Se producen dos vistas de todas las escenas una para cada ojo la primera corresponde al ojo derecho y la segunda al izquierdo.

Un modelo de cámara móvil

La diferencia entre una cámara normal y una cámara móvil es que esta última toma varias fotos de una escena durante un periodo de tiempo. La velocidad de la cámara se define como el número de imágenes tomadas en un segundo y típicamente es de 18 a 24. La cámara móvil se considera como una cámara virtual con velocidad.

SPEED 24,1

En este caso la velocidad es de 24 cuadros/seg, el segundo número indica cuantos duplicados de cada cuadro se producen.

"Panning", "tilting" y "tracking"

El efecto de "panning" es aquél en el cual la cámara se mueve horizontalmente de un punto a otro punto . En el efecto de "tilting" la cámara se mueve verticalmente de un punto a otro. Finalmente "tracking" significa mover la camara hacia el punto de interés.
Puede definirse cada uno de estos movimientos especificando las leyes de variación del ojo o del punto de interés

VECTOR INTEREST, A, <10,30,50>, <50,30,50>
LAW PAN, LINEAR
EVOLUTION INTEREST, PAN, 0, 10:

define un efecto de "panning" de la posición <10,30,50> a la posición <50,30,50>, que varia linealmente en 10 segundos

```
VECTOR INTEREST,A, <10,30,50>, <10,100,50>
LAW TILT, ACC
EVOLUTION INTERES, TILT, 0,8:
```

define un efecto de "tilting" de <10,30,50> a <10,100,50> en 8 segundos de aceleración

```
VECTOR EYE, A, <10,30,50>, <1,3,5>
LAW TRACK, ACCDEC
EVOLUTION EYE, TRACK,0,6
```

define un efecto de "tracking" de <10,30,50> a <1,3,5> en 6 segundos de aceleración con una aceleración seguida de una desaceleración.

Acercamiento y giro

En una cámara móvil es posible cambiar el acercamiento y el giro continuamente. Esto significa que pueden definirse valores de acercamiento o giro que dependan del tiempo ej.

```
REAL ZOOMVALUE,A,1,10
LAW VARIATION, LINEAR
EVOLUTION ZOOMVALUE, VARIATION, 0,10
ZOOM MYCAMERA, ZOOMVALUE
```

define un efecto de acercamiento de 1 a 3 con variación lineal durante 10 segundos.

```
REAL SPINVALUE, A, 0, 360
LAW VAR2, ACC
EVOLUTION SPINVALUE, VAR2, 0, 8
SPIN MYCAMERA, SPINVALUE
```

define un efecto de giro de 0 a 360 grados con una variación lineal durante 8 segundos

Ruta de la cámara arbitraria

Hay veces que es mas conveniente diseñar una trayectoria no lineal para la cámara, esta trayectoria se denomina ruta de la cámara y cuenta con 3 formas de definirla.

El primer camino es aplicar una de las leyes del editor MIRANIM a la variable ojo: movimiento armónico, movimiento circular, movimiento "fuzzy" o movimiento de péndulo. Por ejemplo una cámara que se mueve circularmente alrededor de un eje vertical <0,1,0> pasa a través del punto <5,6,7> con una velocidad angular de un radian por segundo y no tiene acelaración angular se define

como :

```
VECTOR EYE, A, <10,20.30>
LAW TURN, CIRC, <5,6,7>, <0,1,0>,1,0
EVOLUTION EYE, CIRC, 0,10
CAMERA MYCAM, EYE, INT
```

Cámara asignada a un actor.

El propósito principal de una cámara es ver una escena. Hasta ahora se le ha considerado como si viera solamente objetos estáticos decorativos que no interaccionan con objetos dinámicos como son los actores.

```
ACTOR BALL, BALLOBJ
ROTATION BALL, Y, PY, ANGLE
MOVE BALL, POS
```

BALLOBJ es un objeto esférico, PY es un punto móvil, ANGLE es un ángulo dependiente del tiempo y POS es un punto móvil.

Cuando el actor se mueve, el movimiento de la cámara debe ser similar o sea que el ojo o el punto de interés de la cámara deben seguir al actor :

```
DEPENDENT MYCAMERA, E, BALL
o
DEPENDENT MYCAMERA, I, BALL
```

El primer comando indica que el ojo siga al actor BALL y el segundo que el punto de interés siga al actor.

Efectos de matiz para actores, basados en recorte poligonal

El efecto de matiz se define como un actor y se asigna a una cámara. Se aplican entonces transformaciones al matiz durante el proceso de animación. Por ejemplo, el matiz puede cambiar utilizando una interpolación.

```
ACTOR POLY,SQUARE
INBETWEEN POLY, HEART, VAR
CAMERA CAM, EYE, INT
MATTE CAM,POLY
```

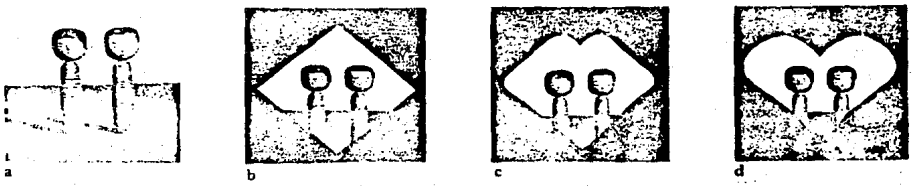


FIG. 3.26 Efectos de matiz

Efectos con varias cámaras virtuales móviles

Usando varias cámaras se simulan efectos especiales como los que se producen con impresoras ópticas. Una impresora óptica es una cámara que se ajusta en la entrada de una lente de un proyector movible, para duplicar una parte de una película en otra. Estas impresoras se utilizan para superimposiciones y efectos de imágenes múltiples, también para efectos de desaparecer y aparecer escenas.

Un "fade in" se utiliza al principio de una escena, para que la escena aparezca gradualmente desde el negro. Un "fade-out" se utiliza al final de la escena, la cual se oscurece gradualmente hasta el negro. Los efectos de "fade" se logran aplicando las leyes de la evolución a los parámetros del filtro de color en la cámara virtual móvil.

Cuando se utiliza un efecto de "wipe", una escena aparece a un lado, sobre la escena precedente.

Se presenta en seguida un ejemplo de un efecto "wipe", utilizando s camaras virtuales CAM1 Y CM2

```
VECTOR EYE,C, <0,60,-150>        mismo ojo para ambas cámaras
VECTOR INT1,C <0,0,50>           punto de interés para la primer
                                 cámara
VECTOR VA,C, <0.25,0.25>         esquinas del puerto de visión
VECTOR VB,C <0.75,0.75>          para la segunda cámara
VECTOR INT2,C,<30,60,-300>       punto de interés para la segunda
                                 cámara

CAMERA                           entra a modo cámara
CAMERA CAM1, EYE,INT1
CAMERA CAM2, EYE, INT2
CAMERA CAM2, T, VA, VB
```

CLIPPING CAM1, VA, VB

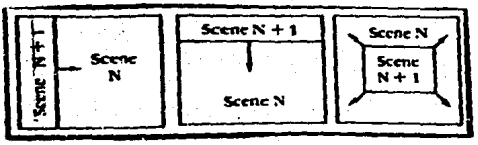


FIG. 3.27 Efecto de "wipe"

CAPITULO IV. ANIMACION EN MICROCOMPUTADORAS

4.1 INTRODUCCION

Los sistemas presentados en los capitulos anteriores se han implementado en sistemas de cómputo muy costosos. Los sistemas para microcomputadoras pueden ser una alternativa para realizar animación.

Muchas microcomputadoras empiezan a ofrecer un amplio rango de colores, resolución elevada y otro tipo de facilidades que sólo se habían encontrado en sistemas grandes y costosos.

En paralelo con estos avances, diversas compañías se encuentran produciendo dispositivos de entrada y salida para microcomputadoras.
Uno de los problemas que se presenta es la no-estandarización en los programas de animación. Los programas que se han desarrollado dependen de las caracteristicas físicas de la microcomputadora.

La mayoría de los programas desarrollados tienen un enfoque de entretenimiento o educacional y por lo regular son en 2D.

BASIC es el lenguaje mas popular para gráficas en microcomputadora. Para entender mejor las capacidades de la máquina puede examinarse las instrucciones de BASIC. Este lenguaje ofrece primitivos gráficos que son ciertas secuencias de caracteres que se envían a un programa especial llamado manejador.

Aunque BASIC es el lenguaje mas popular no se puede decir que es el único, otro lenguaje es Pascal. Como Pascal es un lenguaje compilado sus rutinas se ejecutan mas lentamente que BASIC .

Otro lenguaje gráfico que ha crecido en popularidad es LOGO el cual está basado en la "geometría de la tortuga". Este lenguaje permite a los usuarios ver una tortuga imaginaria, la cual puede moverse con comandos simples. Para los niños este lenguaje es muy fácil de asimilar ya que sus movimientos son muy sencillos. Por ejemplo para dibujar una caja en LOGO se tiene:

```
TO BOX: SIDE
HOME
REPEAT 4 FORWARD : SIDE LEFT 90
END
```

Para hacer la misma caja en Applesoft BASIC se tiene:

```
100 HGR: REM limpia la pantalla
110 H COLOR = 3 : REM establece el color blanco
120 XC= 140 : YC= 80 :REM  coordenadas del centro
130 INPUT "Teclee la longitud del lado "; SI
140 HPLOT XC,YC TO XC, YC-SI TO
    XC-SI, YC-SI TO XC-SI, YC TO XC, YC
150 END
```

Este programa limpia la pantalla, fija las coordenadas XC y YC, pide al usuario que teclee la longitud de los lados y después la instrucción HPLOT dibuja la caja.

El lenguaje C también se utiliza para hacer gráficas. C es similar a Pascal pero es mas útil para crear programas que manipulen bytes y bits y se ejecuta mas rapidament.Con C, una rjeta gráfica S-100 y un mapa de bits de alta resolución se puede tener una máquina potente de bajo costo.
También puede mencionarse a Forth. Aunque es mas dificil de aprender , es un lenguaje muy "elegante" y se le pueden añadir sus propias instrucciones gráficas. Sus ventajas incluyen alta velocidad, ejecución inmediata de programas (no se compila), habilidad para definir sus propios comandos y código compacto.

El lenguaje ensamblador es otra opción que es necesario usar para obtener animación en tiempo real. Permite el movimiento rápido y fluido de los objetos en la pantalla. Se han creado un conjunto de rutinas de extensión gráfica en lenguaje ensamblador 6502 para dibujar circulos, poligonos y llenar formas con colores, estas rutinas son dificiles de crear y requieren cientos de horas de programación.

4.1.1 Instrucciones gráficas de los lenguajes.

Se presentan ahora las diversas facilidades para selección de modos, colores, dibujado de lineas, creación de sombras, rellenado, definición de objetos y otros usos .

Páginas.

Las páginas son secciones de memoria que se utilizan para guardar el contenido de la pantalla. Existen varias páginas pero solamente una estará activa.
Por ejemplo cuando la IBM se usa en modo texto, tiene 8 páginas, una de las cuales es la página activa y otra la página de salida. La página de salida recibe los resultados de cualquier instrucción PRINT. La TRS-80 permite dibujar imágenes gráficas en varias páginas y luego desplegarlas. La Apple tiene dos páginas para gráficos de alta resolución. La idea de las páginas es permitir la generación de gráficas en la página de salida mientras

el usuario está viendo la página activa. Esto permite que la nueva figura sea desplegada inmediatamente antes de que se borre la figura original. Si se tuviera que borrar la figura original, se provocaría parpadeo, y de ésta manera el parpadeo no existe. Este método permite al programador crear animación mostrando el contenido de cada página.

Selección de color

Los colores pueden seleccionarse automáticamente mediante una instrucción de modo o con una instrucción especial para color. En algunas computadoras la instrucción para selección de color permite escoger los colores para el frente, el fondo y el borde. Las computadoras que tienen la facilidad de registros de color, normalmente tienen una instrucción para seleccionar cual registro de color se utilizará para pintar un "pixel" y una segunda instrucción establecerá el valor del registro.

Colores disponibles

La opción de color en las computadoras personales esta muy limitada en relación con las máquinas de alta técnica.

En algunas computadoras como la IBM existen 8 colores con 2 intensidades (high y low) haciendo un total de 16 . Mucha marcas incluyen blanco y negro cuando especifican el número de colores disponibles. En algunas computadoras como la Apple, sólo se tienen 6 colores disponibles. La ATARI utiliza 128 colores en algunos modos y 256 colores en dos modos especiales.

Los nombres que se usan para los colores tampoco son estandar, por ejemplo en una compañía se le llama aqua y en otra azul-verde.

Punteo

El punteo es una función de graficación fundamental que consiste en el uso de coordenadas verticales y horizontales para iluminar un punto en la pantalla. Algunas veces la instrucción para punteo es PSET, PLOT, POINT o HPLOT . En algunos casos es necesario especificar el color dentro de la instrucción y en otros el color se especifica aparte mediante la instrucción SET COLOR por ejemplo. Algunas computadoras como la IBM PC ofrecen un comando especial para borrado de puntos de color que es PRESET X,Y.
En algunos sistemas donde las coordenadas X,Y son relativas se usa la palabra STEP en la instrucción PLOT. Esto significa que el siguiente punto se graficará X,Y localidades más allá del

último punto. La función plot es probablemente la función gráfica mas importante que se puede utilizar dentro de una computadora y usualmente es muy lenta para animación en tiempo real en BASIC. Con C y Forth puede ser suficientemente rápida.

Dibujado de lineas

El dibujado de lineas es una instrucción gráfica importante, porque permite dibujar objetos de muchos lados y elimina la necesidad de repetir la instruccion PLOT varias veces. Las instrucciones para dibujado de lineas pueden ser tan simples como HPLOT x1, y1 TO x2, y2 en la Apple o LINE(x1,y1)-(x2,y2) en la IBM y en la TRS-80. En algunos casos se cuenta con la opción para dibujar un rectángulo.

Formas

Las formas son una facilidad creada para la Apple II. Las formas son objetos gráficos compuestos de vectores, los cuales son pequeños segmentos de línea que pueden dibujarse en cuatro direcciones (arriba,abajo, derecha e izquierda). Una forma se puede integrar de unos cuantos vectores o de cientos de ellos. Una vez que se crean estas formas, pueden dibujarse en la pantalla con una instrucción como: DRAW 1 AT X, Y. También existen conjuntos de instrucciones adicionales para manipularlas incluyendo las que escalan y rotan el objeto.

Estas instrucciones pueden elaborar formas increibles para juegos o programas donde los objetos vuelan, giran o cruzan la pantalla. Se tienen limitaciones para animación en tiempo real porque las instrucciones de BASIC son muy lentas.

Lenguaje de definición de gráficas

Este lenguaje es una facilidad que se encuentra comunmente en la IBM PC y en la TRS-80. El LDG es un conjunto de comandos que se colocan en variables tipo cadena. Estos comandos de dibujo especifican la manera en que los segmentos de línea aparecerán en la pantalla.

Se cuenta con un conjunto muy completo de comandos como son: DRAW UP, DOWN, RIGHT, LEFT. El LDG es muy lento para animación cuando el número de vectores que componen la forma es mayor a 10. Un problema con las formas de el LDG es que los vectores pueden estar sólo en 8 direcciones, esto significa que las curvas suves son imposibles de dibujar.

Rellenado

Rellenar significa llenar con color un espacio delimitado. Existen diferentes tipos de llenado, la principal diferencia reside en la habilidad de envolver esquinas , rellenar rincones y rendijas de un area cerrada. Los rellenados son normalmente lentos, algunos tienden a distorsionar el color de los objetos yacentes.

Instrucciones de definición de objetos.

Frecuentemente es necesario dibujar ciertas formas geométricas en la pantalla como son cuadros, triángulos, polígonos, etc., esto es fácil utilizando una instrucción como LINE. El procedimiento es mas complejo cuando se involucran curvas, el dibujar un simple círculo puede ser complicado para gente que no es matemático porque se requieren funciones como SIN y TAN. Un camino para eliminar este problema es con el uso de instrucciones de definición de objeto. Una de estas instrucciones es el comando CIRCLE (disponible en la IBM, TRS-80 y VIC-20). Esta instrucción permite dibujar un círculo, una elipse o un arco en cualquier localidad x,y de la pantalla. Desafortunadamente es muy lento para animación.

Instrucciones gráficas misceláneas.

Otras instrucciones gráficas útiles son las que permiten limpiar la pantalla con un cierto color. La instrucción WIDTH para controlar el número de columnas de texto a ser desplegadas y las funciones que regresan el valor ASCII de un caracter en una columna o renglón especial. También es útil la función POINT que regresa el color de una localidad especifica de la pantalla, la instrucción LOCATE para posicionar el cursor, el comando VIEWPORT que construye una ventana rectangular en la pantalla a la cual se restringen los dibujos y la instrucción PAGE COPY para mover la información gráfica de una página a otra.

4.1.2 FACILIDADES GRAFICAS DE LAS MICROCOMPUTADORAS

MODO GRAFICO Y MODO TEXTO

Cada marca tiene su propia forma para definir los numerosos modos en los que la computadora va a funcionar. El modo gráfico (también llamado modo mapa) coloca a la pantalla en la posibilidad de responder a las instrucciones gráficas de un lenguaje, y el modo texto es para el despliegue de programas, palabras etc.

Normalmente el modo texo se utiliza para el desarrollo de

programas y el modo gráfico para la ejecución de programas gráficos. Los textos y las gráficas pueden mezclarse por diferentes métodos que varian de una máquina a otra.
Por ejemplo cuando la Apple está en modo gráfico, puede aparecer texto solo en una ventana o en las cuatro lineas del final de la pantalla, en cambio en la IBM puede desplegarse texto en cualquier lugar, en modo gráfico.
La mezcla de estas facilidades es importante para el area de negocios en donde se necesita que las gráficas tengan sus respectivos letreros .

Modo texto

En modo texto debemos tener en cuenta varias cosas, como son: el número de puntos por caracter, el número de caracteres por linea y el número de lineas por pantalla. Estos números corresponden al grado de resolución en modo gráfico. En referencia a la matriz de puntos que forma cada caracter se tiene una matriz minima de 5 x 7 y una máxima de 8 x 8. El número actual de caracteres en una linea varia desde 20 hasta 80 el número de lineas en la pantalla varia de 16 a 25, 24 es el mas popular.

En algunos modos texto se permite utilizar colores. Esto es de gran beneficio para aplicaciones como los procesadores de palabra.

SELECCION DE RESOLUCION

También se puede seleccionar la resolución que se desea, la IBM PC, por ejemplo tiene resolución media y alta, la diferencia entre ambas es el número de pixels y el número de colores permitidos.

Que es alta resolución ?

Existe un fenómeno interesante acerca de las diferentes resoluciónes en términos de alta, media y baja ya que esto tiene significados totalmente diferentes de una marca a otra. Por ejemplo la alta resolución en la VIC-20 es mas que la baja resolución de la ATARI y la alta resolución de esta última es como la resolución media de la IBM PC . La alta resolución de la IBM puede considerarse como la resolución baja de las máquinas de alta técnica. La realidad en el uso de estos términos reside en el número de elementos verticales y horizontales que pueden usarse en la pantalla. Por un buen tiempo la resolùción mas alta en microcomputadoras fue la de la Apple II (280 x 192), después la IBM desarrollo 640 x 200.

Colores

Otro elemento importante a considerar es la capacidad de color. Existen colores de fondo, de bordo y de frente. Color de fondo se refiere al color que toma la pantalla cuando se limpia. Los colores de frente son aquellos que pueden aparecer arriba del color de fondo. Los colores de bordo son los que permiten delimitar los perimetros del area activa. En las computadoras que ofrecen las tres opciones, existe un cierto número de colores que pueden ser usados en cada area. Por ejemplo en resolución media se permiten 16 colores de fondo y 3 colores de frente. Algunas computadoras no tienen selección de colores (Apple).

Máximo color

El máximo número de colores permitido en cualquier modo es función de la resolución. Al incrementarse la resolución se decrementa la capacidad de color. Esto es una consecuencia del hecho de que los puntos que se despliegan están codificados en un area de memoria. Si la medida de la memoria es fija y la resolución se incrementa, entonces la capacidad que mas se afecta es el color . Esto se ilustra claramente en el caso de la Atari la que en resolución media ofrece un máximo de 16 colores y en resolución alta ofrece solo 2.

Arreglos de Imagen

Esta es otra forma de graficar objetos y se realiza mediante las instrucciones GET y PUT. La instruccion GET se utiliza para guardar un objeto en un arreglo bidimensional (una matriz de puntos/bits apagados y prendidos). El par de coordenadas x,y en la instrucción GET especifican el area de la pantalla que se guardará en el arreglo. La instrucción PUT se utiliza para dibujar el objeto . Como el objeto se dibuja utilizando el mapa de bits guardado en el arreglo, pueden usarse instrucciones adicionales para controlar la forma en la que los bits del objeto interactuan con la imágen que se encuentra en la pantalla. Estas instrucciones permiten las operaciones lógicas como AND , OR y XOR entre el contenido del arreglo y lo que se encuentra en la pantalla. En escencia esto significa que pueden dibujarse objetos en un fondo, sin tener que borrarlo.

Como es de esperarse no todos las máquinas ofrecen GET y PUT. El BASIC de la IBM y de la TRS-80 de color si lo tienen. Este método es muy lento para animación compleja pero puede ser útil para movimientos lentos.

Jugadores y Espectros

Los jugadores y espectros son objetos gráficos que pueden moverse usando equipo costoso. ATARI los llama jugadores, y Texas Instruments espectros, pero su función es similar en naturaleza. Los jugadores resuelven un gran problema de gráficos, ya que están separados del fondo de la escena y no requieren de borrados complejos para moverse en la pantalla, son fáciles de actualizar con los métodos normales de graficación y no interfieren con otros objetos.

Los jugadores de ATARI tienen sus propios registros de color, y pueden combinarse para crear objetos.

Los espectros difieren un poco de los jugadores. El espectro es un caracter largo (16 x 16), mientras que un jugador tiene 8 bits de ancho, con un maximo de 256 bits de alto. Los espectros son mas potentes que los jugadores para movimiento en la pantalla, ya que una vez que se ponen en movimiento seguirán moviéndose como se les indique. El espectro tiene un número especial de comandos para moverse, incluyendo MOTION para especificar velocidad y dirección, COINC para detectar coincidencias de espectros (colisiones), DISTANCE para definir la distancia entre dos de ellos y MAGNIFY para cambiar su medida. En algunos casos los espectros están disponibles solamente en TI 99/4, pero se implementan mediante un chip TI especial que se vende en el mercado, por lo que puede comprarse una tarjeta para la Apple que tenga la facilidad de espectros.

Los jugadores y espectros son ideales para animación, ya que fueron diseñados para este propósito.

EFECTO DE "SCROLL" POR MEDIO DE EQUIPO

Convencionalmente el "scroll" se realiza en el area de despliegue de memoria, resultando un efecto visual lento y ondulante. Con el efecto de "scroll" por medio de equipo sólo necesita cambiar los apuntadores de dos bytes para hacer que la pantalla entera se mueva hacia arriba, abajo, derecha, izquierda o diagonalmente, resultando un efecto rápido y suave.

Hay dos diferentes tipos de "scroll" por medio de equipo el burdo y el fino. El "scroll" burdo mueve la ventana de la pantalla muchos bytes a la vez, mientras que el scroll fino, mueve la pantalla solamente en un punto propiciando un efecto continuo y

suave.

Aunque el "scroll" por medio de equipo es perfecto para animación de fondos,rara vez se encuentra en las computadoras comunes. La única computadora personal que actualmente posee esta facilidad es la ATARI.

CARACTERES GRAFICOS

Muchas computadoras personales tiene adicionalmente al juego normal de caracteres, un conjunto de caracteres gráficos. Usualmente son figuras simples como cajas, segmentos de linea, circulos, caritas sonrientes y esquinas, entre otros. En animación estos caracteres suelen ser muy útiles.

Una propiedad muy importante para la animación, es la habilidad de crear y manipular objetos que se construyeron mediante un conjunto de caracteres definido por el usuario.

Esta facilidad está disponible en la mayoria de las computadoras personales de hoy en dia.

Algunas computadoras como Apple, tienen programas especiales que facilitan la creación y uso de estos caracteres.

REGISTROS DE COLOR

Los registros de color son facilidades que apenas empiezan a aparecer en las computadoras personales, se implementaron primero en la ATARI y ahora en la VIC-20. Los registros de color proveen un camino indirecto para especificar el color de un punto a la vez que dan mayor poder y flexibilidd al control de gráficos. Las computadoras personales usan los registros de color de una manera similar a las computadoras de alta técnica, con la excepción de que no son tan anchos, ni tan numerosos (9 en la ATARI , 4 en la VIC-20). Con suficientes registros de color puede realizarse animación a color.

INTERRUPTORES DE BLANCO VERTICAL.

Cada 1/60 de segundo se redibuja la pantalla completa. Entre el periodo en que se finaliza una pantalla y se comienza la siguiente,existe un lapzo de tiempo llamado " blanco vertical". Si la computadora lo permite, el microprocesador puede interrumpirse en este punto y ejecutar un programa en lenguaje máquina. Esta rutina puede usarse para procesar animación de fondo, lo

cual significa que pueden tenerse ciertos eventos gráficos que ocurran automáticamente, como el movimiento de un objeto o el tocar música. La habilidad para interrumpir el microprocesador durante el periodo de blanco vertical se llama interruptor de blanco vertical y es otra facilidad que está disponible en la ATARI.

LISTA DE DESPLIEGUE

Las listas de despliegue son populares es computadoras de animación de alta tecnica, pero rara vez en computadoras personales. Una lista de despliegue es una sección de memoria que contiene un conjunto de instrucciones gráficas para un procesador gráfico. Está disponible en la ATARI.

4.2 ANIMACION CON CARACTERES GRAFICOS

Con este método se usan los caracteres convencionales de la computadora, como son: las letras del alfabeto, números, puntuación y caracteres gráficos especiales para dibujar figuras.

Un conjunto de caracteres gráficos se integra de líneas rectas, diagonales, esquinas, cuadrados y círculos. Cuando todos estos elementos se juntan, puede crearse una figura rudimentaria. Las computadoras como la IBM PC y los cuatro modelos de la Commodore (PET,VIC, CBM y COMMODORE 64), tienen un conjunto de gráficas integrado. Mientras mas grande sea la variedad de caracteres es mas grande la flexibilidad que un animador tiene para crear figuras en baja resolución.
Como antes se mencionó, este tipo de animación va unido al tipo de máquina que se esté usando.

Como ejemplo veremos paso por paso como funciona el programa que produce la animación del pájaro volador en una computadora ATARI.

```
10 REM  *******  FLYING BIRD *****
20 REM
30 REM
40 REM demostración de la animación por caracteres
50 REM gráficos
60 REM
100 REM INICIALIZACION
110 DIM BIRD1$(17), BIRD2$(17), BIRD3$(16), BIRD4$(16)
120 BIRD1$="(DOWN) (F)(T)(G) (DOWN)(5 LEFT)(F)    (G)"
130 BIRD2$="(DOWN)(F)(M)(T)(M)(G)(DOWN)(5 LEFT)      "
140 BIRD3$="      (DOWN)(5 LEFT)(2 M)(T)(2 M)"
150 BIRD4$="(G)   (F)(DOWN)(5 LEFT) (M)(T)(M) "
160 POKE 752,1
170 PRINT "(CLEAR)"
180 REM
200 REM CICLO DE ANIMACION
210 FOR I = 1 TO 6
220     POSITION 17, 10
230     ON I GOSUB 310,320,330,340,330,320
240     FOR W =1 TO 25
245     NEXT W; REM pausa
250 NEXT I
260 GOTO 210
270 REM
300 REM   DIBUJADO DEL CUADRO
310 PRINT BIRD1$;:
315 RETURN
320 PRINT BIRD2$;:
325 RETURN
330 PRINT BIRD3$;:
335 RETURN
340 PRINT BIRD4$;:
```

345 RETURN

En la linea 110 se dimensionan las variables tipo string que se usarán en este programa .El número entre parèntesis indica el máximo número de caracteres que cada string puede contener. Las cuatro variables tipo string (BIRD1$, BIRD2$, BIRD3$ y BIRD4$) mencionadas en las líneas 120 a 150: contienen tres diferentes tipos de caracteres como sigue:

1. Los caracteres gráficos que forman al pájaro.

2. Los caracteres de control de cursor los cuales mueven el mismo antes de imprimir un caracter gráfico .

3. Los espacios que se usan para borrar secciones de cuadros anteriores.

El ciclo de animación (línea 200-270) contiene la lógica que hace que se impriman los cuadros en el orden correcto. Lo único que se tiene que se hacer es poner el cursor a la mitad de la pantalla con el comando adecuado (linea 220) e imprimir los cuadros.

El ciclo completo del movimiento de las alas del pájaro consiste de 6 cuadros 2 de los cuales se repiten.

4.2.1 JUEGO DE CARACTERES DEFINIDO POR EL USUARIO

El juego de caracteres predefinidos en la computadora es muy limitado, de ahi la necesidad de poder definir un juego de caracteres propio y adecuado al tipo de figuras que se desean construir.

Se pueden crear caracteres individuales los cuales se desplegarán en conjunto para que aparezca un número de formas perfectamente diseñadas. Muchas computadoras como la IBM PC, la Apple II y la Apple III tienen esta posibilidad. Las figuras animadas pueden crearse con un alto grado de detalle en lugar de estar limitadas por el juego de caracteres que tiene integrado la máquina.

Cuando se enciende la computadora ATARI aparecerán varias palabras en la pantalla. Lo que sucede dentro de la computadora para que esas palabras aparezcan, es que una serie números clave se depositan en una parte de la memoria RAM llamada memoria para la pantalla. Una clave por cada caracter.

Estas claves que se encuentran en ROM se llaman juego de caracteres.

Cada juego de caracteres está compuesto por puntos en una matriz de 8 puntos de ancho X 8 puntos de alto. Cada uno de esos 64 puntos pueden apagarse o prenderse pudiendo asi definir un caracter. La información que describe cuales puntos deben prenderse o apagarse en cada caracter se llama "definición de caracter".

Las figuras muestran el arreglo de puntos que forman la letra A.

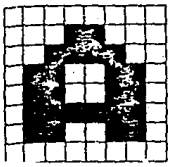
Apariencia de la letra en la pantalla	Representación binaria	decimal
	00000000	0
	00011000	24
	00111100	60
	01100110	102
	01100110	102
	01100110	126
	01100110	102
	00000000	0

Fig. 4.1 Definición de caracteres para la letra "A"

En muchas computadoras personales el juego de caracteres integrado es todo lo que se puede tener, pero la ATARI acepta un juego de caracteres diseñado por el usuario. El juego de caracteres en ROM es permanente y no se puede cambiar ninguno de los caracteres definidos en el ROM sin embargo si se quiere crear un juego de caracteres propio se pueden enviar al RAM mediante la función POKE.

Para informar a la computadora como encontrar este juego de caracteres diseñado por el usuario existe una localidad de memoria reservada para este fin. Esta localidad de memoria en RAM siempre tiene la dirección de la página del juego de caracteres en uso.

Después de que se ha diseñado el juego de caracteres se debe encontrar un lugar seguro en la memoria para guardarlo. Una ubicación conveniente es abajo de la memoria de pantalla.

Para crear un juego de caracteres especial es recomendable hacer lo siguiente:

1. Obtener una hoja de papel para graficar, preferentemente de 8 cuadros por pulgada.

2. Decidir el tamaño de la matriz de caracteres que se quiere que ocupe la figura y dibujarla en el cuadriculado

3. Dibujar la figura que desea representar, dentro de la matriz de caracteres

4. Calcular el valor decimal del caracter que aparece en cada una de las celdas que contiene cada linea

6. Introducir los bytes resultantes en el programa

Una manera mas eficiente es el uso de programas para edición de caracteres o editores de caracteres.

Un producto como éste permitirá trabajar con los caracteres en un ambiente interactivo de esta forma se podrán ver los caracteres en la pantalla a medida que se van editando.

La computadora se encargará de hacer los cálculos necesarios para determinar los bytes que tiene cada caracter.

Otro dispositivo que ahorra trabajo es la tableta digitalizadora que facilita la creación de juegos de caracteres ya que el juego se construirá directamente en la superficie de la tableta.

La animación por juego de caracteres gráficos se implementó de la siguiente manera:
Se construyó un editor de caracteres y luego se creó con éste un hombrecillo y se animó de tal manera que pareciera que caminara.

A continuación se presenta el programa en BASIC, para la IBM PC que permite al usuario definir su propio juego de caracteres y después de este listado se muestra la forma en que se definieron los caracteres para el programa del hombrecillo caminante y el programa que produce la animación del mismo.

```
10 ' ** ESTE PROGRAMA ES UN EDITOR QUE PERMITE DEFINIR CARACTERES QUE SE
20 ' ** ADICIONARAN AL CODIGO ASCII Y CON LOS CUALES PODEMOS DEFINIR FIGU
30 ' ** RAS PARA ANIMACION POSTERIOR.
40 '
50 DIM A(7,7)
60 SCREEN 1
70 CLS
80 '
90 '
100 KEY OFF
110 KEY 1,"":KEY 2,""
120 KEY 3,"":KEY 4,""
130 KEY 10,CHR$(27)
140 '
150 '
160 ' ** SE POSICIONA EN MEMORIA RAM PARA GRABAR LOS CARACTERES ASCII
170 ' ** NUEVOS QUE SE DEFINAN
180 '
190 CBASE=&H4000
200 DEF SEG=0
210 POKE &H7C,0
220 POKE &H7D,&H40
230 POKE &H7E,PEEK(&H510)
240 POKE &H7F,PEEK(&H511)
250 DEF SEG
260 '
270 ACODE=128
280 '
290 ' ** ESCRIBE EL TITULO Y LA MATRIZ DE 8 X 8 SOBRE LA CUAL SE VA A EDITAR
300 '
310 LOCATE 1,5
320 PRINT "EDITOR DE CARACTERES"
330 FOR J=1 TO 8
340   LOCATE 4+J,18
350   PRINT ".......";
360 NEXT J
370 '
380 ' ** DESPLIEGA EL NUMERO DE CODIGO ASCII CON QUE SE TRABAJA
390 LOCATE 4, 1
400 PRINT "CODIGO ASCII :  ";
410 '
420 ' ** ESCRIBE LAS FUNCIONES QUE SE PUEDEN REALIZAR CON EL EDITOR
430 ' ** D PARA DIBUJAR, E PARA BORRAR ALGO PREVIAMENTE DEFINIDO, M PARA
440 ' ** MOVERSE SOBRE LA MATRIZ SIN AFECTARLA, C PARA BORRAR EL CONTENIDO
450 ' ** DE LA MATRIZ, L PARA DESPLEGAR UN CARACTER PREVIAMENTE DEFINIDO Y
460 ' ** S PARA GRABAR UNO NUEVO.
470 ' ** CON F1 F2 F3 Y F4 PUEDO REGRESAR O AVANZAR EN EL CODIGO ASCII.
480 '
490 LOCATE 1,30
500 PRINT "CURSOR"
```

```
500 PRINT "CURSOR"
510 LOCATE 2,30
520 PRINT "D DRAW"
530 LOCATE 3,30
540 PRINT "E ERASE"
550 LOCATE 4,30
560 PRINT "M MOVE"
570 LOCATE 6,1
580 PRINT "F1  -1    F2  +1";
590 LOCATE 7,1
600 PRINT "F3  -5    F4  +5";
610 LOCATE 6,30
620 PRINT "CARACTER";
630 LOCATE 7,30
640 PRINT "C CLEAR"
650 LOCATE 8,30
660 PRINT "L LOAD"
670 LOCATE 9,30
680 PRINT "S SAVE"
690 LOCATE 11,30
700 PRINT "F10 ESCAPE"
710 '
720 GOSUB 1770
730 '
740 '
750 GOTO 1930
760 '
770 '
780 ' ** CONTROLA EL PARPADEO DEL ASTERISCO QUE APARECE SOBRE LA MATRIZ
790 '
800 BLINK%=(BLINK%+1) MOD 20
810 IF BLINK%<10 THEN 890 ELSE 830
820 '
830 ' ** DESPLIEGA EL CARACTER # CUANDO SE ESTA DIBUJANDO LA FIGURA
840 '
850 IF A(ROW,COLUMN)=0 THEN CH$="."
860 IF A(ROW,COLUMN)=1 THEN CH$="#"
870 GOTO 940
880 '
890 ' ** MARCA ESOS SIMBOLOS PARA BORRAR, MOVER Y ADICIONAR
900 '
905 IF CURS=-1 THEN CH$="-"
910 IF CURS=0  THEN CH$="*"
920 IF CURS=1  THEN CH$="+"
930 '
940 LOCATE 5+ROW,18+COLUMN
950 PRINT CH$;
960 RETURN
970 '
980 '
990 IF A(ROW,COLUMN)=0 THEN CH$="."
1000 IF A(ROW,COLUMN)=1 THEN CH$="#"
```

```
1010 LOCATE 5+ROW,18+COLUMN
1020 PRINT CH$
1030 RETURN
1040 '
1050 '
1060 LOCATE 4,13
1070 PRINT USING "###";ACODE;
1080 CH=ACODE
1090 IF CH>6 AND CH<14 THEN CH=32
1100 LOCATE 10,10
1110 PRINT CHR$(CH);
1120 RETURN
1130 '
1140 '
1150 LOCATE 23,18
1160 PRINT "ESPERE"
1170 FOR I=0 TO 7
1180  LOCATE 5+I,18
1190  PRINT ".......";
1200  FOR J= 0 TO 7
1210  A(I,7-J)=0
1220  NEXT J
1230 NEXT I
1240 LOCATE 23,18
1250 PRINT "       "
1260 RETURN
1270 '
1280 ' ** GUARDA EL CARACTER CREADO EN LA MEMORIA RAM BAJO EL NUMERO DE
1290 ' ** CODIGO ASCII QUE SE LE INDICO.
1300 ' **
1310 IF ACODE<128 THEN 1460
1320 FOR I= 0 TO 7
1330  A0=0
1340  FOR J=0 TO 7
1350   A0=A0+A0+A(I,J)
1360  NEXT J
1370  POKE CBASE+8*(ACODE-128)+I,A0
1380 NEXT I
1390 I=INT(ACODE/32):J=ACODE MOD 32
1400 LOCATE 15+I,1+J
1410 PRINT CHR$(ACODE)
1420 LOCATE 23,18:PRINT"       ";
1430 RETURN
1440 '
1450 '
1460 LOCATE 23,1
1470 PRINT "NO PUEDE SALVAR ASCII MENOR DE 128";
1480 FOR I=1 TO 1000:NEXT I
1490 LOCATE 23,10
1500 PRINT"                                    ";
```

```
1510 RETURN
1520 '
1530 ' ** DESPLIEGA UN CARACTER CREADO CON ANTERIORIDAD Y LO LOCALIZA
1540 ' ** MEDIANTE EL NUMERO DE CODIGO ASCII QUE TIENE ASOCIADO
1550 ' **
1560 LOCATE 23,18
1570 PRINT "ESPERE"
1580 DEF SEG
1590 COFF = CBASE+8*(ACODE-128)
1600 IF ACODE >127 THEN 1630
1610 DEF SEG=&HFC00
1620 COFF=&HFA6E+8*ACODE
1630 FOR I= 0 TO 7
1640   A%=PEEK(COFF+I)
1650   FOR J=0 TO 7
1660   X%=A% AND 1
1670   A(I,7-J)=X%
1680   IF X% THEN X$="*" ELSE X$="."
1690   LOCATE 5+I,18+(7-J):PRINT X$
1700   A%=INT(A%/2)
1710   NEXT J
1720 NEXT I
1730 DEF SEG
1740 LOCATE 23,18:PRINT "       ";
1750 RETURN
1760 '
1770 ' ** DESPLIEGA TODO EL CONTENIDO DEL CODIGO ASCII
1780 ' ** INCLUYENDO LOS CARACTERES QUE SE HAN ESTADO CREANDO.
1790 '
1800 LOCATE 23,18:PRINT "ESPERE"
1810 FOR I=0 TO 7
1820  LOCATE 15+I,1
1830  FOR J=0 TO 31
1840    CH=32*I+J
1850    IF CH>6 AND CH<14 THEN CH=32
1860    PRINT CHR$(CH);
1870  NEXT J
1880 PRINT
1890 NEXT I
1900 LOCATE 23,18:PRINT "       ";
1910 RETURN
1920 '
1930 '
1940 '
1950 '
1960 ROW=0: COLUMN=0:CURS=0
1970 '
1980 '
1990 GOSUB 1050
2000 '
```

```
2010 '
2020 BLINK%=0
2030 IF CURS=-1 THEN A(ROW,COLUMN)=0
2040 IF CURS=+1 THEN A(ROW,COLUMN)=1
2050 '
2060 '
2070 GOSUB 770
2080 '
2090 A$=INKEY$
2100 DEF SEG: POKE 106,0
2110 IF LEN(A$)=0 THEN 2060
2120 IF LEN(A$)=1 THEN 2160
2130 IF LEN(A$)=2 THEN 2280
2140 GOTO 2060
2150 '
2160 ' ** DETECTA QUE FUNCION SE DESEA REALIZAR SOBRE LA MATRIZ
2170 '
2180 CODE1=ASC(A$) AND &H5F
2190 IF CODE1=27 THEN 3320
2200 IF CODE1=ASC("E") THEN 2990
2210 IF CODE1=ASC("M") THEN 3030
2220 IF CODE1=ASC("D") THEN 3070
2230 IF CODE1=ASC("C") THEN 3360
2240 IF CODE1=ASC("L") THEN 3400
2250 IF CODE1=ASC("S") THEN 3440
2260 GOTO 2060
2270 '
2280 IF ASC(A$)<>0 THEN 2010
2290 CODE2=ASC(RIGHT$(A$,1))
2300 GOSUB 980
2310 '
2320 '
2330 IF CODE2=71 THEN 2490
2340 IF CODE2=73 THEN 2580
2350 IF CODE2=79 THEN 2650
2360 IF CODE2=81 THEN 2720
2370 IF CODE2=72 THEN 2790
2380 IF CODE2=75 THEN 2840
2390 IF CODE2=80 THEN 2940
2400 IF CODE2=77 THEN 2890
2410 '
2420 '
2430 IF CODE2=60 THEN 3180
2440 IF CODE2=61 THEN 3230
2450 IF CODE2=59 THEN 3110
2460 IF CODE2=62 THEN 3270
2470 GOTO 2060
2480 '
2490 ' ** CONTROLAN EL AVANCE DEL CURSOR SOBRE LA MATRIZ Y PERMITEN
2500 ' ** QUE ESTE SOLO PUDA MOVERSE SOBRE ESTA SIN SALIRSE DE ELLA
2510 '
2520 IF ROW=0 THEN ROW=8
2530 IF COLUMN=0 THEN COLUMN=8
2540 ROW=ROW-1
2550 COLUMN=COLUMN-1
2560 GOTO 2010
2570 '
2580 '
2590 IF ROW=0 THEN ROW=8
2600 IF COLUMN=-7 THEN COLUMN=-1
```

```
2610 ROW=ROW-1
2620 COLUMN=COLUMN+1
2630 GOTO 2010
2640 '
2650 '
2660 IF ROW=7 THEN ROW=-1
2670 IF COLUMN=0 THEN COLUMN=8
2680 ROW=ROW+1
2690 COLUMN=COLUMN-1
2700 GOTO 2010
2710 '
2720 '
2730 IF ROW=7 THEN ROW=-1
2740 IF COLUMN=7 THEN COLUMN=-1
2750 ROW=ROW+1
2760 COLUMN=COLUMN+1
2770 GOTO 2010
2780 '
2790 '
2800 IF ROW=0 THEN ROW=8
2810 ROW=ROW-1
2820 GOTO 2010
2830 '
2840 '
2850 IF COLUMN=0 THEN COLUMN=8
2860 COLUMN=COLUMN-1
2870 GOTO 2010
2880 '
2890 '
2900 IF COLUMN=7 THEN COLUMN=-1
2910 COLUMN=COLUMN+1
2920 GOTO 2010
2930 '
2940 '
2950 IF ROW=7 THEN ROW=-1
2960 ROW=ROW+1
2970 GOTO 2010
2980 '
2990 '
3000 CURS=-1
3010 GOTO 2010
3020 '
3030 '
3040 CURS=0
3050 GOTO 2010
3060 '
3070 '
3080 CURS=+1
3090 GOTO 2010
3100 '
```

```
3110 ' ** DETECTAN CUANTO SE DESEA REFERENCIAR UN CODIGO ASCII ANTERIOR
3120 ' ** O POSTERIOR AL ACTUAL MEDIANTE LAS TECLAS DE FUNCTION.
3130 '
3140 IF ACODE=0 THEN 3160
3150 ACODE=ACODE-1
3160 GOTO 1980
3170 '
3180 '
3190 IF ACODE=255 THEN 3210
3200 ACODE=ACODE+1
3210 GOTO 1980
3220 '
3230 '
3240 IF ACODE<5 THEN 3260
3250 ACODE =ACODE-5
3260 GOTO 1980
3270 '
3280 IF ACODE>250 THEN 3300
3290 ACODE=ACODE+5
3300 GOTO 1980
3310 '
3320 '
3330 LOCATE 23,1
3340 GOTO 3480
3350 '
3360 '
3370 GOSUB 1140
3380 GOTO 1920
3390 '
3400 '
3410 GOSUB 1520
3420 GOTO 1920
3430 '
3440 '
3450 GOSUB 1280
3460 GOTO 1980
3470 '
3480 END
```

```
10 ' ** ESTE PROGRAMA SIMULA LA CAMINATA DE UN HOMBRE POR LA PANTALLA
20 ' ** LA FIGURA SE ENCUENTRA DEFINIDA POR MEDIO DE LOS CARACTERES
30 ' ** DEL CODIGO ASCII QUE SE EDITARON EN EL PROGRAMA ANTERIOR
40 CLS
50 '** DEFINE EL ESPACIO EN MEMORIA EN EL CUAL SERAN CARGADOS LOS
60 '** CARACTERES DEL CODIGO ASCII QUE SE DEFINIERON Y LOS TRAE DE
70 '** UN ARCHIVO LLAMADO WALKCHAR, EL CUAL CARGA EN ESAS LOCALIDADES
80 '** DE MEMORIA.
90 '
100 CLEAR, &H8000
110 DEF SEG=0
120 POKE &H7C,0
130 POKE &H7D,&H40
140 POKE &H7E,PEEK(&H510)
150 POKE &H7F,PEEK(&H511)
160 DEF SEG
170 BLOAD "B:WALKCHAR",&H8000
180 ' ** PONE RESOLUCION MEDIA A LA PANTALLA
190 '
200 SCREEN 1
210 ' **
220 '** REALIZA LA CAMINA DEL HOMBRE, LLAMANDO A LA SUBRUTINA QUE
230 '** DESPLIEGA LOS CARACTERES QUE LO FORMAN .
240 '
250 FOR R = 1 TO 19 STEP 3
260 WHILE B=1
270 FOR F=0 TO 4
280 GOSUB 340
290 NEXT F
300 WEND
310 B=1
320 NEXT R
321 SOUND 150,10
322 SOUND 200,10
323 LOCATE 12,15
325 PRINT " A D I O S"
326 END
330 ' ** LA FIGURA DEL HOMBRE QUE CAMINA
340 K=K+1
350 IF K= 40 THEN K=1: B=0
```

```
360 G=128+6*F
370 LOCATE R,K
380 PRINT CHR$(G+0);CHR$(G+1)
390 LOCATE R+1,K
400 PRINT CHR$(G+2);CHR$(G+3)
410 LOCATE R+2,K
420 PRINT CHR$(G+4);CHR$(G+5)
430 FOR I=1 TO 20:NEXT
440 LOCATE R,K : PRINT " ";" "
450 LOCATE R+1,K : PRINT " ";" "
460 LOCATE R+2,K : PRINT " ";" "
470 RETURN
```

4.3 ANIMACION POR IMAGENES

La animación por imágenes produce los efectos de movimiento de manera similar a la animacion por caracteres. La mayor diferencia entre los dos métodos consiste en el modo en que las celdas individuales se crean y se muestran.

Las celdas de la animación por imágenes se crean mediante puntos individuales especificados por sus coordenadas de localización. Mediante el uso de coordenadas se pueden localizar puntos en un sistema que se origina en la parte superior izquierda de la pantalla . En resolución media el eje de las X tiene un rango de 0 a 319.
Las coordenadas verticales se miden en el eje de las Y y tienen un rango de 0 a 199.

Las celdas de animación pueden colocarse en la pantalla con el comando PUT en las coordenadas adecuadas siempre y cuando la imagen no exceda los lados de la pantalla. Con la animación por imágenes se tiene la posibilidad de reubicar nuevas celdas a unos cuantos puntos de las celdas desplegadas previamente permitiendo una animación suave, con un amplio rango de velocidades.

Las imágenes pueden guardarse en memoria con el comando GET.
El comando PUT restaura imágenes en la pantalla leyendo información numérica del arreglo.

Para cambiar la imagen puede usarse uno de los siguientes comandos: PSET, AND,PRESET, XOR, o, OR de los cuales solamente PSET y XOR pueden crear imagenes desplegadas previamente en una secuencia, lo cual es necesario para producir animación. Por esta razon solamente se presentan las instrucciones PSET y XOR.

4.3.1 MOVIMIENTO MEDIANTE PSET

Las imagenes PSET se despliegan con un fondo de color y un borde en la orilla de la figura. Este borde debe ser tan ancho como el movimiento mas largo de los puntos de la imagen. Cuando la imagen se despliega con PUT en una localidad diferente, la imagen anterior se cubre con la nueva y el borde alrededor de ella . El borde debe de ser suficientemente ancho para que la imagen previa se cubra, a pesar de que la ubicación de la nueva imagen cambie. Si la nueva imagen cambia una distancia mas grande que el ancho del borde, parte de la imagen anterior permanece visible en la pantalla.

Consecuentemente se debe considerar el fondo cuando se diseñe una secuencia de animación que use imágenes PSET.

Las imagenes PSET tienen las siguientes ventajas :

Solamente se necesita un comando PUT para mover una imágen PSET a una nueva posición lo cual permite una gran velocidad en las secuencias.

Las imágenes PSET se borran y se muestran al mismo tiempo. De esta forma no hay parpadeo.

Las imágenes PSET retienen sus colores verdaderos sin importar el color del fondo.

Las imágenes PSET reemplazan el fondo sobre el cual se despliegan.

La velocidad máxima de la animación PSET está limitada por el ancho del borde que se encuentra alrededor de la imágen.

Las imágenes PSET muestran su propio borde coloreado, sobre el fondo actual. Esto puede ser útil o limitante dependiendo de la aplicación y diseño de la secuencia de animación.

A continuación se presenta un programa de movimiento PSET que muestra una pelota de color sobre un fondo multicolor.

```
10    ' ANIMACION POR IMAGENES
20    '
30    '
40    '
50    ' INICIALIZACION DEL SISTEMA
100   '
110 KEY OFF
120 SCREEN 1,0: COLOR 0,1: CLS
130 DEFINT A-Z
140 '
200 ' INCIALIZACION DEL PROGRAMA
210 XLOC= 12:     YLOC=5
220 XCHNG= 2:     YCHNG=3
230 DIM BALL(34)
240 '
500 ' DIBUJA Y TRAE UNA IMAGEN DE UN CIRCULO RELLENO
510 CIRCLE (100,100), 5, 2: PAINT STEP (0,0), 2,2
520 GET (100-8,100-8)-(100+7,100+7), BALL
530 CLS
540 '
700 ' DIBUJADO DEL FONDO
720 LINE (I*80,0)-(I*80+79,199), I, BF
730 NEXT I
740 LOCATE 2,14: PRINT "MOVIMIENTO PSET";
750 '
1000 'CICLO DE ANIMACION
```

```
1010 PUT (XLOC, YLOC), BALL, PSET
1020 XLOC=XLOC+ XCHNG:  YLOC=YLOC+YCHNG
1030 GOSUB 2010
1040 FOR PSE=1 TO 50: NEXT PSE
1050 '
1060 GOTO 1010
1070 '
2000 'SUBRUTINA QUE REVISA QUE EL RENGLON Y COLUMNA SEAN
2005 ' POSICIONES VALIDAS
2010 IF XLOC<0 THEN XLOC=0: XCHNG=-XCHNG
2020 IF XLOC>319-15 THEN XLOC=319-15: XCHNG=-XCHNG
2030 IF YLOC<0 THEN YLOC=0: YCHNG=-YCHNG
2040 IF YLOC>199-15 THEN YLOC=199-15: YCHNG=-YCHNG
2050 RETURN
```

4.3.2 MOVIMIENTO MEDIANTE XOR

Las imágenes XOR usan un proceso diferente de movimiento y animación que las imágenes PSET y a diferencia de éstas,no destruyen los fondos sobre los cuales se despliegan. El método XOR de despliegue crea un compuesto de la imágen y del fondo mencionado, asi que la imágen compuesta cambia a un color complementario del fondo sobre el cual se despliega.
En la animación XOR antes de que la imágen se mueva a otro lugar, la imágen anterior debe borrarse mediante el uso del comando PUT, el cual borra la imagen actual y devuelve a la pantalla su fondo original.

La animación XOR tiene las siguientes ventajas :

-Las imágenes XOR preservan el fondo sobre el cual viajan.

-Las imagenes XOR se pueden mover a cualquier distancia y a cualquier velocidad ya que no están restringidas a moverse menos que el ancho de sus bordes.

El movimiento XOR tiene las siguientes desventajas :

-Las imagenes XOR deben desplegarse dos veces con el comando PUT. El primer PUT es el desplegado inicial de la imagen, el segundo PUT borra la imágen y regresa el fondo original. Como resultado la velocidad de animación es aproximadamente lo doble de lenta que en el movimiento PSET.

-Las imágenes XOR desplegadas sobre cualesquiera de los colores excepto negro tiene una vista transparente ya que las formas y los colores del fondo se pueden ver a través de ella.

-La animación XOR parpadea porque la imágen desaparece de la pantalla durante el ciclo de borrado. Si se aumenta la cantidad y el

tamaño de las imagenes se aumenta también el parpadeo.

4.3.3 RECOMENDACIONES PARA ANIMACION POR IMAGENES

Existen ventajas y limitaciones cuando se usa animación de imagenes para mover las figuras:

-Las imágenes pueden ser de cualquier medida.
-Las imágenes pueden ser multicolores y contener detalles tan finos como un punto
-Las imágenes pueden moverse en incrementos de un punto propiciando movimientos suaves sobre un gran rango de velocidades
-El número de imágenes que pueden crearse está limitado sólo por la medida del arreglo de imágenes y por el monto de la capacidad de almacenamiento
-Las imágenes producidas por PSET son rápidas y de alta calidad.
-Las imágenes XOR conservan los fondos sobre los cuales viajan
-La animación XOR parpadea y es mas lenta que PSET
-Las imágenes XOR cambian de color cuando pasan sobre fondos coloreados
-La velocidad de las imágenes PSET está limitada por el ancho del borde de la celda
-Las imágenes PSET dejan un rastro de color, por esta razón la animación PSET se limita a fondos de colores sólidos.

4.4 ANIMACION POR REGISTROS DE COLOR

Los registros de color proporcionan una posibilidad barata de utilizar una gran variedad de colores. Si se desea que el artista meneje 16 millones de colores una técnica muy costosa sería el que estuviera posibilitado para hacer que cada punto en la pantalla mostrara diferentes colores de tal forma que los 16 millones de colores pudieran utilizarse. Cada "pixel" debe entonces de contener 24 bits de información para poder presentar cualquiera de los 16 millones de colores.

Con el uso de registros de color para tener información acerca del color de cada "pixel" se guarda un byte que apunta a una tabla de colores con 16 bits por "pixel". La tabla contendrá las 256 diferentes descripciones de los 16 millones de colores

Si el valor del "pixel" es 43, la computadora buscará el índice 43 de la tabla . El valor del color que está contenido en esta posición de la tabla se muestra entonces en ese "pixel". La memoria de pantalla tendrá cerca de 1 millón de bytes de los cuales sólo 768 se utilizan para la tabla. Esto representa una inversión de aproximadamente una tercera parte del precio de otros mètodos.

Mediante este sistema el artista solamente podrá mostrar en la pantalla 256 colores al mismo tiempo.

Si se planea una pintura cuidadosamente pueden crearse escenas impresionantes con mucho menos de 256 colores.

Esta técnica de despliegue de colores se llama mapeo de color y la tabla de colores se llama mapa de color.

Mapas de colores y la ATARI PC

La ATARI PC es una de las pocas computadoras personales que usan esta técnica para desplegar colores en la pantalla, sin embargo solo hay 128 colores posibles a escoger y solamente 9 entradas en el mapa de color. Estas 9 entradas se llaman registros de color.

La mayoría de los modos gráficos de la ATARI PC no usan los 9 registros de color, muchos utilizan solamente 4 o menos.

4.4.1 CREACION DE MOVIMIENTO CON REGISTROS DE COLOR

Supongamos que se tienen 9 cubetas de pintura numeradas del 0 al 8, cada una de ellas tiene un color diferente. Se usará también una charola de pintura temporal llamada TEMP. Se van a utilizar

las 9 cubetas y la charola para jugar al "pase de colores".

Primer paso	segundo paso	
cubeta 0 a la charola	cubeta 1 a la cubeta 0	etc...

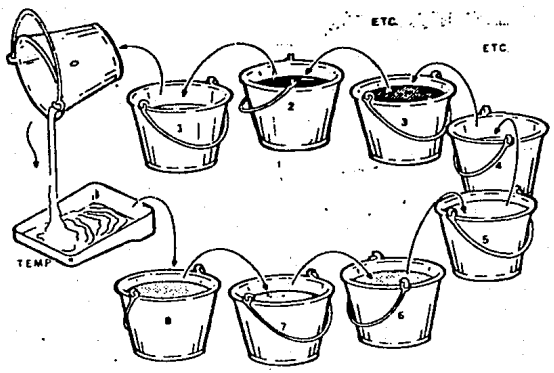


Fig. 4.2 Creación de movimiento con registros de color

Primero se vacia la pintura contenida en la cubeta 0 a la charola temporal después el contenido de la cubeta número 1 a la cubeta número 0 que ahora se encuentra vacia. A continuación vacie el contenido de la cubeta 2 a la cubeta 1 y asi sucesivamente hasta que la cubeta 8 se vacie a la cubeta 7. Ahora no hay mas cubetas con que llenar la cubeta 8, pero en TEMP aun se tiene la pintura original de la cubeta 0, asi que se toma esta pintura y se vacia en la cubeta 8. Ahora se incia nuevamente desde el paso número 1 creando un ciclo interminable de movimiento de colores, lo cual se veria en la pantalla como un patrón que cambia rápidamente.

Mediante el uso de esta técnica pueden crearse efectos especiales de animación en color como: una esfera girando con varios colores, una persiana de color en movimiento, que al desplazarse sobre la pantalla cada uno de sus componentes cambia constantemente de color. También se utiliza para la construcción de gráficas de barras , pie, etc..., en varios colores.

4.5 ANIMACION DE FONDOS

En la animación actual los fondos usualmente cuentan con un sólo nivel el cual se mueve atrás de los caracteres a medida que estos caminan o se mueven sobre de el. Esto crea la ilusión de movimiento, en un mundo bidimensional.

Los fondos son importantes por dos razones, un fondo bien diseñado resalta y complementa la acción en una secuencia de animación. En segundo lugar todo el fondo pueden animarse.

4.5.1 DISEÑO DE FONDOS

Para el diseño de fondos se deben de tomar en cuenta tres elementos: color, balance y líneas.

Color

En la creación de fondos las opciones de color afectan el balance, la composición y en general la calidad de la imágen. Cuando las figuras se combinan con los fondos de animación el color puede enfatizar y remarcar la silueta de la imágen.

Balance

El balance es una sensación de armonía en la imágen, la cual se produce por la distribución adecuada de los componentes de masa, color y líneas. Un fondo balanceado crea un foco visual de la acción central y mejora la apariencia de la pantalla.

Lineas

Para evitar la distracción visual, la mayoría de las líneas o rutas de movimiento en cualquier escena, deben de ser la clave de la imagen o acción. Diagonales, perspectivas u otros tipos de líneas se utilizarán para crear figuras, fondos o efectos tridimensionales.

4.5.2 ANIMACION POR CAMBIO DE PALETAS

Una paleta es una selección de colores disponibles en la microcomputadora. La paleta de la IBM PC tiene 16 colores pero no todos están disponibles en cualquier resolución. En modo SCREEN 1, la IBM PC tiene dos paletas cada una de las cuales contiene dos grupos de tres colores para las figuras superpuestas al fondo de la pantalla. El rango completo de 16 colores sólo está disponible en modo SCREEN 1 para colores de fondo, es por esta limitación que la IBM PC no puede producir animación por cambio

de paletas.
La IBM PC JR. tiene dos paletas, la primera de éstas se denomina paleta física la cual contiene 16 colores numerados del 0 al 15. La segunda paleta puede diseñarse utilizando los colores de la paleta anterior.
Esta segunda paleta tiene una propiedad especial, cuando se dibuja en la pantalla, la computadora recuerda la localidad de la paleta conocida como atributo, si se cambia el color del atributo, también cambia el color en la pantalla. Los colores se diferencian mediante su número de atributo como se muestra en la figura:

La paleta física
COLORES ORGINALES DE PALETA Y SUS ATRIBUTOS

Atributo	Color	Atributo	Color
0	Negro	8	Gray
1	Azul	9	Azul brillante
2	Verde	10	Verde brillante
3	Cyan	11	Cyan brillante
4	Rojo	12	Rojo brillante
5	Magenta	13	Magenta brillante
6	Cafe	14	Amarillo
7	Blanco	15	Blanco de alta intensidad

Cuando los colores en la paleta se cambian rapidamente para las imágenes de la pantalla, se crea el efecto de movimiento.
Con el cambio de paleta todos los objetos que se van a animar se despliegan simultáneamente a pesar de que algunos son invisibles. Un objeto se vuelve invisible cuando se le asigna el atributo del color del fondo. Si se asigna un color diferente, el objeto se vuelve visible. Cuando el elemento visible necesita aparecer en la secuencia de animación, el atributo del color se cambia a uno que contraste con el fondo.

Cuando este proceso de cambio de color es muy rápido los objetos en el fondo parece que se mueven aunque en realidad están estáticos y sólo los colores cambian. Ya que la animación por cambio de paleta se efectua sin cambio en la memoria de pantalla se produce una animación rápida.

4.5.3 ANIMACION POR MOVIMIENTO DE PAGINAS EN LA PANTALLA

Este tipo de animación es similar a la animación por imágenes desde el punto de vista que ambas usan un cambio en la secuencia

de las imágenes para simular el movimiento. En lugar de usar secuencias de figuras individuales como en la animación por imágenes, este tipo de animación utiliza secuencias de pantallas completas (páginas).
Las páginas son areas de memoria reservadas que congelan la imágen de una pantalla completa, la cual puede contener fondos, gráficas o textos. El monto de memoria que utiliza una página depende del modo SCREEN seleccionado. El número de páginas que pueden usarse es igual a la medida de la memoria de pantalla dividida por la medida requerida para una página.

Esta técnica es la mejor para animar fondos que requieren movimiento repetitivo.

Algunas aplicaciones de esta técnica son: escenas de fondo como arboles movidos por el viento de lado a lado, lluvia que cae o simulación de una linea de ensamblaje.

Para crear este tipo de animación las escenas de fondo deben dibujarse una en cada página. Para hacer que un objeto de fondo mude su posición, se debe cambiar un poco de una página a otra. Los cambios muy grandes hacen que el movimiento aparezca rudimentario.

Es necesario seleccionar cada página antes de dibujar. Las páginas están numeradas empezando con el 0. La página activa es la que se está dibujando actualmente, la página visible es la que se está viendo en este momento.

Secuencias del Despliegue

Existen tres caminos para mover las páginas en la pantalla: "round-robin, reciprocating y ad hoc". Las secuencias "round robin" son las mejores para el movimiento continuo en un ciclo repetitivo. Una secuencia "round-robin" es:

0 a 1 a 2 a 3 a 0 a 1 a 2 a 3 a 0 a 1 a 2 a 3 .....

Las secuencias "reciprocating" trabajan bien produciendo movimientos de un lado a otro como el movimiento de un pistón o el de una ola marítima. La secuencia "reciprocating" es como sigue:

0 a 1 a 2 a 3 a 2 a 1 a 0 a .1 a 2 a 3 a 2 a 1 a 0 .....

Las secuencias "Ad hoc" despliegan las páginas en el orden y frecuencia apropiados para el movimiento y los efectos deseados por el programador.

Para ejemplificar se muestra el siguiente programa:

```
10    '
20    ' PARA IBM PC
22    '
25    '
30    '
40    ' INICIALIZACION DEL SISTEMA
100   '
120 SCREEN 0,1: WIDTH 40: KEY OFF
130 LOCATE,,0 'DESACTIVA EL CURSOR
140 DEFINT A-Z
150 COLOR 0,7,7  'ESTABLECE EL COLOR BLANCO PARA CLS
160'
500   'INICIALIZACION DEL PROGRAMA
510 DIR =1
520 PAGE= 9-1
530   '
700   'IMPRIME EL TITULO
710 ANM$="ANIMACION MAGICA"
720 FOR I = 0 TO PAGE
730     SCREEN,,I: CLS
740     COLOR 6,7,7: LOCATE I+2, I+2: PRINT ANM$;
750     COLOR 1: LOCATE I+2, 24-I: PRINT ANM$;
760     COLOR 2: LOCATE 24-I, 13: PRINT ANM$;
770     COLOR 3: LOCATE 13, 2+I: PRINT ANM$
780     COLOR 4: LOCATE 13, 24-I: PRINT ANM$;
800 NEXT I
810   '
890   'CICLO DE ANIMACION
1000  '
1010 BEEP
1020 I=0
1030 SCREEN,,,I
1040 FOR PSE=1 TO 150: NEXT PSE
1050 I=I+DIR
1060 IF I=PAGE THEN DIR=-1: ELSE IF I=0 THEN DIR=0
1070 GOTO 1030
```

4.5.4 RECOMENDACIONES PARA LA ANIMACION DE FONDOS

Hay varias ventajas y limitaciones que se deben de tener en cuenta cuando se escriben programas para animación de fondos:

-El cambio de paleta es útil para animar objetos que se mueven en ciclos repetitivos, sin embargo los fondos que tienen muchos colores usan muchos atributos lo cual reduce el número de atributos que no cambiarán.

-El cambio de paletas puede usarse para hacer sobresalir

información importante en los títulos de las páginas.

-El cambio de paletas puede combinarse con el movimiento de páginas en la pantalla y con la animación de figuras. Esto es porque es rápido y no utiliza memoria, sin embargo el uso de fondos complejos y figuras con muchos colores requiere una gran planeación para asegurar la animación adecuada.

-La animación por movimiento de páginas en la pantalla puede crear fondos complejos de animación que utilicen un número ilimitado de figuras, colores y objetos en movimiento, pero esto sólo puede realizarse en la IBM PC JR. con al menos 128K.

4.6 ANIMACION EN TIEMPO REAL EN SISTEMAS BASADOS EN EL 8086/8088

Intel cuenta con los siguientes procesadores:

Unidad de procesamiento (CPU) 8086
El coprocesador de datos 8087
El procesador de entrada/salida 8089

El 8086 se encarga de controlar todo el sistema distribuyendo tareas a los otros procesadores. Los cálculos numéricos de alta precisión se realizan con el procesador 8087 y el procesador de entrada/salida 8089 es el encargado de los movimientos de bloques dentro de la memoria entre el CRT, la memoria principal y las memorias de visualización del video.

Si se piensa en una resolución estandar de 480 x 512 pixels se tienen 245,760 pixels en toda la pantalla donde cada pixel necesita 3 bits y por lo tanto se necesitan 245,760 x 3=737,280 bits. Puesto que cada byte tiene 8 bits esto nos da 92,160 o sea aproximadamente 94 KB necesarios por dibujo, o sea mas de los 64KB que el 8086 o 8088 pueden direccionar directamente.

Si se quiere producir animación, se necesita guardar en memoria cuatro o cinco figuras simultáneamente, por ejemplo, dibujar un cierto animal "ensamblando" figuras separadas de una cabeza, patas y rabo. Por otro lado, también se necesita mantener varios dibujos en memoria para pasar al anterior, o al siguiente (o a partes de ambos) y producir la impresión de movimiento.

Para realizar animación se necesita cambiar escenas a una razón de 10 imágenes por segundo.

Mientras se está visualizando una escena, siempre hay otra que se está dibujando. Cuando la segunda escena se acaba de dibujar, se despliega y la primera desaparece de la pantalla y queda preparada para ser utilizada como tercer escena.

La siguiente escena debe aparecer "instantaneamente" en el lugar de la imágen anterior. Este efecto es dificil (y caro) de conseguir si el dibujo se calcula durante el despliegue o si se "lleva" a su sitio moviéndolo de una posición en memoria al area de visualización.
El 8089 no puede transferir 92KB con tanta rapidez.
Con el bus de datos de 16 bits y una frecuencia de 5 Mhz, el 8089 necesita del orden de 0,075 segundos para la transferencia. Para conseguir un cambio de imágen instantáneo, la transferencia deberá realizarse entre dos refrescamientos, esto es, durante lo que recibe el nombre de re-trazado vertical.

El 8089 tiene una parte importante en el movimiento de partes de

un dibujo o de una figura entera.

4.6.1 UNA APLICACION EN GRAFICAS TRIDIMENSIONALES EN TIEMPO REAL

Tipicamente la trigonometría y las matemáticas de punto flotante de representación tridimensional llevan a resultados que dejan mucho que desear. La velocidad a la que las perspectivas se generan es insuficiente para simular el tiempo real. El sistema NEC video sintetizador de tres dimensiones (3 VSD) intenta a resolver este problema mediante la eliminación de funciones trigonométricas, escribiendo el algoritmo resultante en código ensamblador 8086 y usando solamente aritmética integrada de 16 bit.

El 3 VSD está diseñado para trabajar con el sistema operativo CPM/86 DOS de Digital Research. El código requiere menos de 3 K Bytes de memoria y debe cargarse lo mas arriba posible en la memoria o bien dentro de ROM.

Los requerimientos de datos del RAM dependen del tamaño de la base de datos tridimensional, la cual puede grabarse y cargarse del disco . El sistema también incluye un controlador gráfico de pantalla NEC PD7220 igual al que se encuentra en la computadora NEC PC . Una segunda versión del 3 VSD puede manejarse en el procesador MOTOROLA 6845 como el de la computadora IBM PC. El 3 VSD incluye el archivo Graphic A86, el cual provee el código para manejar el dispositivo gráfico de pantalla.

El dispositivo gráfico de pantalla necesita ser capaz de manejar las rutinas INIT,CLEAR, SWAP, y PLOT . La rutina de PLOT puede ser tan simple como una rutina de posición de cursor (de 80 caracteres por 24 líneas) o puede ser implementada a través de un paquete de gráficas estandar como el de Digital Research GSX .

Al observar un objeto tridimensional, el ángulo de visión forma un conjunto de vectores que comienzan dentro del ojo y apuntan hacia las partes visibles del objeto.

El modo usual de calcular esta perspectiva es encontrando la intersección de esas líneas de vista poniendo un plano de proyección entre los ojos y el objeto. Se puede obtener una perspectiva mas natural encontrando la intersección de las líneas de visión con una esfera centrada en el ojo. De esta forma un objeto es igualmente proporcional en toda la pantalla no importa donde esté físicamente localizado en la misma. Ambos tipos de perspectiva y también la proyección isométrica se incorporan dentro del 3 VSD.

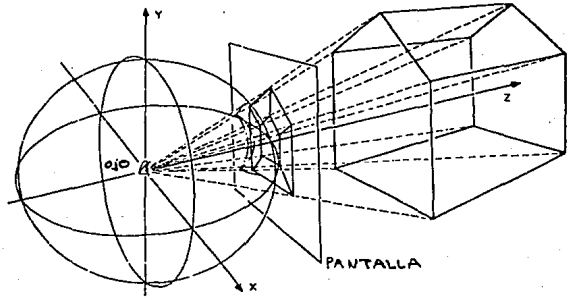


Fig. 4.3 Las perspectivas dependen de la intersección de un vector con la superficie a cierta distancia del ojo. Ya que la esfera es curva, presenta una vista mas natural.

Solución matemática

El espacio del objeto (3 DAE), debe contener tres números por cada punto que representa. Esos números o coordenadas especifican la localización del punto dentro del espacio del objeto. Podria haber otros parámetros asociados con el punto que indiquen su color o bien si ese u otros puntos forman una línea segmentada etc... Por ahora se consideran solamente tres coordenadas.

Supongamos que una cámara u ojo humano está situada en el origen (0,0,0) del 3 DAE y está orientada de tal forma en que el ojo está mirando hacia abajo a uno de los tres ejes mas importantes. Designaremos este como el eje z. Inmediatamente hacia arriba se encuentra el eje y-positivo y en una dirección horizontal hacia la derecha el eje x-positivo. Esto lleva a una perspectiva bidimensional que está orientada en el plano x-y. La línea de visión a un punto en el 3 DAE es justamente el vector o rayo (x,y,z).
Si se coloca la pantalla plana imaginaria lejos de la cámara sobre el eje z positivo el punto de intersección con el vector (x,y,z), será el punto con su tercera coordenada igual a 1, (x/z, y/z, 1) . Igualmente en una perspectiva real la intersección de este vector con la esfera cerca del ojo será (x/q, y/q, z/q) y la distancia al origen es 1.

Si se considera que la cámara está en una posición y dirección arbitraria, el origen puede moverse a la posición de la cámara mediante una transformación. Durante una transformación se aumentan las mismas tres constantes a cada punto en el 3 DAE, lo cual

conserva relativamente sin cambio las posiciones de los puntos. Estas tres constantes deben ser (-a,-b,-c) en donde (a,b,c) es la posición de la cámara.

Una vez que la cámara está en una posición usual (0,0,0) su orientación puede ajustarse mediante rotación. Cualquier rotación sobre el origen puede dividirse en tres rotaciones una de ellas alrededor de cada uno de los ejes mayores. Una rotación alrededor de un eje mayor es una rotación bidimensional. Por ejemplo el 3 DAE completo puede ser rotado 30 grados sobre el eje "y" de la siguiente manera :

x' = xCos30 - zSin30
y' = Y
z' = xSin30 + zCos30

en donde (x,y,z) es un punto en la 3 DAE.

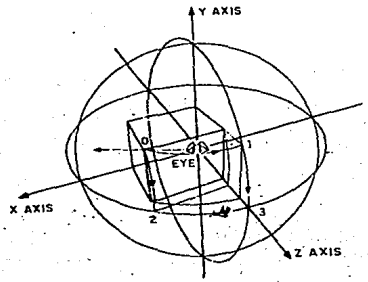


Fig. 4.4 El vector representado por la linea punteda puede alinearse con eje z positivo de dos maneras. La multiplicación de matrices no-conmutativa asegura que los dos casos requieren diferente monto de alineamiento y rotación.

El vector que atravieza el punto 0, puede alinearse con el eje z positivo mediante la rotación de los ejes x y y .

Si se pone (a,b,c) en la misma posición de la cámara (Y,P,R) a igual desviación, inclinación y giro, y (x,y,z) es igual a un punto arbitrario en el universo del programa , se puede obtener (i,j) , una posición de un punto en perspectiva mediante la

multiplicación de matrices.

```
q     =     longitud de el vector (i,j,k)
      =     raiz cuadrada (i^2 + j^2 +k^2)
      =     raiz cuadrada de ((x-a)^2 +(y-b)^2 +(z-c)^2)
```

$$\begin{pmatrix} I \\ J \\ K \end{pmatrix} = \begin{pmatrix} CosR & -SinR & 0 \\ SinR & CosR & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & CosP & -SinP \\ 0 & SinP & CosP \end{pmatrix} \begin{pmatrix} CosY & 0 & -SinY \\ 0 & 1 & 0 \\ SinY & 0 & CosY \end{pmatrix} \begin{pmatrix} x-a \\ y-b \\ z-c \end{pmatrix}$$

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 1/q & 0 & 0 \\ 0 & 1/q & 0 \end{pmatrix} \begin{pmatrix} I \\ J \\ K \end{pmatrix}$$

Fig. 4.5 Ya que las tres matrices de 3 x 3 pueden aplicarse a cualquier punto (x,y,z) en el 3DAE, se tiene un algoritmo rápido si primero se multiplican juntas en una matriz de transformación simple de 3 x 3.

En vista de que las tres matrices de 3 x 3 en la figura deben ser aplicadas a cada punto (x,y,z) en el 3DAE se puede obtener un algoritmo mas rápido si primero se multiplican por una matriz de orientación y transformación de 3 x 3 (MOT). Se proveen los valores (Y,P,R) y también a, b y c (la posición de la cámara) en tiempo real. Esos valores pueden ser absolutos o relativos a una orientación y posición actuales.

Si se guarda la posición actual (a,b,c) entonces dado un cambio (aa,bb,cc) en (a,b,c) puede usarse (a+aa, b+bb, c+cc) como la siguiente posición absoluta de la cámara.

Solución por microcomputadora

Habiendo reducido la MOT a una forma de dos puntos la única operación matemática aparte de la suma , resta, división y multiplicación es la función de raiz cuadrada. Esta puede calcularse en términos de otras cuatro operaciones por aproximación

sucesiva.

La forma en que una microcomputadora realiza la aritmética con números reales se llama aritmética de punto flotante.

Cada punto en el 3DAE consistirá de cuatro palabras de memoria de 16 bits. Las primeras tres palabras son las tres coordenadas absolutas (x,y,z). La cuarta palabra se interpreta como cuatro nibbles (dígitos hexadecimales). El nibble mas significativo o de mayor orden indica la dimensión de la gráfica primitiva. Un 0 es un simple punto, un 1 es un segmento de línea etc... los otros tres nibbles indican los patrones de colores que serán usados.

El programa 3 VSD es un ciclo dentro de un ciclo . El ciclo mas largo, CAMARA calcula la posición de la cámara o los cambios de orientación . El ciclo mas pequeño PERSP debe de ser lo mas rápido posible ya que se ejecuta una vez por cada punto del universo del programa. PERSP, en turno, llama a la rutina PLOT GDC la cual dibuja en la pantalla los puntos, los segmentos de línea etc....

Cuando se dibujan perspectivas se debe rotar el componente z tal y como está especificado por el MOT, antes de que se pueda determinar la intersección de la línea de visión y la pantalla plana. El componente z (H) es usado para otros propósitos, permite medir cuando un objeto están enfrente (H > 0) o atrás (H<0).

El tiempo necesario para grabar el 3 DAE y para procesar el PERSP puede ser reducido mediante el uso de primitivos gráficos altamente dimensionables . A un segmento de línea se le puede codificar como un par de primitivos gráficos 0-dimensionales. Por ejemplo sus puntos finales. La cara de un poliedro puede codificarse como un juego de segmentos de línea.

Dos problemas que son triviales cuando se manejan primitivos gráficos 0-dimensionales se vuelven mas difíciles en un universo de mas dimensiones. Los primitivos gráficos se recortan y se les eliminan las líneas ocultas.

Si se están usando dos areas de refrescamiento se debe borrar el dibujo que se trazó inmediatamente antes y no el que se acaba de dibujar. En cualquier caso se deben de guardar los puntos conforme se van dibujando.

Si se añade otro conjunto de programas RAM de 128 Kbyes al sistema, el 3DAE puede volverse considerablemente complejo y la velocidad del 3VSD disminuirá. Si se está dispuesto a sacrificar control de tiempo real, los cuadros individuales pueden guardarse en videotape y mostrarse tan rápidamente como se desee.

Esta primera versión de 3VSD fue diseñada para ser lo mas rápida

posible de tal forma que pueda ser competitiva con otros paquetes de gráficas.

El Video sintetizador 3-VSD permite al usuario crear y observar un universo tridimensional artificial (3DAE). Los parámetros alimentados por el teclado manipulan dos puntos tridimensionales dentro del universo. (a,b,c) es la posición del ojo del usuario o cámara, las coordenadas que representan derecha, arriba y atrás. (D,E,F) es la orientación de la cámara o del punto al cual el ojo está mirando . Este punto siempre aparecerá en la mitad de la pantalla. La única limitación inherente es que la cámara no puede mirar directo hacia arriba o directo hacia abajo (por ejemplo A=D y C=F). En este caso el sistema literalmente desconoce que está pasando.

Los parámetros (A,-W) son alimentados tecleando la letra y el signo "=", seguidos hasta por cuatro dígitos hexadecimales.

Todos los parámetros de alimentación usan una notación de complemento a dos. El valor actual de cualquier parámetro puede examinarse tecleando la letra seguida por un signo de interrogación.

El parámetro (W) indica mediante dígitos diferentes a cero de izquierda a derecha lo siguiente: dibuje un vector desde este punto al siguiente, dibuje utilizando rojo, dibuje utilizando verde, dibuje utilizando azul. Los dígitos de color también indican el patrón a usar (F=sólido, etc...). Valores mas grandes que 1 en el primer dígito de W se reservan para gráficas primitivas de alta dimensión sin embargo éstas no están implementadas.
Por ejemplo , una línea amarilla podría dibujarse desde (0,0,0) a (100,140,180) mediante el uso de los dos comandos siguientes :

```
C = FA00(cr)            -Para regresar un poco
W = 1FF0(cr)            -Vector amarillo
<cr>                    -Proceso
D = 100(cr)
E = 140(cr)
F = 180(cr)
W = FF0(cr)             -Punto amarillo
(cr)                    -Proceso                    *   cr= return
```

Para comenzar estamos se asume que todos los parámetros tienen un valor de 0 .
También pueden indicarse letras dobles (AA-FF). Estos valores se an para incrementar o decrementar los parámetros correspondientes de una sola letra. El parámetro R se utiliza para indicar cuántas veces se debe de dar una indicación antes de que se requiera mas información. Por consiguiente la cámara o punto de partida puede moverse a través de una línea recta sin necesidad de ayuda. RR regresa el signo de los parámetros de letra doble.

También se permiten letras triples (AAA-FFF) y éstas incrementan o decrementan los parámetros de doble letra. Su efecto se produce al mover la cámara o el punto de partida en un patrón circular o elíptico sin necesidad de mayor alimentación de información por el teclado mediante el uso del parámetro (RRR).

Pueden obtenerse patrones circulares o elípticos en el plano XY mediante la siguiente fórmula:

Dados X0, Y0, Dx, Dy, y n como

(x0,y0)=centro

n^* n^* dx = diámetro x

n^* n^* dy = diámetro y

4^*n = número de puntos ó vectores

Sea

X = x0 + n^* n^* dx/2, Y = y0 + n^* dy/2
XX = 0, YY=n^* dy
XXX = -dx YYY=-dy
R = 4n, RR = 0 RRR=2n

donde
X= A o D Y= B, C o E, F

Por ejemplo la cámara puede ser movida en un circulo alrededor de (0,B,0) de la siguiente forma:

```
A = 480(cr)
C = 60(cr)
AA= (cr)
CC= C0(cr)
AAA=FFF0(cr)
CCC=FFF0(cr)
R=30(cr)
RR=(cr)
RRR=18(cr)
```

Un archivo 3DAE, SPIRL , el cual se dibuja a si mismo puede producirse de la siguiente forma

```
A=480(cr)    D=120(cr)    FF=4(cr)     DDD=FF0(cr)   R=c0(cr)
C=C0(cr)     E=30(cr)     AAA=FFC0(cr) EEE=FFF0(cr)  W=10FF(cr)
CC=180(cr)   EE=60(cr)    CCC=FFC0(cr) RRR=C(cr)     (cr)
```

Ninguno de estos parametros de doble o triple letra se muestran automaticamente pero cualquiera de ellos puede examinarse tecleando el

nombre y un signo de interrogación.

El ultimo digito de la derecha de P, es el parametro de perspectiva y determina:

0 = verdad
1 = dibujo
2 = isométrico

Los cuatro bits menos significativos de P son reservados para usarse por la rutina de PERSP. Los bits mas significativos estan disponibles para opciones de control dentro del manejador de control.

Los comandos de letra simple Z y U, tienen funciones especializadas :

Z causa que todos los parametros dobles y triples se inicializen a 0.

U causa que el ultimo punto escrito se borre del 3DAE. Este es siempre el ultimo punto en el archivo. Los comandos G y S son seguidos por el nombre de un archivo y permiten que se adicionen nuevos archivos al actual 3DAE (G) o que el 3DAE se grabe (S) en el disco bajo un nuevo nombre.

El archivo DOTCUBE de 3DAE podria producirse de la siguiente forma:

```
C=FA00(cr)      EE=(cr)     FF=(cr)       EE=10(cr)      (cr)
W=F0(cr)        DD=10(cr)   R=20(cr)      (cr)           D=100(cr)
R=20(cr)        (cr)        DD=FFF0(cr)   EE=(cr)        F+FF10(cr)
D=100(cr)       DD=(cr)     (cr)          R=(1F(cr)      FF=10(cr)
E=100(CR)       EE=10(cr)   DD=(cr)       D=FF00(CR)     (cr)
F=100(cr)       (cr)        EE=FFF0(cr)   F=FF10(cr)     W=(cr)
DD=FFF0(cr)     EE=(cr)     (cr)          FF=10(cr)      R=(cr)
(cr)            R=1F(cr)    EE=(cr)       (cr)           Z(cr)
DD=(cr)         F=F0(cr)    DD=10(cr)     E=FF00(cr)     (cr)
EE=FFF0(cr)     FF=FFF0(cr) (cr)          F=F0(cr)       SDOTCUBE(cr)
(cr)            (cr)        DD=(cr)       FF=FFF0(cr)
```

El archivo SPHERE de 3DAE podria producirse parcialmente de la siguiente forma:

```
C=FA00(cr)      R=(cr)      W=1FA5(cr)    RRR=(cr)       (cr)
W=1FA5(cr)      W=FA5(cr)   R=10(cr)      D=40(cr)   R=(cr)
R=10(cr)        (cr)        D=100(cr)     E=40(cr)   W=FA5(cr)
RRR=8(cr)       W=1FA5(CR)  E=FFC0(cr)    F=100(cr)  EEE=FFE0(cr)
D=100(cr)       R=10(cr)    F=40(cr)      DD=(cr)    (cr)
E=40(cr)        D=100(cr)   DD=(cr)       EE=80(cr)  W=1FA5(cr)
F=40(cr)        F=FFFX0(cr) EE=(cr)       R=4(cr)
EE=80(cr)       DD=(cr)     FF=80(cr)     DDD=(cr)   D=FFC0(cr)
DDD=FFE0(cr)    EE=80(cr)   EEE=(cr)      EEE=FFE0(cr)   E=40(cr)
EEE=FFE0(cr)    FF=(cr)     FFF=FFE0(cr)  FFF=FFE0(cr)   F=100(cr)
```

```
(cr)          EEE=FFE0(cr)   (cr)          (cr)       EE=80(cr)
EE=(cr)       FFF=(cr)       R=(cr)        FFF=20(cr)    FF=(cr)
```

INDICE DEL PROGRAMA

VIDEO	- Inicialización y alimentación de parámetros
CAMARA	- Secuencia en que las perspectivas se despliegan
PERSP	- Aplicar MOT a todos los puntos produciendo PERS.
ISQRT	- Encontrar la raiz cuadrada entera del doble entero de 32 bits
INORM	- Normalizar un componente de vector a una unidad esférica
SETFCB	- Mover determinado archivo al bloque de control de archivos de CPM
INFILE	- Leer un determinado archivo 3DAE del disco
OUTFILE	- Guardar un determinado archivo 3DAE en disco
ATOH	- Convertir ASCII a HEX
HTOA	- Convertir HEX a ASCII
PRINT	- Obtener los valores A, B,C,D,E F,R,P, y W en la consola

Las siguientes rutinas deben ser proporcionadas por el usuario para el desplegado de un sistema particular de gráficas.

INIT	- Inicializar el dispositivo de desplegado de gráficas
CLEAR	- Limpiar el area de desplegado inactiva
SWAP	- Comenzar utilizando otra area de desplegado al reproducir
PLOT	- Graficar los puntos actuales o dibujar vectores desde el punto anterior.

A continucion se presenta un listado del programa extraido literalmente de la fuente bibliográfica original.

```
        MOV   [BX+SI],AH      ;END OF INPUT
        MOV   AL,[BX]
        CMP   AL,'C'          ;GET ANOTHER FILE?
        JNC   VID15           ;NO
        CALL  SETFCB          ;PUT FILENAME INTO FCB
        PUSH  CS
        POP   DS
        MOV   BX,OFFSET FCB
        MOV   BYTE PTR FRECD[BX],2 ;READ FROM THIRD RECORD
        MOV   DX,ENDSEG       ;APPEND TO CURRENT 3DAE
        MOV   BX,ENDPOINT
        CALL  INFILE
        MOV   DS,MEMBOT
        CALL  CLEAR           ;CLEAR INACTIVE DISPLAY AREA
        CALL  PERSP           ;GENERATE PERSPECTIVE
        CALL  SWAP            ;SWAP ACTIVE DISPLAY AREAS
        CALL  PRINT           ;PRINT PARAMETERS
        JMPS  VIDEO
VID15:  CMP   AL,'S'          ;SAVE 3DAE?
        JNE   VID20           ;NO
        CALL  SETFCB          ;PUT FILENAME INTO FCB
        PUSH  CS
        POP   DS
        MOV   BX,OFFSET FCB
        MOV   BYTE PTR FRECD[BX],AL ;WRITE TO RECORD ZERO
        MOV   DX,MEMBOT
        MOV   BX,OFFSET A
        CALL  OUTFILE
        MOV   DS,MEMBOT
        JMP   VIDEO
VID20:  CMP   AL,'W'          ;SET WRITE MODE?
        JNE   VID30           ;NO
        MOV   SI,OFFSET W
        JMPS  VID70
VID30:  CMP   AL,'R'          ;SET REPEAT FACTOR?
        JNE   VID40           ;NO
        MOV   SI,OFFSET R
        JMPS  VID70
VID40:  CMP   AL,'P'          ;SET PERSPECTIVE?
        JNE   VID50           ;NO
        MOV   SI,OFFSET P
        JMPS  VID70
VID50:  CMP   AL,'U'          ;UNDO LAST POINT (ERASE)?
        JNE   VID55           ;NO
        MOV   ES,ENDSEG
        MOV   DI,ENDPOINT
        SUB   DI,8
        MOV   ENDPOINT,DI
        CMP   DI,8
        JNE   VID54
        MOV   AX,ES
        SUB   AX,1000H
        MOV   ENDSEG,AX
        MOV   ES,AX
VID54:  XOR   AX,AX
        MOV   ES:[DI],AX      ;ZERO PIXEL
        JMP   VIDEO
VID55:  CMP   AL,'Z'          ;ZERO 1ST AND 2ND DERIVATIVES?
        JNE   VID60           ;NO
        XOR   AX,AX
        MOV   [BX+SI+3],AA
        MOV   CX,10
        MOV   [BX],AX         ;ZERO AA THROUGH RRR
        INC   BX
        INC   BX
        LOOP  $-4
        JMP   VIDEO
VID60:  CMP   AL,'A'
        JB    VIDERR
        CMP   AL,'I'
        JA    VIDERR
        XOR   AH,AH           ;PARAMETER A THROUGH I
        MOV   SI,AX
        SUB   SI,41H
        SHL   SI,1
        ADD   SI,OFFSET A
VID70:  MOV   AH,AL           ;SAVE LAST LETTER
        INC   BX
        MOV   AL,[BX]
        CMP   AL,'-'
        JNE   VID80
        MOV   AL,'='
        MOV   [BX],AL
        CALL  HTOA            ;OUTPUT CURRENT VALUE
        INC   BX
        MOV   AL,' '
        MOV   [BX],AL
        INC   BX
        MOV   AL,CR
        MOV   AH,'S'
        MOV   [BX],AX
        MOV   DX,OFFSET INBUFF+2
        MOV   CL,OSTR
        INT   BDOS
        JMP   VIDEO
VID80:  CMP   AL,'='          ;INPUT HEX WORD
        JNE   VID90
        CALL  ATOH
        JMP   VIDEO
VID90:  CMP   AH,AL
        JNE   VIDERR
        ADD   SI,16           ;AA THROUGH FF FOR BEYOND
        JMPS  VID70
VIDERR: PUSH  DS
        PUSH  CS
        POP   DS
        MOV   DX,OFFSET WMSG  ;WHAT?
        MOV   CL,9STR
        INT   BDOS
        POP   DS
        JMP   VIDEO
;------------------------------ SUBROUTINES ------------------------------
;
;
CAMERA: XOR   AX,AX           ;DISPLAY SEQUENCE OF CAMERA ANGLES
        MOV   K,AX
        MOV   KK,AX
        MOV   KKK,AX
CAM10:  INC   K
        INC   KK
        INC   KKK
        TEST  W,-1
        JZ    CAM20
        MOV   ES,ENDSEG
        MOV   DI,ENDPOINT
        MOV   AX,0            ;WRITE POINT
        MOV   ES:[DI],AX
        MOV   AX,E
        MOV   ES:2[DI],AX
        MOV   AX,F
        MOV   ES:4[DI],AX
        MOV   AX,W
        MOV   ES:6[DI],AX
        XOR   AX,AX
        MOV   ES:8[DI],AX
CAM20:  MOV   AX,AA2          ;ADJUST CAMERA
        ADD   AA,AX
        MOV   AX,BBB
        ADD   BB,AX
        MOV   AX,CCC
        ADD   CC,AX
        MOV   AX,DDD
        ADD   DD,AX
```

```
        MOV   AX,EEE
        ADD   EE,AX
        MOV   AX,FFF
        ADD   FF,AX
        MOV   AX,AA
        ADD   A,AX
        MOV   AX,BB
        ADD   B,AX
        MOV   AX,CC
        ADD   C,AX
        MOV   AX,DD
        ADD   D,AX
        MOV   AX,EE
        ADD   E,AX
        MOV   AX,FF
        ADD   F,AX
        MOV   AX,D          ;COMPUTE OTM
        SUB   AX,A
        MOV   X,AX          ;X = D-A
        MOV   ZX,AX         ;ZX(1) = D-A
        NEG   AX
        MOV   XZ,AX         ;XZ(1) = A-D
        MOV   AX,E
        SUB   AX,B
        MOV   Y,AX          ;Y = E-B
        MOV   ZY,AX         ;ZY(1) = E-B
        MOV   AX,F
        SUB   AX,C
        MOV   Z,AX          ;Z = F-C
        MOV   XX,AX         ;XX(1) = F-C
        MOV   ZZ,AX         ;ZZ(1) = F-C
        IMUL  AX
        MOV   Q,AX
        MOV   Q+2,DX
        MOV   AX,X
        IMUL  AX
        ADC   Q,AX
        JNC   CAM40
        INC   DX
CAM40:  ADD   Q+2,DX        ;Q = X*X+Z*Z
        MOV   AX,Q
        MOV   DX,Q+2
        CALL  ISQRT
        MOV   XX+2,BX       ;XX(2) = SQRT Q
        MOV   XZ+2,BX       ;XZ(2) = SQRT Q
        MOV   YY,BX         ;YY(1) = SQRT Q
        MOV   AX,XZ
        IMUL  Y
        IDIV  BX
        MOV   YX,AX         ;YX(1) = Y*XZ(1)/SQRT Q
        MOV   AX,XX
        NEG   AX
        IMUL  Y
        IDIV  BX
        MOV   YZ,AX         ;YZ(1) = -Y*XX(1)/SQRT Q
        MOV   AX,Y
        IMUL  AX
        ADC   AX,Q
        JNC   CAM50
        INC   DX
CAM50:  ADD   DX,Q+2        ;Q = X*X+Y*Y+Z*Z
        CALL  ISQRT
        MOV   YX+2,BX       ;YX(2) = SQRT Q
        MOV   YY+2,BX       ;YY(2) = SQRT Q
        MOV   YZ+2,BX       ;YZ(2) = SQRT Q
        MOV   ZX+2,BX       ;ZX(2) = SQRT Q
        MOV   ZY+2,BX       ;ZY(2) = SQRT Q
        MOV   ZZ+2,BX       ;ZZ(2) = SQRT Q
        CALL  CLEAR         ;CLEAR INACTIVE DISPLAY AREA
        CALL  PERSP         ;GENERATE PERSPECTIVE
        CALL  SWAP          ;SWAP ACTIVE DISPLAY AREAS
        CALL  PRINT         ;PRINT PARAMETERS
        MOV   AX,RRR
        CMP   AX,KKK
        JNE   CAM60
        NEG   BBB
        NEG   CCC
        NEG   EEE
        NEG   FFF
        XOR   AX,AX
        MOV   KKK,AX
        JMPS  CAM70
CAM60:  SHR   AX,1          ;ADJUST AA2 AND DDD AT HALF PER
        CMP   AX,KKK
        JNE   CAM70
        NEG   AA2
        NEG   DDD
CAM70:  MOV   AX,RR
        CMP   AX,KK
        JNE   CAM80
        NEG   AA
        NEG   BB
        NEG   CC
        NEG   DD
        NEG   EE
        NEG   FF
        XOR   AX,AX
        MOV   KK,AX
CAM80:  MOV   AX,R
        CMP   AX,K
        JBE   CAM90
        JMP   CAM10         ;NEXT PERSPECTIVE
CAM90:  RET
;-------------------------------------------------------------
PERSP:  MOV   ES,MEMBOT     ;GENERATE PERSPECTIVE OF 3D POIN
        MOV   DI,OFFSET POINTS
PER10:  MOV   AX,ES:6[DI]   ;NEXT PIXEL
        TEST  AX,-1         ;ALL DONE?
        JNZ   PER20         ;NOT YET
        MOV   ENDSEG,ES
        MOV   ENDPOINT,DI
        RET
PER20:  MOV   AX,ES:[DI]    ;COMPUTE PERSPECTIVE OF NEXT PO
        SUB   AX,A
        MOV   X,AX          ;X = ES:[DI]-A
        MOV   AX,ES:2[DI]
        SUB   AX,B
        MOV   Y,AX          ;Y = ES:2[DI]-B
        MOV   AX,ES:4[DI]
        SUB   AX,C
        MOV   Z,AX          ;Z = ES:4[DI]-C
PER25:  IMUL  AX
        MOV   Q,AX
        MOV   Q+2,DX
        MOV   AX,Y
        IMUL  AX
        ADC   Q,AX
        JNC   PER30
        INC   DX
PER30:  ADD   Q+2,DX
        MOV   AX,X
        IMUL  AX
        ADC   Q,AX
        JNC   PER40
        INC   DX
PER40   ADD   Q+2,DX        ;Q = X*X+Y*Y+Z*Z
        MOV   AX,Q
        MOV   DX,Q+2
        CALL  ISQRT         ;BX = SQRT Q
        MOV   AX,I
        MOV   II,AX
        MOV   AX,J
        MOV   JJ,AX
PERSU:  MOV   AX,X
        IMUL  XX
```

```
        IDIV    XX+2
        MOV     I.AX
        MOV     AX.Z
        IMUL    XZ
        IDIV    XZ+2
        ADD     I.AX            ;I = XX*X+XZ*Z
        MOV     AX.X
        IMUL    YX
        IDIV    YX+2
        MOV     J.AX
        MOV     AX.Y
        IMUL    YY
        IDIV    YY+2
        ADD     J.AX
        MOV     AX.Z
        IMUL    YZ
        IDIV    YZ+2
        ADD     J.AX            ;J = YX*X+YY*Y+YZ*Z
        TEST    P.0EH           ;ISOMETRIC PERSPECTIVE
        JNZ     PER75           ;COMPUTE H
PER60:  MOV     AX.X
        IMUL    ZX
        IDIV    ZX+2
        MOV     H.AX
        MOV     AX.Y
        IMUL    ZY
        IDIV    ZY+2
        ADD     H.AX
        MOV     AX.Z
        IMUL    ZZ
        IDIV    ZZ+2
        ADD     H.AX            ;H = ZX*X+ZY*Y+ZZ*Z
        TEST    P.20H
        MOV     AX.H            ;BX STILL = SQRT Q
        CALL    INORM
        CMP     AX.CCLIP        ;POINT NOT IN FRONT OF CAMERA
        JLE     PER90
        TEST    P.0FH           ;TRUE PERSPECTIVE
PER70   JZ      PER80
        MOV     BX.H
        MOV     AX.I
        CALL    INORM           ;I = NORM*I/BX
        MOV     I.AX
        MOV     AX.J
        CALL    INORM           ;J = NORM*J/BX
        MOV     J.AX
PER80:  CALL    PLOT            ;PLOT PIXEL OR DRAW VECTOR
        ADD     DI.8            ;NEXT POINT
PER90:  INZ     PER95
        MOV     AX.ES
        ADD     AX.1000H
        MOV     ES.AX
PER95:  JMP     PER10
;--
ISQRT:  MOV     BX.DX           ;BX = INTEGER SQUARE ROOT OF DXAX
        SHL     BX.1
        OR      BL.AH
        OR      BL.AL
        ADD     BX.DX           ;INITIAL GUESS
        JG      ISQ10           ;DON'T RETURN ZERO
        INC     BX
        JG      ISQ10
        MOV     BX.7FFFH
        RET
ISQ10:  PUSH    AX
        PUSH    DX
        DIV     BX
        SUB     AX.BX
        CMP     AX.1
        JBE     ISQ90
        CMP     AX.-1
        JAE     ISQ90           ;0.1 OR -1 IS CLOSE ENOUGH
        SAR     AX.1
        ADD     BX.AX           ;NEXT APPROXIMATION
        POP     DX
        POP     AX
        JMPS    ISQ10
ISQ90:  POP     DX
        POP     AX
        RET
;--
INORM:  MOV     CX.NORM         ;AX = NORM*AX/BX
        IMUL    CX
        IDIV    BX
        RET
;--
SETFCB: PUSH    BX              ;MOVE FILENAME AT BX+1 INTO FCB
        PUSH    CS
        POP     ES
        MOV     DI.OFFSET FCB
        XOR     BX.BX
        MOV     ES:[DI].BL      ;DEFAULT DRIVE
        MOV     ES:FEX[DI].BL   ;CURRENT EXTENT
        MOV     AL.'
        MOV     CX.11           ;BLANK NAME
        INC     BX
SETF10: MOV     ES:[DI+BX].AL
        LOOP    SETF10
        POP     BX
        INC     BX
        MOV     AX.[BX]
        CMP     AH.':'
        JNE     SETF20
        SUB     AL.40H          ;DRIVE DESIGNATION
        MOV     ES:[DI].AL
        INC     BX
        INC     BX
        INC     DI
SETF20: MOV     AL.[BX]
        CMP     AL.'.'
        JE      SETF40
        CMP     AL.0
        JE      SETF60
        MOV     ES:[DI].AL
        INC     BX
        JMPS    SETF20
SETF40: INC     BX              ;FILENAME EXTENSION
        MOV     CX.3
        MOV     DI.OFFSET FCB+9
SETF50: MOV     AL.[BX]
        CMP     AL.0
        JE      SETF60
        MOV     ES:[DI].AL
        INC     BX
        INC     DI
        LOOP    SETF50
SETF60: RET
;--
INFILE: PUSH    DX              ;SEGMENT OF MEMORY AREA
        PUSH    BX              ;OFFSET OF MEMORY AREA
        MOV     DX.OFFSET FCB
        MOV     CL.FOPEN
        INT     BDOS
        POP     BX
        POP     DX
        CMP     AL.-1
        JNE     INF40
        MOV     DX.OFFSET FNFMSG ;FILE NOT ON DISK
        MOV     CL.COSTR
        INT     BDOS
        RET
INF40:  PUSH    DX
        PUSH    BX
```

```
        MOV   CL,FDMAS
        INT   BDOS              ;SET DMA SEGMENT
        POP   DX
INF50:  PUSH  DX
        MOV   CL,FDMAO
        INT   BDOS              ;SET DMA OFFSET
        MOV   DX,OFFSET FCB
        MOV   CL,FREAD
        INT   BDOS
        TEST  AL,-1
        JNZ   INF90             ;END OF FILE
        POP   DX
        ADD   DX,80H            ;NEXT RECORD
        JNZ   INF50
        MOV   BX,DX
        POP   DX
        ADD   CX,1000H
        JMPS  INF40
INF90:  ADD   SP,4
        RET
;--
OUTFILE:PUSH  DX                ;SEGMENT OF MEMORY AREA
        PUSH  BX                ;OFFSET OF MEMORY AREA
        MOV   DX,OFFSET FCB
        MOV   CL,FDELETE
        INT   BDOS
        MOV   DX,OFFSET FCB
        MOV   CL,FMAKE
        INT   BDOS
        POP   BX
        POP   DX
        CMP   AL,-1
        JNE   OUTF40
        MOV   DX,OFFSET NRODMSG ;NO ROOM IN DIRECTORY
        MOV   CL,COSTR
        INT   BDOS
        RET
OUTF40: PUSH  DX
        PUSH  BX
        MOV   CL,FDMAS
        INT   BDOS              ;SET DMA SEGMENT
        POP   DX
OUTF50: PUSH  DX
        MOV   CL,FDMAO
        INT   BDOS              ;SET DMA OFFSET
        MOV   DX,OFFSET FCB
        MOV   CL,FWRITE
        INT   BDOS
        TEST  AL,-1
        JZ    OUTF70            ;RECORD ON DISK
        MOV   DX,OFFSET NRODMSG
        MOV   CL,COSTR
        INT   BDOS
        ADD   SP,4
        JMPS  OUTF90
OUTF70: POP   DX
        ADD   DX,80H            ;NEXT RECORD
        JZ    OUTF80
        CMP   DX,ENDPOINT
        JBE   OUTF50
OUTF80: MOV   BX,DX
        POP   DX
        CMP   BX,0
        JE    OUTF85
        CMP   DX,ENDSEG
        JAE   OUTF90
OUTF85: ADD   DX,1000H
        JMPS  OUTF40
OUTF90: MOV   DX,OFFSET FCB
        MOV   CL,FCLOSE
        INT   BDOS              ;CLOSE FILE
        RET
```

```
        INT   BDOS
        RET
;*****************************************************************
;       IF ROMED, THE FOLLOWING 3 WORDS AS WELL AS THE FCB
;                 MUST BE MOVED INTO A DATA SEGMENT
;
MEMBOT  RW    1                 ;BOTTOM OF AVAILABLE MEMORY
ENDPOINT RW   1                 ;OFFSET OF LAST POINT IN 3DAE
ENDSEG  RW    1                 ;SEGMENT OF LAST POINT IN 3DAE
GETBOT  DB    18,0,0,0,0        ;GET BOTTOM OF MEMORY BIOS CALL
WMSG    DB    'WHAT?',CR,'$'
FNFMSG  DB    'FILE NOT FOUND',CR,'$'
NRODMSG DB    'NO ROOM ON DISK',CR,'$'
        DSEG                    ;THE FOLLOWING DATA IS ASSIGNED
        ORG   0                 ;TO THE BOTTOM OF AVAILABLE MEMORY
;
INBUFF  RB    80H               ;CONSOLE INPUT BUFFER
OUTBUFF RB    80H               ;CONSOLE OUTPUT BUFFER
;-----------------------------------------------------------------
        ORG   100H
A       RW    1                 ;CAMERA POSITION
B       RW    1
C       RW    1
D       RW    1                 ;CAMERA ORIENTATION
E       RW    1
F       RW    1
R       RW    1                 ;REPEAT FACTOR
K       RW    1                 ;STEP FACTOR
AA      RW    1                 ;POSITION CHANGE
BB      RW    1
CC      RW    1
DD      RW    1                 ;ORIENTATION CHANGE
EE      RW    1
FF      RW    1
RR      RW    1                 ;PERIOD FACTOR
KK      RW    1
AA2     RW    1                 ;SECOND DERIVATIVES
BBB     RW    1
CCC     RW    1
DDD     RW    1
EEE     RW    1
FFF     RW    1
RRR     RW    1                 ;2ND DERIVATIVE PERIOD
KKK     RW    1
W       RW    1                 ;WRITE MODE (PIXEL CODE)
X       RW    1                 ;WORKING STORAGE
Y       RW    1
Z       RW    1
V       RW    1
P       RW    1                 ;TYPE OF PERSPECTIVE
Q       RW    2                 ;LONG INTEGER Q÷Z = HIGH-ORDER BITS
XX      RW    2                 ;FRACTION XX÷Z = DENOMINATOR
XZ      RW    2                 ;FRACTION
YX      RW    2                 ;FRACTION      THESE 8 FRACTIONS ARE THE
YY      RW    2                 ;FRACTION      NONZERO ENTRIES OF THE 3×3
YZ      RW    2                 ;FRACTION      ORIENTATION TRANSFORMATION
ZX      RW    2                 ;FRACTION      MATRIX (OTM)
ZY      RW    2                 ;FRACTION
ZZ      RW    2                 ;FRACTION
H       RW    1                 ;DISTANCE OF SIGHT
I       RW    1                 ;(I,J) = CURRENT DISPLAY POINT DISPLACEMENT
J       RW    1                 ;      FROM CENTER OF SCREEN
II      RW    1                 ;LAST I
JJ      RW    1                 ;LAST J
```

4.7 ANIMACION DE IMAGENES DE ALTA CALIDAD

Para realizar animación de alta técnica en microcomputadoras es necesario antes que nada contar con el equipo adecuado y utilizar técnicas de graficación especializadas.

La animación de alta técnica es realizada en 2D y en 3D.

4.7.1 SISTEMA DE GRAFICACION PROFESIONAL PARA IBM PC

El Sistema Profesional para gráficos (SPG) de IBM, consta de un monitor de alta resolución de color y una tarjeta controladora gráfica que muestra una imágen libre de parpadeo de 640 x 480 "pixels" con un rango de 256 a 4096 colores simultáneos posibles. Para entender mejor la diferencia entre SPG y otros sistemas, se presenta una breve reseña de las posibilidades disponibles para la IBM PC.

Muchas computadoras personales vienen equipadas de fábrica con capacidad para gráficas en alta resolución, la IBM PC ofrece ranuras de expansión. La tercera parte de los vendedores llenan estas ranuras con una amplia variedad de tarjetas controladoras que proveen una gran flexibilidad para el trabajo con gráficas.

Muchas de estas tarjetas gráficas, cumplen con un doble propósito, simular una tarjeta gráfica estandar de 320 x 200 "pixels" haciendo a la PC compatible con los programas para negocios y proveer una gran variedad de opciones gráficas de alta resolución en el rango del monocromático al color de 1024 x 1024 "pixels".

La tarjeta maestra para gráficos Tecmar, es popular porque simula una tarjeta estandar y provee una resolución a color de 640 x 480 pixles con opción para 16 diferentes colores simultáneos. Utiliza un conector TTL (transistor-transistor-logic) de nueve pins y cuenta con generación de caracteres para despliegue de texto. Como la tarjeta emite una señal de interlace, el monitor parpadea cuando despliega imágenes de alta resolución. Muchos usuarios han tenido que utilizar un monitor de fósforo de larga pesistencia para un despliegue aceptable. Esta configuración está limitada a 16 colores.

Algunas tarjetas gráficas como la Conographic's Cono Color 40, requiere un monitor especial de alta velocidad no estandar, de búsqueda horizontal y entrada analógica.

La "Artist 1" provee una resolución de 1024 x 1024 con 256 colores simultáneos desplegando una paleta de 16 millones de

colores , pero no tiene simulación para generación de textos, por lo que se necesita un doble monitor para trabajo no gráfico. También se necesita un monitor de 19 pulgadas y de fósforo de larga persistencia.

El sistema profesional para gráficas.

El IBM SPG consta de un monitor de color de alta resolución y una tarjeta controladora de gráficos. El sistema emula la generación de texto y la tarjeta estandar para gráficos a color para ejecutar programas de negocios. Provee gráficas de una resolución de 640 x 480 "pixels" desplegando 256 colores simultáneos en un sistema integrado.

La tarjeta SPG ocupa dos ranuras de la IBM. Un conector TTL de nueve pins transporta la señal a un monitor especial de color que se ve mucho mejor que el monitor estandar de la máquina, SPG no controla un monitor estandar a color.

Las fotos que aparecen a continuación muestran la comparación de paletas de colores y representación de líneas finas para la tarjeta estandar, la tarjeta "Tecmar Graphics Master" y la "IBM SPG".

Con la "Graphics Master" la imágen tiende a parpadear a pesar del uso del monitor de fósforo de larga perisistencia, SPG no parpadea mientras se estén desplegando las mismas imágenes.

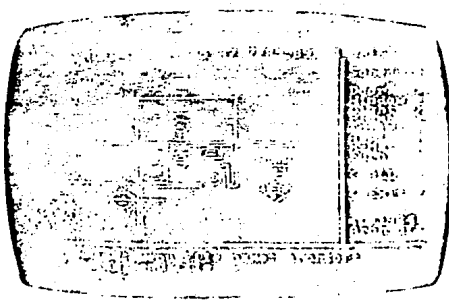


Fig. 4.6 Un dibujo arquitectónico usando Archsoft's AE/CAD . Con la tarjeta gráfica estandar , 4 colores y 320 x 320 "pixels".

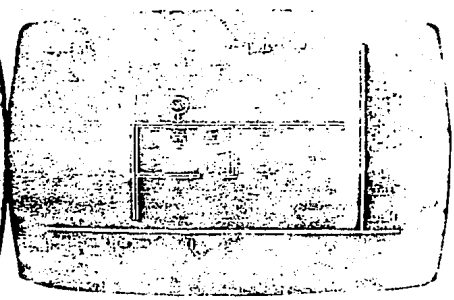


Fig. 4.7 El mismo dibujo usando la tarjeta Tecmar, con 16 colores y 640 x 480 "pixels"

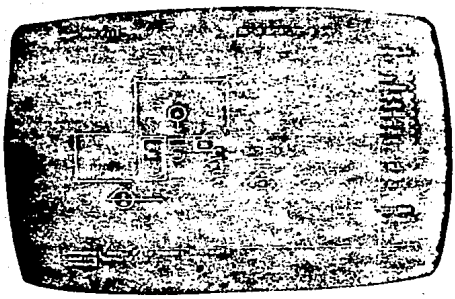


Fig. 4.8 El mismo dibujo usando el Sistema Profesional para gráficos SPG, con 256 colores y 640 x 480 "pixels"

Al usar el SPG con un sistema de monitor simple, la simulación de generación de textos , es un poco mas lenta que en la tarjeta gráfica de color y la Tecmar. Durante la edición de textos SPG fue lenta y esto puede interferir con el trabajo de producción. Hay que tener en mente que puede utilizarse SPG en una configuración de monitor dual con despliegue de texto en un monitor monocromático.

4.7.2 ANIMACION DE ALTA TECNICA EN 2D

Antecedentes

Los primeros intentos en hacer animación por computadora en 2D, utilizaron un graficador para generar cada una de las imágenes.

En los años 60's se hizo la animación de un comercial de la Boeing (compañía de aviación norteamericana), usando un graficador y coloreando a mano cada uno de los cuadros.

De la misma forma, se hizo la pelicula candiendse "hambre".

Este tipo de técnica se ha aplicado principalmente para la producción de caricaturas, en las cuales la computadora apoya en la producción de cuadros intermedios.
El animador diseña exclusivamente los cuadros inicial y final (cuadros clave) y la máquina genera mediante un algoritmo, las posiciones intermedias a estos cuadros clave, ahorrando mucho trabajo de dibujo y diseño.

En la figura se muestra la producción de cuadros intermedios para la figura de un dragón.

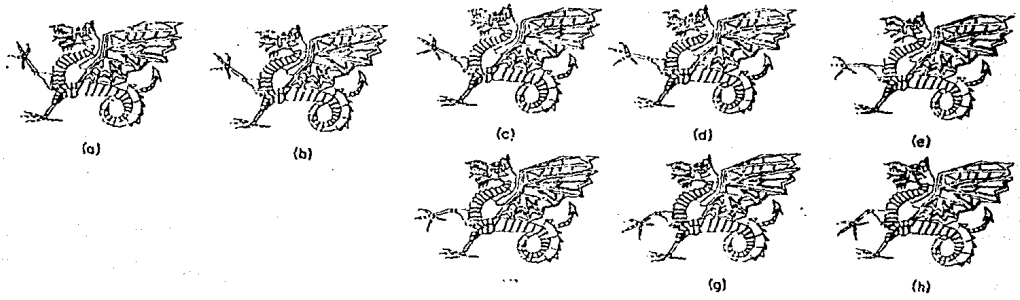


Fig. 4.9 Producción de cuadros intermedios para un dragón.

El problema puede reducirse a lo siguiente, dadas dos figuras claves, generar una o mas figuras que se encuentren enmedio de las anteriores.

Este proceso de interpolación, requiere que ambas figuras (la inicial y la final), tengan el mismo número de líneas y que solamente varíen en movimientos continuos visibles.

Para la animación en dos dimensiones se aplicarán los algoritmo convencionales para generación de ventanas, puntos de visión, algortimos de recorte y dibujo de líneas.

También será necesario efectuar transformaciones, rotaciones y escalamientos.

Desafortunadamente en este tipo de animación, cualesquiera dos cuadros, involucran partes que no son visibles. Algunas partes del objeto o de la figura pueden desaparecer de un cuadro a otro. Para hacer los cuadros intermedios correctamente, debe proporcionarse información acerca de las diferentes secciones del

cuerpo que pueden moverse enfrente o atrás de otras secciones.

4.7.3 ANIMACION DE ALTA TECNICA EN 3D

Lo mas sofisticado y difícil del uso de la graficación por computadora es la producción de imágenes que puedan manejarse en tres dimensiones.

Pueden producirse imágenes de muy alta calidad en una microcomputadora con equipo profesional como el que se describió anteriormente, aunque el tiempo en el cual se obtiene una imagen sea mucho mayor que para un sistema de gran tamaño.

Las imágenes de alta calidad se obtienen usando modelos primitivos, modelado de sólidos y técnicas de sombreado junto con otras técnicas especializadas de graficación.

4.7.3.1 Una Apple para animación.

James Leatham, talentoso programador y productor de películas que reside en Chester, New York, utiliza una Apple II, un paquete gráfico "Sublogic A2-3D1" y un equipo de mesa hecho en casa, con lo cual ha creado fantásticas escenas de animación para la película Asteroide. La película trata acerca de una época espacial del cinturón de asteriodes. En la escena que Jim trabajó, la computadora "nave", detecta y analiza un asteriode de particular valor, crea una simulación del asteriode y lo rota en tres dimensiones. Un campo magnético dentado rota alrededor del eje. La foto de abajo muestra otra de las creaciones de James. Es una función matemática. En la película, las dos funciones aparecen como un modelo primitivo de una montaña que crece y decrece.

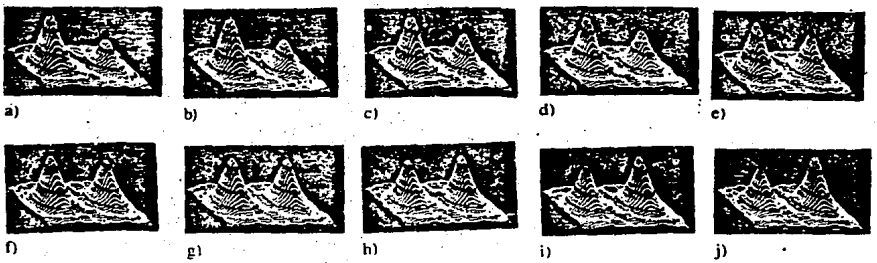


Fig. 4.10 Cuadros generados por James Leatham. Utilizó una cámara

de cine Super-8 con el control directo de una microcomputadora ple.

James utiliza el paquete "SubLogic A2-3D1" para definir una base tridimensional para el asteroide. Se introducen las coordenadas del objeto y mediante un programa de control en BASIC se rota en incrementos de ángulo simples. James diseño una mesa especial para colocar la cámara y el rotador de filtros. El programa de control puede mover el filtro adecuado al frente de la cámara y accionar el disparador para cada filtro de diferente color . El programa de control y el mecanismo de la cámara realizan toda la labor de filmación de la secuencia de animación.

La pelicula se proyectó a 18 cuadros por segundo, se utilizó una Apple II con una cámara Super 8 Euming 881 PMA. Se usa un monitor blanco y negro para máxima resolución y es por esto que se necesitan filtros de color. La computadora puede abrir el obturador de la cámara y mantenerlo asi el tiempo necesario. También puede capturar el modo de despliegue de texto de las dos páginas de alta resolución. Cada nueva imágen se dibuja en una página alterna, cuando se hace esto, la nueva página se coordina mediante el programa de la computadora y se borra la página anterior, entonces se rota el filtro y se abre el obturador el tiempo necesario.

James Leatham es un pionero en el campo de la animación casera.

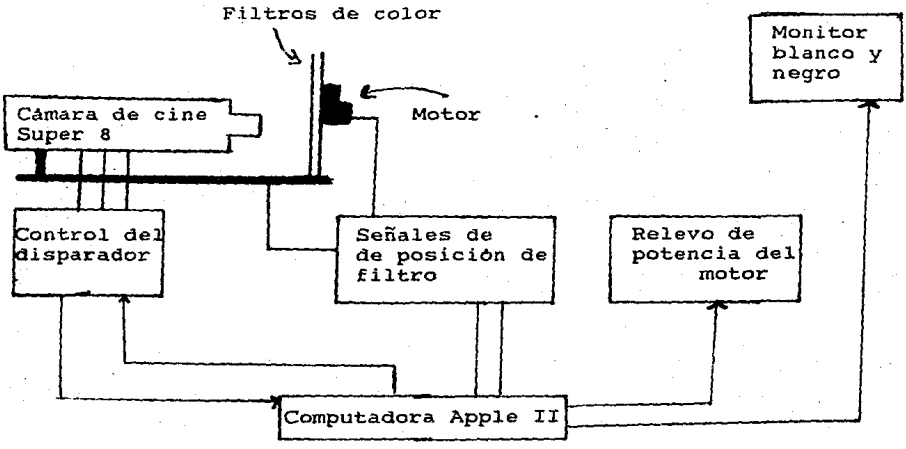


Fig. 4.11 Equipo casero de animación

4.7.3.2 MODELADO DE OBJETOS

Consiste en la descripción o construcción de objetos tridimensionales . Pueden implementarse dos clases de modelos:

- Modelos primitivos basados en dibujos de lineas tridimensionales.

- Modelos sólidos basados en superficies bidimensionales y tridimensionales.

El mas viejo y simple tipo de modelos tridimensionales es el modelo primitivo. Este se compone de una lista de puntos dados por sus coordenadas o por una secuencia de instrucciones del tipo:

moveabs <<X,Y,Z>>	Se mueve a una nueva localidad
lineabs <<X,Y,Z>>	Dibuja una linea de la localidad actual a una nueva localidad

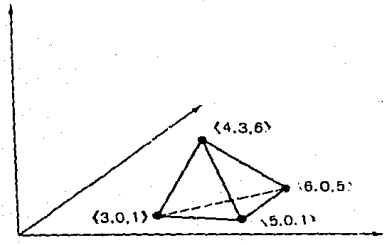


Fig. 4.12 El tetraedro

El tetraedro que se muestra en la figura puede representarse como:

```
moveabs <<3,0,1>>;
lineabs <<5,0,1>>;
lineabs <<6,0,5>>;
lineabs <<3,0,1>>;
lineabs <<4,3,6>>;
lineabs <<6,0,5>>;
moveabs <<5,0,1>>;
lineabs <<4,3,6>>;
```

Un modelo primitivo es muy simple y no tiene un alto grado de realismo. Para éste propósito se usa un modelo sólido basado en la descripción de superfices.
Se usan tres categorias de descripciones dependiendo de la forma del objeto:

- Descripción por un conjunto de poligonos
- Descripción por ecuaciones de superficies algebraicas
- Descripcion por fragmentos de superficies o parches

La técnica mejor conocida es la que describe al sólido mediante una colección de poligonos. Se especifica una lista de poligonos o puntos donde cada poligono se define por sus vértices. Por ejemplo el tetraedro puede definirse como:

```
POINT <<3,0,1>>
POINT <<5,0,1>>
POINT <<6,0,5>>
POINT <<4,3,6>>
POLYGON 1,2,4
POLYGON 2,3,4
POLYGON 1,3,4
POLYGON 1,2,3
```

Cualquier objeto puede modelarse con poligonos. Si el objeto es una curva muy grande se requieren muchos poligonos, por esta razón se utilizan superficies algebraicas y parches.

Superficies algebraicas

Son superficies descritas matemáticamente mediante ecuaciones. Las mas populares incluyen superficies cuadráticas como esferas, conos, cilindros y elipsoides.

Las dos superficies de campo mas conocidas son los trabajos de Coons y Bezier. En la técnica introducida por Coons, una superficie de campo se determina por las curvas de acotamiento P(U,0), P(U,1), P(O,V), P(1,V) donde U y V están en el rango 0,1 y por una interpolación lineal entre estas cuatro curvas.

$$
\begin{aligned}
Q(U,V) = {} & P(U,0)F_{00}(V) + P(U,1)F_{00}(V) + P(0,V)F_{00}(U) \\
& +P(1,V)F_{01}(U) - P(0,0)F_{00}(U)F_{00}(V) \\
& - P(0,1)F_{00}(U)F_{01}(V) \\
& -P(1,0)F_{01}(U)F_{00}(V) - P(1,1)F_{01}(U)F_{01}(V)
\end{aligned}
$$

Donde las funciones $F_{ij}(U)$ sirven para las condiciones de acotamiento que forman las superficies.

En el método introducido por Bezier (1972), una superficie se calcula de N curvas, dados M puntos. Estos puntos P se denominan puntos de control y la superficie se calcula como sigue:

$$Q(U,V) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_{in}(U) B_{jm}(V)$$

donde U y V estan en el rango (0,1)

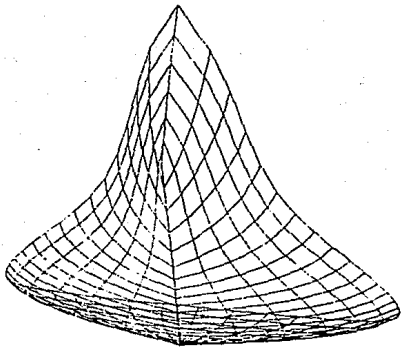


Fig. 4.13 Una superficie de Bezier

La figura muestra un ejemplo de una superficie de Bezier en un dibujo de lineas.

Es importante notar que un objeto se construye sólo como una combinación de objetos simples que pueden ser modelados de diferentes maneras.

Creación de objetos

Existen tres caminos para construir objetos tridimensionales de acuerdo con la clase de objeto y su grado de complejidad.

-Digitalización
-Edición gráfica
-Programación

Digitalización

Consiste en tomar una fotografia o hacer el dibujo de un objeto trazando un cuadriculado del objeto, y luego fotografiándolo desde varios puntos de vista, frente, lado, arriba y abajo. Los puntos de intersección del cuadriculado se numeran de tal forma que el mismo punto, en varias fotografias tenga el mismo número. Se digitalizan varias fotografias, usando un programa que reconstruye la imágen en tres dimensiones. Esta técnica es usada

en particular para objetos con formas irregulares.
Otro método descrito por Blum se basa en fotos tomadas desde dos distancias diferentes.

Edición gráfica

Con un editor de gráficas, se pueden construir objetos en tres dimensiones, mediante el ensamble y combinación de objetos simples. Esta es una operación interactiva ya que el usuario debe estar posibilitado para ver inmediatamente los efectos de los comandos dados.

Programación

Para los objetos basados en patrones complejos repetitivos o funciones matemáticas se utiliza la programación.

Es conveniente contar con la posibilidad de definir tipos gráficos, como puede ser el tipo figura. Esto se haria como sigue:

1. Se determinan las caracteristicas de la figura o sea los parámetros.
2. Se desarrolla un algoritmo para construir la figura con la ayuda de parámetros.

El uso de tipos gráficos proporciona las siguientes ventajas a los programadores :

1. Las operaciones pueden restringirse a tipos especificos.
2. Las figuras pueden usarse exactamente como otros tipos cualesquiera (Ej. podriamos definir un arreglo de cubos, o un registro con figuras de campo).

4.7.3.3 SUPERFICIES OCULTAS

En un dibujo tridimensional es necesario borrar las lineas que no deben verse para alcanzar el realismo de una imagen. Este proceso ha sido un tema de investigación común desde los primeros sistemas tridimensionales. Se han propuesto numerosos algoritmos para resolver este problema , los cuales pueden clasificarse en tres categorias:

1. El espacio del objeto

Estos algoritmos estan basados en cálculos y comparaciones entre los objetos geométricos tal y como fueron definidos por el usuario en el espacio tridimensional o en el espacio del objeto.

A pesar de que son muy acertados, su costo crece muy rapidamente con la complejidad de una escena. A pesar de que varios de esos algoritmos son bien conocidos (Appel 1967, Galimberti y Montanari 1969) no han sido usados aun.

El algoritmo de Appel está basado en el concepto de "invisibilidad" cuantitativa. Un segmento de linea es visible solamente si todos los puntos sobre el tienen una invisibilidad cuantitativa de 0, esta técnica detecta cambios en la invisibilidad cuantitativa a lo largo de un segmento y dibuja las porciones visibles

2. Espacio de la imágen

Estos algoritmos hacen un uso extensivo del hardware. Están basados en el principio de que los objetos se componen de caras poligonales y debe decidirse cual cara está visible para cada "pixel" de la pantalla. Es mucho mas eficiente que el algoritmo previo, tambien se está limitado por el costo porque el número de "pixels" que permanece constante depende de la complejidad de la escena, mas aun estos algoritmos están atados a la tecnologia de pantalla tipo rastreador. Los algoritmos mas conocidos de este tipo son los de Watkins 1970 y Warnock 1969.

3. Lista de prioridades

Estos algoritmos representan un compromiso entre los dos algoritmos anteriores. Cuentan con dos etapas:

-En el espacio del objeto, el proceso consiste en la construcción de una lista de prioridades entre los objetos de acuerdo a su profundidad.
-En el espacio de la imagen el proceso consiste en determinar la visibilidad de los objetos.
Los algoritmos mas conocidos para lista de prioridades fueron desarrollados por Schumacker 1969 y Encarnacao 1970.
Se resumen a continuación algunos de los algoritmos que se usan comunmente en animación por computadora:
-Subdivision de Warnock
-Lineas de búsqueda
-Buffer profundo
-Tamaño del buffer

Subdivisión de Warnock

En este algoritmo la pantalla se divide en varias ventanas. Se consideran tres casos:

1. No hay nada que ver en la ventana
2. Lo que se va a ver en la ventana es muy fácil de dibujar
3. Lo que se va a ver en la ventana es muy dificil de dibujar, en cuyo caso la ventana debe de subdividirse en cuatro

pequeñas ventanas.

Este algoritmo es tipicamente recursivo, terminando su procesamiento bajo una de las siguientes condiciones:

1. No hay nada que ver y la ventana se colorea con el color de fondo
2. La ventana se reduce a un "pixel" y ya que no es posible subdividirla, se colorea con el color apropiado.
3. La ventana es fácil de colorear. Esto es posible:

   A) Cuando un polígono rodea la ventana. En este caso la ventana se colorea con el color apropiado.

   B) Sólo un polígono intercepta la ventana. En este caso la ventana se colorea parcialmente con el color de fondo y parcialmente con el color apropiado para el polígono.

Lineas de búsqueda

Los algoritmos de líneas de búsqueda (Wylie 1967; Bouknight y Kelley 1970; Watkins 1970) están basados en el siguiente esquema codificado en un lenguaje similar a Pascal.
Considérense dos arreglos:

```
INTENSITY: array (1..LENGTH) of PIXEL;
DEPTH: array (1..LENGTH) of REAL;
```

1. Para cada "pixel" IX en la línea de búsqueda.
   DEPTH (IX):=valor de fondo;
   INTENSITY (IX)= valor máximo

2. Para cada polígono en la escena se encuentran los "pixels" en la línea de búsqueda que están dentro del polígono.

```
Para cada uno de estos "pixels" IX:
 A. Calcular la profundidad Z del polígono en <<X,Y>>.
 B. if Z<DEPTH(IX)
    then begin
              DEPTH(IX):=Z;
              INTENSITY (IX):=valor de sombreado correspondiente
                              al polígono

         end;
```

3. Cuando se termina de procesar una línea de búsqueda el arreglo INTENSITY contiene los valores correctos y los despliega.

Buffer profundo

Desarrollado por Catmull en 1975. Este algoritmo es un método simple de eliminar las superficies ocultas, requiere de un buffer profundo (Z-Buffer) que consiste de un arreglo que contiene los valores Z para cada "pixel". Este algoritmo es similar al de líneas de búsqueda básicas, pero los valores de la intensidad se guardan en un arreglo bidimensional en lugar de en un arreglo de una dimensión.

```
INTENSITY: array(1..HEIGHT, 1..LENGTH) of PIXEL (*frame buffer*)
DEPTH: array (1..HEIGHT, 1..LENGTH) of REAL; (*Z-buffer*)


1. Para cada "pixel"<<IX,IY>>
   INTENSITY(IX,IY):=valor de fondo
   DEPTH[IX,(Y):=valor máximo

2.  Para cada polígono en la escena encontrar todos los  "pixels"
<<IX,IY>>  que  están  dentro del polígono proyectado.  Para cada
"pixel":
   A.Calcular la profundidad Z en <<IX,IY>>
   B.if Z<DEPTH(IX,IY) (el polígono mas cercano)
        then

        begin
           DEPTH(IX,IY):=Z;
           INTENSITY(IX,IY):=valor de sombreado del polígono  co-
rrespondiente
        end;

3.  Al final del proceso de la escena,  el arreglo INTENSITY con-
tiene el cuadro requerido.
```

Tamaño del buffer

Este algoritmo fue desarrollado especialmente por sistemas de animación ANTTS (CSURI 1979). Está basado en el uso de un "frame buffer" el cual guarda secuencias de "pixels" con el mismo valor en una línea de búsqueda. El "buffer" es una lista de arreglos fijos en memoria, uno para cada línea de búsqueda en la pantalla.

Despues de la conversión de objetos el algoritmo los pasa al "buffer". Cuando se ha procesado completamente el cuadro los datos se descomponen en valores tridimensionales de "pixels". Los valores Z se comparan con los mismos valores XY en el buffer de "pixels", causando un fuerte efecto en la eliminación de líneas ocultas.

El "buffer" está guardado en memoria principal. El "buffer" de "pixels" está en memoria secundaria y contiene el color y profundidad de cada "pixel" así como el tipo de imagen, información de luz e identificación del objeto. Este enfoque requiere mucho menos memoria principal que el algoritmo de Z-buffer

Modelos de reflexión de luz

Si se eliminan las caras ocultas de una esfera aproximada por polígonos y se colorean todos los polígonos visibles con el mismo color rojo se obtiene un círculo rojo, esto es debido a que nuestra percepción de la tercera dimensión se mejora grandemente por la reflexión de la luz. En el caso de la esfera, los diferentes puntos de la superficie no reflejan la luz del mismo modo, esto quiere decir que la esfera no debe colorearse uniformemente.

Teoricamente, hay dos tipos de superficie:

-Reflectores ideales especulares que son como espejos perfectos
-Reflectores ideales difusos que corresponden a dos superficies identicas.

En efecto, la mayoria de las superficies reales no son reflectoras ideales especulares, ni tampoco reflectores ideales difusos. Por esta razón se han desarrollado modelos de reflexión. Los componentes del ambiente corresponden a la luz que es uniformemente incidente y que es reflejada por la superficie en iguales direcciones .Los componentes difusos consisten de la luz que emana desde un punto fuente, pero que se difunde igualmente en todas las direcciones. El componente especular representa la luz de alta intensidad, la luz que está concentrada alrededor del punto de impacto del rayo de incidencia. La luz de alta intensidad tiene el color de la luz fuente.
El primer modelo de luz que tomó en cuenta esos componentes fue desarrollado por Bui-Tuong Phong (1975).

4.7.3.4 Sombreado

Ya se ha mencionado que existen tres clases de descripciones para modelos sólidos:
-Un conjunto de polígonos
-La ecuacion de superficies algebraicas
-Segmentos de superficie o parches

Para cada una de estas descripciones el sombreado debe calcularse usando los modelos de reflexión antes presentados, sin embargo

los modelos de reflexión no proveen caminos directos para calcular el sombreado completo de un objeto, sino solamente para la intensidad de la luz en un punto especifico. Las técnicas de sombreado usadas dependen del tipo del objeto.
Para la interacción de poligonos, se han desarrollado tres caminos básicos para el sombreado de objetos: sombreado constante, sombreado de Gouraud y sombreado de Phong.

Sombreado constante

Este modelo involucra cálculos de intensidades simples para cada polígono, esto implica las siguientes suposiciones:

1.-La fuente de luz está en el infinito
2.-El observador está en el infinito
3.-El poligono no es una aproximación de una superficie curva.

Esto produce un buen resultado para un cubo, pero muy pobre para una esfera. Mas aun el sombreado constante produce el efecto de bandas descrito por E.Mach en 1865 como sigue:

"En dondequiera que la curva de intensidad de luz de una superficie iluminada tiene una curva cóncava o convexa con respecto al eje de la abcisa, aparece una parte brillante o mas obscura, que las que la rodean."

Fig. 4.14 Texto sombreado a 3D (sombreado constante)

Sombreado de Gouraud

Gouraud 1971, introdujo un método de interpolación de la intensidad de sombreado que elimina las discontinuidades del sombreado

constante. Sin embargo el efecto de bandas permanece visible cuando la pendiente de la función de sombreado cambia. El principio del sombreado de Gouraud es como sigue:

1. Para cada vèrtice común a varios polígonos, la normal a cada polígono se calcula como un vector perpendicular al plano de dicho polígono.

2. Para cada vèrtice se calcula una única normal o promedio de las normales obtenidas previamente para la superficie.

3. Las intensidades de las superficies se calculan utilizando las normales a los vèrtices y uno de los modelos de luz presentados anteriormente.

4. Como cada polígono tiene un sombreado diferente en cada vèrtice, el sombreado en cualquier punto dentro del polígono se encuentra por interpolación lineal de las intensidades de los vèrtices a lo largo de cada orilla y por lo tanto entre orillas a lo largo de cada línea de bùsqueda (El sombreado de Gouraud está basado en un algoritmo de líneas de bùsqueda como el algoritmo de Watkins)

Sombreado de Phong

Bui-Tuong Phong (1975) propuso un mètodo de sombreado por interpolación de vectores normales. Con este enfoque el sombreado de un punto se calcula por medio de la orientación de la normal aproximada. Con el sombreado de Phong se obtiene una mejor aproximación de la curvatura de la superficie y debido a la simulación de la reflexión especular se obtiene un acabado fuera de lo común, sin embargo el mètodo requiere mas cálculos.

Aparte del tiempo de cómputo, existe otro problema en los algoritmos de Gouraud y Phong.
Si un objeto y su fuente de luz se rotan juntos en el plano de la imagen, el sombreado del objeto cambiará de manera contraria, a lo que se espera. Esto se debe al hecho de que la interpolación de las intensidades se arrastra utilizando los valores en una línea de bùsqueda y cuando los objetos y las líneas se rotan, la línea de bùsqueda no corta las orillas en los mismos puntos. Duff (1979) propuso, para solucionar este problema la interpolación independiente de intensidades.

4.7.3.5 Acabado de superficies paramètricas y parches

Los segmentos de superficie curva (también llamados parches) pueden usarse en lugar de los polígonos para modelar superficies curvas. Catmull (1975) propuso un mètodo para la producción de figuras sombreadas por computadora de las superficies curvas ya

mencionadas. Este método involucra tres pasos:

1. Establecer una correspondencia entre los puntos de la superficie y los "pixels".
2. Remover las partes ocultas de los parches
3. Calcular las intensidades de la luz

De acuerdo con Catmull, el algoritmo puede describirse como sigue:

"Si el parche es suficientemente pequeño para que su proyección cubra solamente un punto, entonces se calcula la intensidad del parche y se escibe en el "Frame-Buffer", de otra forma se subdivide el parche en pequeños subparches y se repite el proceso para cada sub-parche."

Algoritmo de Blinn

En la primera fase se determina la intersección de los bordes de la curva, la silueta de la orilla y la línea de búsqueda actual. Todos los cálculos de intersección se realizan utilizando el método de Newton-Raphson, para solución de ecuaciones. El proceso resultante es una lista de bordes para la línea de búsqueda actual.

En una segunda fase, los bordes se clasifican en orden del valor X de la línea de búsqueda. Para cada elemento de la figura, se genera la información Z de la superficie y puede representarse el sombreado requerido. Este algoritmo tiende a ser fuerte y reelevante excepto cuado las iteraciones de Newton fallan.

Algoritmo de Lane-Carpenter

En este algoritmo se deriva una aproximación de las superficies suaves por polígonos y se le da un acabado. Es una técnica de subdivisión similar al algoritmo de Catmull combinado con el algoritmo de polígonos de líneas de búsqueda. Este método puede resumirse como sigue:

1. Los parches se clasifican por los valores Y máximos posibles
2. Cuando se procesa cada línea de búsqueda los parches con los máximos valores Y posibles se subdividen hasta que:

   i. Ninguna pieza se encima con la línea de búsqueda y entonces se coloca en la lista de parches inactivos o
   ii. El parche está adentro de un conjunto de tolerancia por ser un polígono plano de cuatro lados.

Blinn (1982) propuso una solución general al problema de imágen para estas superficies. Las coordenadas del "pixel" se sustituyen por X y Y y la ecuación se resuelve para Z. Blinn mostró también como resolver el problema para la suma de varias distribuciones de densidad Gaussiana.

El método ha sido aplicado a la representación de mapas de densidad de electrones en estructuras moleculares.

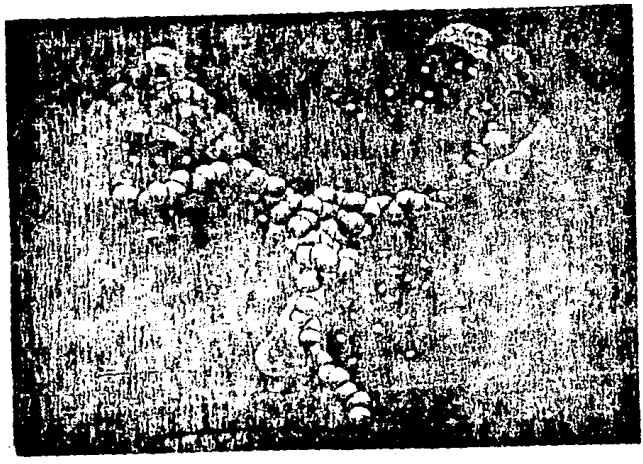


Fig. 4.15 Molécula por Nelson Max.

4.7.4 ALGORITMOS DE TRANSPARENCIA, TEXTURA, TRAZADO DE RAYO Y ANTIESCALONAMIENTO

4.7.4.1 TRAZADO DE RAYO

El trazado de rayo es una técnica antigua basada en la simulación numérica de la óptica geométrica. Los rayos de luz pueden trazarse desde una fuente de luz a través de sus trayectorias, hasta que alcancen al observador, sin embargo este es un punto de vista inadecuado porque sólo unos cuantos rayos llegan al observador. Esta es la razón por la que los primeros algoritmos que involucran el trazado de rayo, realizan el proceso en la dirección opuesta. Los rayos se trazan desde el observador hasta los objetos en la escena como se muestra en la figura.

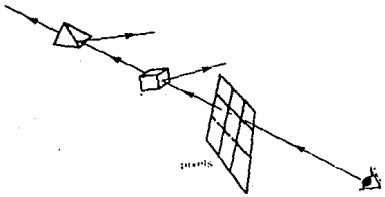


Fig. 4.16 Principio del trazado de rayo.

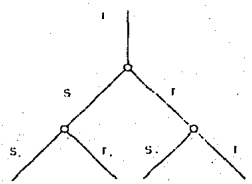
El primer uso práctico de la técnica de trazado de rayo en la animación por computadora, fue el de "Magi System" (Goldstein y Nagel 1971) en el cual se usó el algoritmo desarrollado por Appel (1968) .

El algoritmo de trazado de rayo consiste en el disparo de rayos, la intersección de estos rayos con los objetos en la escena y la información fotométrica requerida para colorear el "pixel".
Por cada superficie impactada por un rayo, puede generarse un rayo reflejado o no reflejado. El proceso debe aplicarse recursivamente para determinar qué otras superficies se intersectan.

Se construye un diagrama de árbol para cada "pixel".

Cuando el árbol se ha creado , se aplica una ecuación en cada nodo para calcular la intensidad. Con el proceso recursivo se obtiene la intensidad para el nodo actual cuando todos los subnodos se han evaluado.

La parte mas importante del algoritmo de trazado de rayo es el proceso de intersección. Whitted 1980 descubrió que hasta el 95% del tiempo del CPU se usa en este proceso.



I: Incidente
S: Reflexión especular
T: Transmisión

Fig. 4.17 Arbol de intersección.

Fragmentos de superficies paramétricas (parches)

Hay dos métodos básicos para el cálculo de la intersección entre fragmentos de superficies paramétricas y rayos.

1. El método recursivo de Whitted (1980) genera esferas acotadas por cada parche. Si una esfera acotada es tocada por un rayo, entonces el parche se subdivide y se producen esferas acotadas para cada subparche. El proceso continua hasta que la esfera acotada que fue intersectada, es mas pequeña que un mínimo dado o cuando ninguna otra esfera acotada es intersectada. Este algoritmo es similar al de Catmull (1974).

2. El método algebraico Kajiya (1982) transforma el problema de detectar una intersección de rayo-parche al problema de intersectar dos curvas algebraicas del sexto grado. La solución de este problema está dada por el teorema de Bezout.

Superficies de revolución y cilindros.

Para esta clase de objetos el problema puede reducirse a dos dimensiones. Para las superficies de revolución se define un plano cortante y se pasa a través del rayo paralelo al eje de revolución. En ambos tipos de objetos la solución del problema bidimensional de trazado de rayo se obtiene usando estructuras de árbol (Ballard 1981).

Las imágenes mas reales son aquellas originadas por el trazado de yo, sin embargo el método continua siendo muy caro desde el punto de vista de la aritmética de punto flotante. Con el desarrrollo de nuevo equipo y el refinamiento de los algoritmos pueden considerarse mejoras en la realización de esas técnicas.

4.7.4.2 TRASPARENCIA

Una superficie puede permitir que la traspase algo de luz desde atrás, dependiendo del tipo de material usado. La transmisión

puede ser reflejada, transparente o difusa (material traslùcido).
Fisicamente las tres leyes de Descartes indican las reglas relativas a la reflexiòn y la refracciòn de la luz.

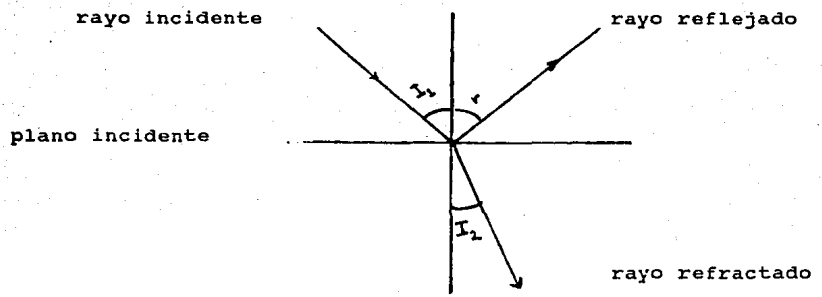
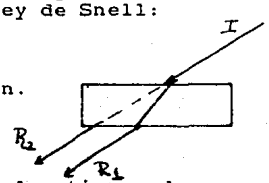


Fig. 4.18 Reflexiòn y refracciòn de la luz.

1. Los rayos reflejados y refractados están en el plano pasando a través del rayo normal y un rayo incidente.
2. El ángulo de incidencia i es igual al ángulo refractado r.
3. Para luz monocromática se aplica la ley de Snell:

$$n_1 \sin i_1 = n_2 \sin i_2$$

donde n_1 y n_2 son los indices de refracciòn.



El algoritmo de transparencia mas simple tiene dos grandes problemas :

1. La intensidad de la luz transmitida a través del objeto no toma en cuenta la profundidad del mismo.

2. Se ignora la refracciòn. Esto significa como se muestra en la figura que el rayo R_2 debe usarse en vez del rayo R_1.

Estos algoritmos simulan transparencia mediante la modificaciòn de la imagen de fondo. El color C de los "pixels" modificados se calcula (Newell y otros 1972) mediante la siguiente expresiòn :

C=tC1 + (1-t)C2

donde C1 es el color del "pixel" en la imágen del fondo, C2 es el

color aplicado al objeto transparente, y t es la transparencia del objeto (0 para completamente opaco, 1 para completamente transparente)

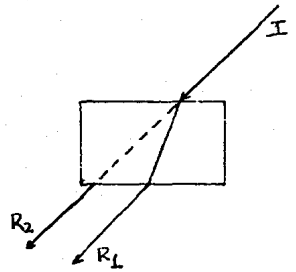


Fig. 4.19 Refracción a través de un objeto con profundidad

Esta técnica puede aplicarse en algoritmos de sombreado como los de Gourant y Phong. Por ejemplo , en un algoritmo de líneas de búsqueda donde un polígono que se encuentra en el fondo es transparente, el mas cercano de los otros polígonos que se encuentran atrás es visible. La intensidad I se calcula como la suma resultante de las intensidades individuales I_1 , I_2 calculadas para los dos polígonos. $I = t I_1 + (1-t) I_2$

Kay y Greenberg (1979) propusieron un algoritmo basado en el trazado de rayo, el cual toma en cuenta la refracción. Esto quiere decir que cada rayo se modifica en cualquier parte en la que incida sobre una nueva superficie transparente.

Para trazar la trayectoria de un rayo a través de un material grueso, Kay y Greenberg aplican el siguiente proceso:

1. Encontrar el vector unitario que define la dirección del rayo refractado mediante el uso de una aproximación de la ley de Snell.($\vec{u}$)

2. Encontrar la distancia D que el rayo debe recorrer dentro del material transparente.

3. Multiplicar $\vec{U}$ x D para obtener los valores de cambio Δx y Δy.

4. Sumar Δx y Δy a las localidades X y Y, esto nos da la localización visual del rayo tal como emerge del material transparente.

4.7.4.3 TEXTURA

Las imágenes generadas por computadora pueden tener un alto grado de realismo cuando se eliminan las superficies ocultas y se sombrean, sin embargo en muchos casos se ven artificiales porque aparecen demasiado suaves. Las imágenes de tazas de plástico o metálicas se ven muy reales, pero las imágenes de una naranja o del cuerpo humano no se ven asi.

Casi todas las superficies fisicas, de hecho, tienen una microestructura visible para el ojo humano. Esta microestructura llamada textura provee un alto grado de información acerca de la naturaleza de la superficie.
Catmull (1975), mostró con su algoritmo que las fotografias, dibujos o cualquier pintura, pueden ser transformada en parches. Sin embargo este enfoque falla cuando el número de puntos desplegados es menor que el número de elementos en la pintura.

Para resolver este problema Catmull sugiere el mapeo de areas sobre areas, en lugar de puntos sobre puntos, sub-dividiendo el parche y la pintura al mismo tiempo.

Blinn y Newell 1976, han extendido la técnica de Catmull introduciendo un método de filtro mas sofisticado, aplicando una deformación controlada al patrón.

Schweizer (1983) describe una técnica de textura que aproxima los cambios que se originan por la distancia y orientación sin dar exactamente una textura muy realista a la superficie.

Estos métodos son interesantes pero no simulan superficies ásperas. Schachter distingue dos categorias básicas de textura: la natural y la hecha por el hombre. Para la primer categoria se proponen patrones gaussianos al azar para modelar fenómenos naturales. Para las texturas hechas por el hombre, que son mas regulares se usan patrones modelados con una ecuación que envuelve una función prioritaria .

Blinn (1978) ha desarrollado un método el cual usa una función de textura, para alterar la dirección de la superficie normal de una manera leve antes de usarla en los cálculos de intensidad. Esta técnica fue utilizada previamente por Baston y otros (1975) para generar imágenes sombreadas.

4.7.4.4 SOMBRAS

Tal y como fue notado por Crow en 1978 los algoritmos para sombras requieren un tiempo considerable de cálculo y se usan rara vez en peliculas de animación por computadora.
Sin embargo la imagen debe tener sombras al menos que la fuente de luz esté localizada en el punto visual o que la iluminación sea muy difusa.

Antes de ver algunos algoritmos para la implementación de sombras se debe definir el término "sombra". Una sombra es la oscuridad producida por un objeto que intercepta la luz. Esta sombra se encuentra en el lado opuesto de la fuente de luz y como ya se mencionó solamente es visible cuando el punto visual se mueve de la fuente de luz.

El primer algoritmo para la generación de sombras fue sugerido en (1968) por Appel como una extensión de su algoritmo de líneas ocultas (1967). Las partes del segmento que están dentro de la sombra se determinan por el cálculo para todos los vértices, de la invisibilidad cuantitativa con respecto de la fuente de luz.

Bouknight y Kelly 1970 diseñaron un método que recorre un objeto línea por línea para determinar su visibilidad.

Otro camino para la generación de sombras es el uso del algoritmo de superficies ocultas para determinar cuales superficies están escondidas de la fuente de luz.Se usa un orden prioritario para tratar grupos de objetos.

Nishita y Kakamae 1974 han propuesto un método para la generación de sombras mediante el uso de un algoritmo de recorte de poliedros convexos como paso inicial, seguido de un método similar al usado por Bouknight y Kelly para eliminar superficies ocultas.
Crow diseñó un algoritmo basado en el principio del "volúmen de sombra". La superficie acotada del "volúmen de la sombra" se obtiene mediante la definición de todos los planos de la fuente de luz y los contornos de las orillas del objeto original.

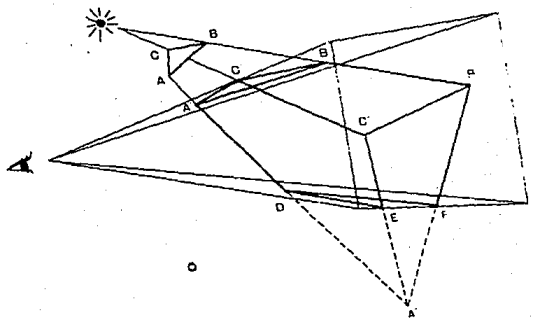


Fig. 4.20 Volúmen de sombras

Cuando el volúmen de las sombras se ha determinado, puede utilizarse cualquier algoritmo de superficies ocultas, para el despliegue.

La mejor opción es probablemente el algoritmo de generación de sombras de polígonos propuesto por Atherton y otros 1978. El método está basado en el algoritmo de "recorte poligonal" diseñado por Weiler y Atherton 1977 para eliminar superficies ocultas. Este algoritmo elimina todas las superficies que se encuentren bajo cualquier area poligonal única y sobre sus bordes.

Mediante el uso de algoritmos de eliminación de superficies ocultas para cada fuente de luz, pueden producirse imágenes sombreadas con varias fuentes de luz.

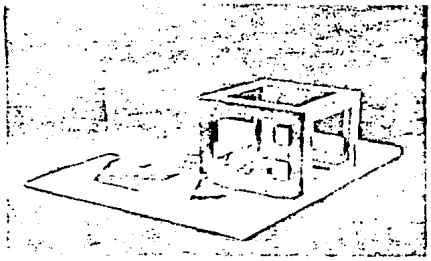


Fig. 4.21 Sombras de imágenes desplegadas con dos diferentes fuentes de luz.

4.7.4.5 ANTI-ESCALONAMIENTO

El fenómeno llamado escalonamiento es el mayor enemigo del animador por computadora. Este fenómeno ocurre cuando una señal de baja frecuencia aparece sobrepuesta como un escalón de la señal de alta frecuencia. Practicamente esto quiere decir que la resolución en el espacio del objeto es infinita cuando se compara con la resolución en el espacio de la pantalla.

Tipicamente los efectos del problema son los siguientes :

1. Se observa una especie de "escalera" en las lineas o en las orillas de dos superficies contrastantes.

2. Los objetos pequeños pueden desaparecer entre los puntos porque es posible que ninguna parte de ellos coincida con un punto muestra.

3. Existe un efecto de ruptura de linea en el lugar donde el ancho de una banda es del orden de un "pixel" y se pierde en el centro del mismo.

Las anomalias básicas son:

-Objetos que aparecen y desaparecen
-Las formas de los objetos cambian
-Efecto de parpadeo

Los efectos de escalonamiento fueron mencionados antes por Shoup (1973) y Catmull (1974) sin embargo el problema fue estudiado originalmente por Crow (1977) quien propuso tres clases generales de algoritmos de antiescalonamiento:

1. Cuando el problema de escalonamiento se debe a una baja resolución, se tiene la posibilidad de incrementar la resolución sin embargo este sistema tiene limitaciones muy severas porque incrementa el costo de la producción de la imágen.

2. La imágen puede generarse a alta resolución y filtrarse digitalmente. Este método elimina las altas frecuencias que son causa del escalonamiento. La técnica ha sido usada por Crow con excelentes resultados. Sin embargo requiere una copia de la imágen de alta resolución.

Las investigaciones en métodos de antiescalonamiento continuan progresando y se ha investigado en diferentes direcciones.

Piller, Gupta y Sprowll (1981) han propuesto un enfoque de procesos paralelos utilizando equipo especial. Fiume y otros 1982 han propuesto un algoritmo de conversión de búsqueda paralela.

Tukowski ha introducido un método de calcular antiescalonamiento a través del uso de transformaciones en donde el nucleo del filtro del anti-escalonamiento se aproxima mediante una función de la distancia perpendicular de un "pixel" a una linea. Bloomenthal 1983 ha presentado algoritmos para la detección y suavizado de las orillas y para el filtrado de una imágen de acuerdo con sus orillas.

Otras técnicas antiescalonamiento interesantes han sido aplicadas por Blinn, Newell (1976), Dungan (1978) , Feibush (1980).

Whitted (1980) y Roth (1982) también han estudiado el problema de escalonamiento con luz de alta intensidad.

Como se observa la mayoria de los métodos para la obtención de imágenes de alta calidad involucran algoritmos complejos, los

cuales utilizan mucho tiempo de máquina.

Esto significa que al ser implementados en una microcomputadora deben tomarse en cuenta las capacidades del microprocesador y los tiempo de respuesta para operaciones de punto flotante.

Como ya se mencionó en el capitulo dos, este tipo de imágenes no pueden animarse en tiempo real y se producen exclusivamente para su filmación y animación en película o en videograbadora.

C O N C L U S I O N E S

La animación realizada por medio de una computadora viene a revolucionar los métodos convencionales de producción de figuras, agilizando el proceso y ahorrando muchas horas de trabajo a las personas que trabajan en el area.

Otra de las grandes innovaciones que trae consigo el uso de la computadora es el hecho de que pueden producirse figuras con efectos especiales, nunca antes vistos, como los que se utilizan en las peliculas de ciencia ficción.

Entre algunas de las aplicaciones de la animación por computadora se encuentran: la elaboración de material didáctico educativo, el diseño de máquinas-herramienta, como apoyo en CAD/CAM, para el diseño de comerciales y peliculas y en medicina para el modelado de organos y moléculas, entre otras.

En México el area se encuentra aun en sus principios y las empresas que actualmente utilizan este tipo de efectos como pueden ser instituciones publicitarias e informativas en los medios de comunicación nacionales, importan los sistemas de E.U. o de otros paises que cuenten con este tipo de programas y equipo pero no realizan desarrollo localmente.

La animación por computadora puede dividirse en dos tipos:

- Animación de alta técnica.
  Este tipo de animación produce imágenes de alta calidad que se elaboran cuadro por cuadro, y que posteriormente se filman animándose por medios externos a la computadora (grabador de peliculas o videograbadora).

- Animación de bajo costo.
  Es la animación en tiempo real que se produce principalmente en microcomputadoras, como la que se observa en los videojuegos.

Cualquier sistema de graficación necesita de dispositivos especiales de entrada y salida como son los digitalizadores, las plumas de luz, el ratón y los graficadores. En el caso de la animación también serán necesarios este tipo de dispositivos y adicionalmente, dependiendo de la calidad de imágen que se desee se necesitarán tarjetas gráficas, monitores de alta resolución, chips especiales para artimética de punto flotante, y en algunos casos grabadoras de peliculas.

Los dos principales aspectos que se deben tomar en cuenta para la producción de la animación son : las capacidades de memoria de la

computadora y la velocidad de cálculo de operaciones.Esta última es importante porque para el modelado y construcción de imágenes será necesario aplicar transformaciones a las figuras y objetos, las cuales involucran una serie de operaciones aritméticas que deben efectuarse rapidamente.

Los requerimientos de memoria aumentan proporcionalmente a la calidad de la imágen que se desea obtener. Esto significa que si se tienen limitaciones de almacenamiento, las imágenes serán de baja calidad.

La segunda restricción se refiere al hecho de que no se pueden animar imágenes de alta calidad en tiempo real, ya que la velocidad de los dispositivos actuales no lo permite .

Se están diseñando chips especiales que agilizen el cálculo de las operaciones involucradas en el proceso de animación , uno de estos chips se llama Chip SLAM.

Existen diversos sistemas de animación por computadora que producen imágenes de alta calidad. Estos sistemas se han desarrollado en equipos medianos y grandes (como la computadora VAX). Esto se debe al hecho de que estos equipos cuentan con una mayor capacidad de memoria y velocidad de procesamiento, lo cual hace mas rápida la generación de imágenes.

Uno de los sistemas mas notables es el MIRA, el cual es un sistema orientado al artista. MIRA provee un lenguaje completo de animación con términos familiares al artista y con tipos de datos especiales como son el tipo actor y el tipo cámara entre otros.

La animación en microcomputadoras se encuentra seriamente limitada por las capacidades de la máquina (monitores, memoria y velocidad de procesamiento). Esto significa que las imágenes producida en la máquina serán de una calidad mediana, manejando un cierto número de colores límite (aproximadamente 256 colores simultáneo). Entre mas colores se manejen y mayor sea la resolución de la pantalla , nos vamos alejando de la producción de imágenes en tiempo real.

Una gran ventaja que se tiene en este caso es el bajo costo de los sistemas, ya que no hay comparación entre los costos de un equipo para animación de alta técnica y el de una microcomputadora.

Se espera que con el avance de la tecnologia y la mejora de los procesadores se pueda llegar a tener animación en tiempo real en una computadora desplegando imágenes de alta calidad.

El area está totalmente limitada por la técnica actual, y su

avance será paralelo al de la tecnologia, este hecho puede observarse diariamente al ver el diseño de nuevos chips, monitores y tarjetas gráficas mas potentes y de bajo costo.

También el aspecto de programación es de vital importancia por lo que es necesario el desarrollo de lenguajes especiales para anmacin que eliminen el uso de instrucciones poco apropiadas de los lenguajes de programacin. Estos lenguajes especiales deberan estar construidos de tal forma que el artista los pueda utilizar de una forma muy sencilla y familiar.

```
100 ' ** JUEGO DE PING PONG
110 '
120 SCREEN 0,1
130 KEY OFF
140 S$=SPACE$(37)
150 '
160 ON KEY(1) GOSUB 1490
170 ON KEY(3) GOSUB 1550
180 ON KEY(11) GOSUB 1610
190 ON KEY(14) GOSUB 1670
200 '
210 'EMPIEZA
220 KEY (1) ON
230 KEY(3) ON
240 KEY(11) ON
250 KEY(14) ON
260 '
270 ' POSICION INICIAL
280 '
290 Y1=12 : Y1SCR=Y1
300 Y2 = 12: Y2SCR=Y1
310 XOLD=20: YOLD=12
320 DELX=1 :DELY=0
330 PAD1=-1 : PAD2= -1
340 BCOUNT =0
350 '
360 COLOR 15,0
370 CLS
380 ' IMPRIME LOS SCORES
390 LOCATE 1,1 : PRINT PLAYER1;
400 LOCATE 1,35 :PRINT PLAYER2;
410 '
420 ' PAREDES LATERALES
430 COLOR 1,1
440 LOCATE 2,2 :PRINT S$;
450 LOCATE 24,2: PRINT S$;
460 COLOR 15,0
470 '
480 ' PROGRAMA PRINCIPAL
490 SND =0
500 '
510 ' ACTUALIZA PADDLE 1 SI ES NECESARIO
520 IF NOT (PAD1 OR PAD2) THEN 820
530 IF NOT PAD1 THEN 670
540 PAD1=0
550 ' BORRA EL VIEJO PADDLE
560 COLOR 15,0
570 LOCATE Y1SCR-1, 1,0 : PRINT " ";
580 LOCATE Y1SCR,1,0: PRINT " ";
590 LOCATE Y1SCR+1,1,0 : PRINT " ";
600 ' DIBUJA UN NUEVO PADDLE
```

```
610 Y1SCR=Y1
620 COLOR 1,1
630 LOCATE Y1SCR-1,1,0: PRINT " ";
640 LOCATE Y1SCR,1,0: PRINT " ";
650 LOCATE Y1SCR+1,1,0: PRINT " ";
660 '
670 ' ACTUALIZA PADDLE 2 SI ES NECESARIO
680 IF NOT PAD2 THEN 820
690 PAD2=0
700 ' BORRA EL VIEJO PADDLE
710 COLOR 15,0
720 LOCATE Y2SCR-1,39,0: PRINT " ";
730 LOCATE Y2SCR,39,0:PRINT " ";
740 LOCATE Y2SCR+1, 39,0: PRINT " ";
750 '
760 Y2SCR=Y2
770 COLOR 1,1
780 LOCATE Y2SCR-1,39,0: PRINT " ";
790 LOCATE Y2SCR,39,0: PRINT " ";
800 LOCATE Y2SCR+1,39,0 : PRINT " ";
810 '
820 'ACUTALIZA LA POSICION DE LA PELOTA
830 IF BCOUNT <> 0 THEN 860
840 X = XOLD +DELX
850 Y = YOLD+ DELY
860 BCOUNT = (BCOUNT+1) MOD 2
870 '
880 IF X<>2 THEN 1040
890 SND =500
900 DELX=1
910 IF Y<Y1SCR-1 THEN 980
920 IF Y>Y1SCR+1 THEN 980
930 IF Y=Y1SCR-1 THEN 960
940 IF Y=Y1SCR+1 THEN 970
950 DELY=0: SND=500 : GOTO 1200
960 DELY=-1: SND= 1000: GOTO 1200
970 DELY=1 : SND=1000: GOTO 1200
980 COLOR 15,0
990 LOCATE YOLD,3,0 : PRINT " ";
1000 LOCATE Y,2,0:  PRINT " *OUT";
1010 PLAYER2=PLAYER2+1
1020 GOTO 1320
1030 '
1040 IF X<>38 THEN 1210
1050 SND=500
1060 DELX=-1
1070 IF Y<Y2SCR-1 THEN 1140
1080 IF Y>Y2SCR+1 THEN 1140
1090 IF Y=Y2SCR-1 THEN 1120
1100 IF Y=Y2SCR+1 THEN 1130
```

```
1110 DELY=0 : SND=500 : GOTO 1200
1120 DELY=-1 : SND=1000:GOTO 1200
1130 DELY=1 : SND=1000: GOTO 1200
1140 COLOR 15,0
1150 LOCATE YOLD,37,0: PRINT " "
1160 LOCATE Y,35,0 :PRINT " OUT"
1170 PLAYER1=PLAYER1+1
1180 GOTO 1320
1190 '
1200 '
1210 IF Y=3 THEN DELY=1: SND=600
1220 IF Y=23 THEN DELY=-1 : SND=600
1230 '
1240 '
1250 COLOR 14,0
1260 LOCATE YOLD,XOLD,0: PRINT " ";
1270 LOCATE Y,X,0 : PRINT "*";
1280 XOLD=X : YOLD=Y
1290 IF SND THEN SOUND SND,2
1300 GOTO 480
1310 '
1320 '
1330 KEY(1) OFF
1340 KEY(3) OFF
1350 KEY(11) OFF
1360 KEY(14) OFF
1370 '
1380 '
1390 FOR I=800 TO 400 STEP -100
1400 SOUND I,1
1410 NEXT I
1420 '
1430 DEF SEG : POKE 106,0
1440 LOCATE 25,1
1450 PRINT "PRESIONE CUALQUIER TECLA PARA RESTAURAR";
1460 A$=INPUT$(1)
1470 GOTO 210
1480 '
1490 '
1500 Y1=Y1-1
1510 IF Y1<2 THEN Y1=24
1520 PAD1=-1
1530 RETURN
1540 '
1550 '
1560 Y1=Y1+1
1570 IF Y1>24 THEN Y1=2
1580 PAD1=-1
1590 RETURN
```

```
1590 RETURN
1600 '
1610 '
1620 Y2=Y2-1
1630 IF Y2<2 THEN Y2=24
1640 PAD2=-1
1650 RETURN
1660 '
1670 '
1680 Y2=Y2+1
1690 IF Y2>24 THEN Y2=2
1700 PAD2=-1
1710 RETURN
```

```
10 ' ** ESTE PROGRAMA PERMITE ANIMAR LA IMAGEN DE UNA HORMIGA LA CUAL VIAJARA
20 ' ** POR LA PANTALLA EN CUATRO DIRECCIONES, TIENE UN PEQUEÑO ALGORITMO DE
30 ' ** RETARDO  PARA DISMINUIR LA VELOCIDAD CON LA CUAL CAMINA
40 ' **
50 ' ** PONE EL VIDEO EN RESOLUCION MEDIA
60 SCREEN 1
70 CLS
80 KEY OFF
90 '
100 ' ** ARREGLOS EN LOS QUE SE GUARDARAN LAS CUATRO POSICIONES DE LA HORMIGA
110 DIM BUG0(54),BUG1(54)
120 DIM BUG2(54),BUG3(54)
130 '
140 ' ** DEFINE EL RECTANGULO EN EL CUAL APARECERAN LAS IMAGENES Y SERAN
150 ' ** CAPTURADAS DE EL MEDIANTE EL COMANDO GET. SE ESPECIFICAN EN ESTOS
160 ' ** TERMINOS PORUE 160,101 ES EL CENTRO DEL SCREEN.
170 '
180 X1=160-15: Y1=101-13
190 X2=160+15: Y2=101+13
200 '
201 ' ** SE DESPLIEGA CADA UNA DE LAS 4 IMAGENES SUCESIVAMENTE EN LA PANTALLA
202 ' ** SOLAMENTE SE GIRA LA FIGURA Y SE VA CAPTURANDO EN CADA UNO DE LOS
203 ' ** CUATRO ARREGLOS DECLARADOS. LA SUBRUTINA QUE SE LLAMA ES PRECISAMEN
204 ' ** TE LA QUE SE ENCARGA DE DESPLEGAR LA IMAGEN Y ROTARLA.
205 '
210 GOSUB 790
220 GET (X1,Y1)-(X2,Y2),BUG0
230 '
240 '
250 DRAW "A1":GOSUB 790
260 GET (X1,Y1)-(X2,Y2), BUG1
270 '
280 DRAW "A2":GOSUB 790
290 GET (X1,Y1)-(X2,Y2),BUG2
300 DRAW "a3": GOSUB 790
310 GET (X1,Y1)-(X2,Y2),BUG3
320 LINE (X1,Y1)-(X2,Y2),0,BF
330 CLS
340 '
341 ' ** PRINCIPIA LA FASE DE ANIMACION
342 '
350 LOCATE 25,1
360 PRINT "PRESIONE CUALQUIER TECLA PARA DETENERLA"
370 '
380 WHILE LEN(INKEY$)=0
390 '
391 ' ** SE ENCARGA DE QUE LA HORMIGA CAMINA EN DIRECCION VERTICAL DE ABAJO
392 ' ** HACIA ARRIBA EN EL EXTREMO DERECHO DE LA PANTALLA
393 '
400 X=260
410 FOR Y=110 TO 50 STEP -10
420   PUT (X,Y),BUG0
430   PUT (X,Y),BUG0
440 GOSUB 740
450 NEXT Y
```

```
460 '
461 ' ** SE ENCARGA DE QUE LA HORMIGA VIAJE DE DERECHA A IZQUIERDA EN
462 ' ** DIRECCION HORIZONTAL POR LA PARTE SUPERIOR DE LA PANTALLA
463 '
470 Y=30
480 FOR X=240 TO 60 STEP -10
490   PUT (X,Y),BUG1
500   PUT (X,Y),BUG1
510 GOSUB 740
520 NEXT X
530 '
531 ' ** SE ENCARGA QUE LA HORMIGA CAMINE EN DIRECCION VERTICAL DE ARRI
532 ' ** HACIA ABAJO POR EL EXTREMO IZQUIERDO DE LA PANTALLA
533 '
540 X=40
550 FOR Y=50 TO 110 STEP 10
560   PUT (X,Y), BUG2
570   PUT (X,Y), BUG2
580 GOSUB 740
590 NEXT Y
600 '
601 ' ** SE ENCARGA QUE  LA HORMIGA CAMINE EN DIRECCION HORIZONTAL DE
602 ' ** IZQUIERDA A DERECHA POR LA PARTE INFERIOR DE LA PANTALLA
603 ' **
610 Y=130
620 FOR X=60 TO 240 STEP 10
630  PUT (X,Y),BUG3
640  PUT (X,Y),BUG3
650 GOSUB 740
660 NEXT X
670 '
680 WEND
690 LOCATE 10,15
700 PRINT "ADIOS HORMIGUITA"
710 END
720 '
730 ' ** SUBRUTINA PARA RETARDAR LA VELOCIDAD DE VIAJE DE LA HORMIGA
740 '
750 'FOR I=1 TO 30 :NEXT I
760 RETURN
770 '
780 ' ** INICIALIZA
790 LINE (X1,Y1)-(X2,Y2),0,BF
800 DRAW "c3bm160,101"
810 '
820 ' ** DIBUJA LA CABEZA DE LA HORMIGA
830 DRAW "bu6re2u2h212g2d2f2r"
840 '
850 ' ** DIBUJA EL CUERPO Y LAS PATAS DE LA HORMIGA
860 DRAW "rf2e2f2b14f2"
870 DRAW "d2f2nd2h2d6"
880 DRAW "g2f2nd2h2g21"
890 DRAW "1h2g2nd2e2h2"
900 DRAW "u6g2nd2e2u2"
910 DRAW "e2h2g2br4e2r"
920 '
930 DRAW "BM160,101"
940 DRAW "P2,3 BU8 P1,3"
950 RETURN
```

```
100 ' secuencia de animacion
110 '
120 FOR I=0 TO 3
130 '
140  SCREEN 0,0,I,I
150  CLS
160  '
170 ' dibuja una caja
180 '
190 ' parte superior
200  L$=STRING$(2*I+1,205)
210  L$=CHR$(201)+L$+CHR$(187)
220  LOCATE 10-I,18-I: PRINT L$
230 '
240 ' mitad
250  L$=STRING$(2*I+1,32)
260  L$=CHR$(186)+L$+CHR$(186)
270  FOR J= 1 TO 2*I+1
280   LOCATE 10-I+J, 18-I : PRINT L$
290  NEXT J
300 '
310 ' parte inferior
320  L$=STRING$(2*I+1,205)
330  L$=CHR$(200)+L$+CHR$(188)
340  LOCATE 10+I+2, 18-I: PRINT L$
350 '
360 NEXT I
370 '
380 'secuencia de animacion
390 '
400 SCREEN 0,0,0,0
410 GOSUB 580
420 SCREEN 0,0,1,1
430 GOSUB 580
440 SCREEN 0,0,2,2
450 GOSUB 580
460 SCREEN 0,0,3,3
470 GOSUB 580
480 SCREEN 0,0,3,3
490 GOSUB 580
500 SCREEN 0,0,2,2
510 GOSUB 580
520 SCREEN 0,0,1,1
530 GOSUB 580
540 SCREEN 0,0,0,0
550 GOSUB 580
560 GOTO 380
570 '
580 ' subrutina para pausa
590 FOR K=1 TO 40 : NEXT K
600 RETURN
```

```
10 '*****************************
20 ' PROGRAMA QUE GIRA UNA GRAFICA
30 ' EN TRES DIMENSIONES
40 '*****************************
50 '
60 '
70 SCREEN 2
80 CLS:KEY OFF
90 WINDOW(-5,-5)-(5,5)
100 VIEW(120,20)-(500,198),,1
110 '
120 '
130 READ X3MIN,X3MAX,Y3MIN,Y3MAX
140 DATA -3,3,-3,3
150 '
160 ' ** FUNCION A GRAFICAR Y GIRAR
170 ' ** PUEDE CAMBIARSE OPCIONALMENTE
180 ' ** LA FORMULA
190 '
200 DEF FNF(X,Y)=COS(X)*COS(Y)
210 '
220 C=1
230 XSP=(X3MAX-X3MIN)/20
240 YSP=(Y3MAX-Y3MIN)/20
250 '
260  R$="S"
270  WHILE R$="S"
271  LOCATE 1,1
280    INPUT "ANGULO DE GIRO SOBRE EJE Z";SPIN
290    INPUT "ANGULO DE GIRO SOBRE EJE X";TIP
300 '  PONE ALTA RESOLUCION A LA PANTALLA
310 '
320 '
330    CLS:KEY OFF
340    GOSUB 630
350 '
360 '  **DIBUJA LA GRAFICA EN LA NUEVA
370 '  **POSICION
380 '
390    FOR X=X3MIN TO X3MAX STEP XSP
400       Y=Y3MIN: Z=FNF(X,Y)
410       PSET(FNX(X,Y,Z),FNY(X,Y,Z))
420       FOR Y=Y3MIN TO Y3MAX STEP YSP
430          Z=FNF(X,Y)
440          LINE -(FNX(X,Y,Z),FNY(X,Y,Z)),1
450       NEXT Y
460    NEXT X
470 '
480    FOR Y=Y3MIN TO Y3MAX STEP YSP
490       X=X3MIN: Z=FNF(X,Y)
```

```
500         PSET(FNX(X,Y,Z),FNY(X,Y,Z))
510         FOR X=X3MIN TO X3MAX STEP XSP
520            Z= FNF(X,Y)
530            LINE -(FNX(X,Y,Z),FNY(X,Y,Z)),1
540         NEXT X
550      NEXT Y
560 '
570 '
571 LOCATE 1,50
580      INPUT "DESEAS CONTINUAR ";R$
590 WEND
591 CHAIN "B:GRAFUN.BAS
592 END
593 '
594 '
600 '************************************
610 ' SUBRUTINA MATRIZ DE ROTACION
620 '************************************
630 RD=3.14159/180
640 '
650 A11=COS(SPIN*RD)
660 A12=SIN(SPIN*RD)
670 A13=0
680 A21=-SIN(TIP*RD)*SIN(SPIN*RD)
690 A22= SIN(TIP*RD)*COS(SPIN*RD)
700 A23= COS(TIP*RD)
710 '
720 DEF FNX(X,Y,Z)=A11*X+A12*Y+A13*Z
730 DEF FNY(X,Y,Z)=A21*X+A22*Y+A23*Z
740 '
750 RETURN
```

```
10 '
20 KEY OFF: SCREEN 2:CLS
30 DIM BOX(200)
40 '
50 CIRCLE(60,20),40
60 GET (0,0)-(100,40),BOX
70 '
80 '
90 PUT (0,150),BOX
100 '
110 FOR X=0 TO 500 STEP 20
120 PUT (X,150),BOX,PSET
130 NEXT
```

```
100 '
110 ' ESTE PROGRAMA GIRA UN CUBO EN TRES
120 ' DIMENSIONES. SPIN ES EL ANGULO DE
130 ' GIRO RESPECTO AL EJE Z Y TIP ES EL
140 ' ANGULO DE GIRO RESPECTO AL EJE X.
141 ' ESTE PROGRAMA FUNCIONA PARA CUALQUIER
142 ' FIGURA QUE SE DESCRIBA, SOLAMENTE HAY
143 ' QUE CAMBIAR LOS DATA CON LAS COORDENADAS
144 ' DE LA FIGURA DESEADA.
150 '
160 '
170 'ARREGLOS PARA LEER VERTICES,CARAS Y
180 'LINEAS DE LA FIGURA.
190 '
200 DIM VER(8,3),CAR(6,3),LIN(12,4)
210 CLS:KEY OFF
220 '
230 'PONE LA PANTALLA EN RESOLUCION MEDIA
240 '
250 SCREEN 1
260 '
270 '
280 'SE DECLARA UNA VENTANA PARA CONTROLAR
290 'LAS DIMENSIONES DEL CUBO DE ACUERDO A
300 'LAS COORDENADAS QUE SE MANEJAN
310 '
320 WINDOW (-100,-100)-(100,100)
330 '
340 '
350 'DATOS PARA LOS VERTICES
360 '
370 DATA 8
380 DATA -10,-10,10
390 DATA 10,-10,10
400 DATA  10,10,10
410 DATA -10,10,10
420 DATA -10,-10,-10
430 DATA 10,-10,-10
440 DATA 10,10,-10
450 DATA -10,10,-10
460 '
470 ' DATOS PARA LOS VECTORES PERPENDICULARES
480 ' A CADA UNA DE LAS CARAS DEL CUBO.
490 '
500 DATA 6
510 DATA 0,0,1
520 DATA 0,0,-1
530 DATA 1,0,0
540 DATA -1,0,0
550 DATA 0,1,0
560 DATA 0,-1,0
570 '
580 ' DATOS PARA DEFINIR LAS LINEAS ENTRE
590 ' VERTICE Y VERTICE.
600 ' SE DETALLA LINEA QUE VA DE V1 A V2
```

```
610 ' Y QUE ESTA DELIMITADA POR C1 Y C2
620 '
630 DATA 12
640 DATA 1,2,1,6
650 DATA 2,3,1,3
660 DATA 3,4,5,1
670 DATA 4,1,1,4
680 DATA 5,6,2,6
690 DATA 6,7,2,3
700 DATA 7,8,2,5
710 DATA 8,5,2,4
720 DATA 1,5,4,6
730 DATA 2,6,3,6
740 DATA 3,7,5,3
750 DATA 4,8,4,5
760 '
770 '
780 'PROGRAMA PRINCIPAL
790 '
800 '
810 R$="S"
820 WHILE R$="S"
830   CLS
840   LOCATE 9,1
850   INPUT "     ANGULO DE GIRO SOBRE EJE Z"; SPIN
860   INPUT "     ANGULO DE GIRO SOBRE EJE X"; TIP
865   PRINT:PRINT: PRINT "     RANGO DE TRASLACION -100 A 100":PRINT:PRIN
870   INPUT "     TRASLACION SOBRE EJE X ";TX
880   INPUT "     TRASLACION SOBRE EJE Y ";TY
890   INPUT "     TRASLACION SOBRE EJE Z ";TZ
900   RESTORE 370
910   GOSUB 1360
920   GOSUB 1720
930   GOSUB 1870
940   LOCATE 24,6
950   INPUT "QUIERES CONTINUAR ?";R$
960 WEND
970 '
980 '
990 'LAS RUTINAS QUE VIENEN A CONTINUACION
1000 'SON DE DEMOSTRACION. LA PRIMERA GIRA
1010 'EL CUBO DESDE 20 HASTA 180 GRADOS SOBRE
1020 'EL EJE Z. LA SEGUNDA LO GIRA SOBRE
1030 'EL EJE X.
1040 'SE DECLARA UN NUEVO VIEW PORT Y UNA
1050 'VENTANA PARA QUE EL CUBO SALGA MAS
1060 'GRANDE Y PARA MANTENER EL LETRERO QUE
1070 'APARECE EN LA PARTE SUPERIOR
1080 '
1081 TX=0:TY=0:TZ=0
1090 CLS
1100 VIEW (80,20)-(240,180)
1105 SPIN=30: TIP=20
1110 WINDOW(-20,-20)-(20,20)
1120 LOCATE 1,5
1130 PRINT "GIRO SOBRE EJE Z"
1140 FOR SPIN=20 TO 180 STEP 10
1150 RESTORE 370
```

```
1160    GOSUB 1360
1170    GOSUB 1710
1180    GOSUB 1870
1190 NEXT SPIN
1200 '
1205 FOR I=1 TO 50:NEXT I
1206 SOUND 100,6
1210 LOCATE 1,5
1220 PRINT "GIRO SOBRE EJE X"
1230 FOR TIP=0 TO 180 STEP 10
1240 RESTORE 370
1250    GOSUB 1360
1260    GOSUB 1710
1270    GOSUB 1870
1280 NEXT TIP
1290 CLS
1300 END
1310 '
1320 '*******************************
1330 ' SUBRUTINA DE LECTURA DE DATOS
1340 '*******************************
1350 '
1360 READ NV
1370 FOR I=1 TO NV
1380    FOR J=1 TO 3
1390       READ VER(I,J)
1400       VER(I,J)=VER(I,J)+T
1410    NEXT J
1420 NEXT I
1430 '
1440 FOR I=1 TO NV
1450    VER(I,1)=VER(I,1)+TX
1460    VER(I,2)=VER(I,2)+TY
1470    VER(I,3)=VER(I,3)+TZ
1480 NEXT I
1490 '
1500 '
1510 READ NC
1520 FOR I=1 TO NC
1530    FOR J=1 TO 3
1540       READ CAR(I,J)
1550    NEXT J
1560 NEXT I
1570 '
1580 READ NL
1590 FOR I=1 TO NL
1600    FOR J=1 TO 4
1610       READ LIN(I,J)
1620    NEXT J
1630 NEXT I
1640 '
1650 RETURN
1660 '
1670 '*********************************
1680 ' SUBRUTINA QUE CONTIENE LA MATRIZ
1690 ' DE ROTACION
1700 '*********************************
```

```
1710 '
1720 RD=3.14159/180
1730 '
1740 C11=COS(SPIN*RD)
1750 C12=0
1760 C13=-SIN(SPIN*RD)
1770 C21=-SIN(TIP*RD)*SIN(SPIN*RD)
1780 C22=COS(TIP*RD)
1790 C23=-SIN(TIP*RD)*COS(SPIN*RD)
1800 C31=COS(TIP*RD)*SIN(SPIN*RD)
1810 C32=SIN(TIP*RD)
1820 C33=COS(TIP*RD)*COS(SPIN*RD)
1830 '
1840 RETURN
1850 '
1860 '
1870 FOR I=1 TO NV
1880 '************************************
1890 ' SUBRUTINA QUE ROTA LA FIGURA Y
1900 ' LA DIBUJA
1910 '************************************
1920 '
1930  X=VER(I,1): Y=VER(I,2): Z=VER(I,3)
1940  VER(I,1)=C11*X+C12*Y+C13*Z
1950  VER(I,2)=C21*X+C22*Y+C23*Z
1960  VER(I,3)=C31*X+C32*Y+C33*Z
1970 NEXT I
1980 '
1990 FOR I=1 TO NC
2000  X=CAR(I,1): Y=CAR(I,2): Z=CAR(I,3)
2010  CAR(I,1)=C11*X+C12*Y+C13*Z
2020  CAR(I,2)=C21*X+C22*Y+C23*Z
2030  CAR(I,3)=C31*X+C32*Y+C33*Z
2040 NEXT I
2050 '
2051 '
2052 '*********************************************************************
2053 ' ALGORITMO DE LINEAS OCULTAS
2054 '*********************************************************************
2055 '
2060 CLS:KEY OFF
2070 FOR I=1 TO NL
2080 IF CAR(LIN(I,3),3)>0 THEN 2110
2090 IF CAR(LIN(I,4),3)>0 THEN 2110
2100  GOTO 2130
2110  PSET (VER(LIN(I,1),1),VER(LIN(I,1),2))
2120 LINE -(VER(LIN(I,2),1),VER(LIN(I,2),2))
2130 NEXT I
2140 RETURN
```

B I B L I O G R A F I A

1 Appel, A. Stein y Landstein J. " The Interactive Design of Three Dimensional Animation", Novena Asamblea Anual de la UAIDE, 1970

2 Baecker, Ronald. "Picture driven animation", Memorias de la conferencia "Spring Joint Computer", 1969

3 Baecker, R.M., "Interactive Computer Mediated Animation". Instituto de Tecnologia de Massachusetts, 1969

4 Barsky B y Beatty, J.C.," Local Control of Bias and Tension in Beta-Splines", ACM Graphics, Vol.2, No.2, 1983

5 Barsky, B., "A Description and Evaluation of Various 3-D Models", IEEE Computer Graphics and Applications, Vol.4, No.1, Enero 1984

6 Blinn,J.F., "Simulation of Wrinkled Surfaces", Computer Graphics, Siggraph 1977, Vol.12, No. 3

7 Blinn,J.F. y Newell, M.E., "Texture and Reflection in Computer Generated Images", Comun. de ACM, Vol.19, No.10, 1976

8 Booth, S Kellog y Kochanek, Doris. "Computer animate films and video", IEEE, U.S.A., Febrero 1983

9 Booth K.S. y MacKay S., "Techniques for Frame Buffer Animtion", Graphics Interface 1982

10 Boyse, J.W., "Interference Detection Among Solids and Surfaces", Comun. ACM, Vol. 22, No.1, Enero 1979

11 Cann, Peter. "Photographic animation of microcomputers graphics", BYTE Vol. 8, No. 11, U.S.A., Octubre 1983

12 Carl O. Rosendahl, "Designing for Computer Animation", Computer Graphics World, Vol. 6, No. 10, Octubre 1983.

13 Catmull, E., "The Problem of Computer Assisted Animation", Computer Graphics (Siggraph'78), Vol. 12, No. 3, Agosto 1978

14 Catmull,E.,"New Frontiers in Computer Animation",Cinematografo americano Octubre 1979

15 Catmull,E., "A subdivision Algorithm for Computer Display of Curved Surfaces", Reporte de ense|anza UTEC-CSC-74-133, Universidad de UTAH.1974

16 Chuang, Richard y Entis, Gleen. "3-D Shaded computer animation-step by step", IEEE, Vol.3, No. 9, Diciembre 1983

17 Craig, Reynolds, "Computer Animation with Scripts and Actors", Computer Graphics (Siggraph '82) , Vol 16, No.3, Julio 1982.

18 Crow, F.C., "Shaded Computer Graphics in the Entertainment Industry", Computer, Vol. 11, No. 3, Marzo 1978

19 Deily, D. "New principles for producing computer animated motion pictures" Escuela de Ingenieria Electrica, Universidad de Pensylvania, Diciembre 1968

20 Deken, Joseph. "Computer images. State of the art", Thames and Hudson, 1a ed., Italia 1983

21 Demetrescu, Stefan. "Moving pictures", Vol. 10, No. 12, U.S.A. Noviembre 1985

22 Doyne, J.Farmer, "Dimension, Fractal Measures, and Chaos Dynamic", Evolution of Order and Chaos, Springer-Verlag, 1982

23 Esterling D.M., " An Intersection Algorithm for moving parts". Modelado geometrico auxiliado por computadora (NASA Publicacion 2272), Abril 1983

24 Foley, J.D. y Van, Dam A. "Fundamentals of interactive computer graphics", Addison Wesley, U.S.A. 1982

25 Fox, David y Waite, Mitchelle. "Computer animation primer", Mc.Graw Hill, 1a ed., U.S.A. 1984

26 Gouraud,H., "Continuos Shading of Curve Surfaces", IEEE Trans. Computer, Vol. C-20, No. 6

27 Hackthorn, Ronald. "ANIMA-II: A 3-D color animation system", Computer Graphics, Vol. 11, No.2, Verano 1977

28 Halas J. y Manvell, R. "The technique of film animation", Hasting House New York 1959

29 Harris,Denis. "Computer graphics and applications", Chapman and Hall, 1a ed., Gran Breta|a 1984

30 Kerlow, Isaac. "The computer as an artistic tool", BYTE Vol. 9, No. 10, U.S.A., Septiembre 1984

31 Knowlton, K.C.A. "A computer technique for producin animated movies", AFIPS 1964, SICC Vol. 25, Spartan books New York

32 Knowlton, K.C. Computer-produced movies. Science 150, Noviembre 1985

33 Knowlton K.C., "A computer technique for the production of animated movies", Bell Telephone Laboratories Film

34 Leon, Gerardo "Un sistema para animacion de dibujos esquematicos por computadora", Tesis UNAM Facultad de Ciencias, Mexico

35 Levitan, Eli L. "Electronic imaging thechniques", Van Nostrand Reinhold Company, U.S.A., 1977

36 Magnenat, Nadia y Thalmann, Daniel. "Computer animation theory and practice", Springer-Verlag, Japon 1985

37 Magnenat, Nadia y Thalmann, Daniel "The use of high-level 3-D graphical types in the Mira animation system", IEEE, Vol. 3 No. 9, Diciembre 1983

38 Magnenat, Nadia y Thalman, Daniel. "Miranim: An extensible director-oriented system for the animation of realistic images", IEEE, Vol. 5, No. 3, Marzo 1985

39 Magnenat, Nadia y Thalman, Daniel. "Three-Dimensional computer animation: More an evolution than a motion problem", IEEE, Vol. 5, No. 10, Octubre 1985

40 Magnenat, Nadia y Thalman, Daniel. "Special cinematographics effects with virtual movie cameras", IEEE, Abril 1986

41 Mandelbrot, "The Fractal Geometry of Nature", W.H. Freemaan, 1982

42 Miura, Iwata y Tsuda, "An application of hybrid curve generation- cartoon animation by electronic computers", SJCC 1967

43 Morgan, Christopher y Waite, Mitchel. "Introduccion al microprocesador 8086/8088 (16 bits)", Mc. Graw Hill, 1a ed. Espa|a 1984

44 Myers, A.J., "An Efficient Visible Surface Program", Reporte tecnico de la Fundacion Nacional de Ciencia,Grant, No. DCR 74-00768A01. 1975

45 Myers, A.J., "A Digital Video Information Storage and Retrieval Systems", Tercera conferencia anual de computacion grafica y tecnicas interactivas, Siggraph, 1976

46 Newell, M., "The Utilization of Procedure Models in Digital Imge Synthesis, Ph.D. Universidad de Utah, 1975

47 Newton, Marcus. "Real time 3-D graphics for microcomputers", BYTE , Vol. 9, No. 10, U.S.A., Septiembre 1984

48 Nolan, J. y Yarbrough, L. "An on line computer drawing and animation system", IFIP Cong 1968, North Holland Pub. Co. Amsterdam

49 A procedure for detecting Intersections of Three Dimensional Objects". Asociacion de Maquinaria para computacion, Vol. 15, No. 3, Julio 1968

50 Pearson, Ron. "Animation magic with your IBM PC and PC jr.", Osborne Mc. Graw Hill, 1a ed., U.S.A. 1985

51 Phong Bui-Tuong, "Illumination for Computer Generated Pictures", Comun. ACM., Vol.18, No.6, 1975

52 Potmesill,M y Chakravarty, "Synthetic Image Generation with a Lens and Aperture Camera Model", ACM, Vol.1, No.2, Abril 1982

53 Reeves, W.T., "Particle Systems- A Technique for Modeling a Class of Fuzzy Objects", Computer Graphics, Siggraph 83

54 Revees,W.T., "Inbetweening for Computer Animation Utilizing Moving Point Constraints," Computer Graphics , Siggraph 81.

55 Reynold,"Computer Animation in the World of Actor and Scripts", Tesis, Instituto de Tecnologia de Massachusetts, 1978

56 Stephenson, R., "Animation in the cinema", A Zwemmer Limited London A.S. Barnes an Co New York 1967

57 Talbot, Peggy et al. "Animator: An on line two dimensional film animation system", Communications of the ACM, Vol. 14, No.4, Abril 1971

58 Talbot, P.A. "Animator- a System using the DEC 338 as an input terminal for movie making." Tesis de Maestria. Universidad de Pensylvania, Agosto 1969

59 Thalmann et al, "Drawing Bridges by computer" Computer-Aided design, Vol. 14, No.4, 1982

60 Thalmann et al, "Dream Flight: A Fictional Film Produced by 3-D Computer Animation" Computer Graphics '82, Publicaciones Online, Ltd.

61 Thalmann et al, "Controlling Evolution and Motion Using the CINEMIRA-2 Animation Sublanguage". Computer Generated Images, Tokyo, 1985.

62 Tsukasa, Norma y Kunii, Tosiyasu, "ANIMENGIME: An engineering animation system", IEEE, Vol. 5, No. 10, Octubre 1985.

63 Waite, Mitchel y Morgan Chirstopher, "Graphics primer for the IBM PC", Mc Graw Hill, U.S.A. 1983

64 Wein, M. y Burtynk,N. "A computer Animation System for the Animator", Decima asamble anual del UAIDE 1971

65 Wein,M. y Burtnyk, N.,"Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation Comun.ACM, Vol.19, No.10, 1976