UNIVERSIDAD LAS PALMAS GRAN CANARIA

ESCUELA UNIVERSITARIA DE INFORMATICA

Informática Aplicada a la Ciencia y a la Técnica

INFORMATÍCA GRÁFICA
(Conceptos Básicos)

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Juan Carlos Quevedo Losada
Máximo Juan Méndez Babey
Septiembre 1998

Depósito Legal: GC 826-1998
ISBN: 84-8416-774-7
Obra impresa en el servicio de reprografía del Dpto. de informática y Sistemas de la Universidad de Las Palmas de G.C.
Dirección: Edf. De Informática y Matemáticas
Campus de Tafira 35017
Las Palmas de Gran Canaria

CAPITULO 1

INTRODUCCION A LOS GRAFICOS POR COMPUTADOR

1.1 INTRODUCCION

Los gráficos por computador pueden definirse como la creación y manipulación de imágenes gráficas por medio de un computador. Sin embargo, esta corta definición no describe la variedad de aplicaciones y el impacto de este campo desarrollado por la ciencia de la computación. Ciertamente, los gráficos por computador comenzaron como una técnica para incrementar la información de la pantalla generada por un computador. Esta capacidad de interpretar y representar datos numéricos en imágenes ha aumentado significativamente la capacidad del computador de presentar información al usuario de una forma clara y comprensible. Grandes cantidades de datos se convierten rápidamente en diagramas de barras, diagramas de "tarta" y gráficos. Las pantallas gráficas también han mejorado nuestro entendimiento de sistemas complejos tales como la biología molecular, reafirmando lo dicho que una imagen vale más que mil palabras.

La mayor parte de los computadores de antes de 1980 tenían poca capacidad gráfica. La información se transmitía hacia y desde el computador en formatos numéricos y de textos. Existían dos razones por las cuales no se utilizaban las pantallas gráficas. Primero, el ambiente del tiempo compartido de los años 70 no era apropiado para los gráficos. Las necesidades de procesamiento de varios usuarios de gráficos simultáneos hubiesen producido efectos catastróficos en el tiempo de respuesta de otros usuarios del sistema. Una segunda razón para la carencia de gráficos era el coste exorbitante de estos sistemas de pantallas. En general, era demasiado caro justificar su compra.

Sin embargo, en los años 80, con la introducción de las estaciones de trabajo de microcomputadores cambió el aspecto global de las instalaciones de computadores. Estos computadores individuales tienen la ventaja de un bajo coste de los microprocesadores y de la memoria. Varios usuarios ya no deben compartir el mismo computador. De hecho, una sola estación de trabajo gráfica consiste, generalmente, de varios microprocesadores, cada uno con su propia memoria, y cada una diseñada para realizar una función alfanumérica o gráfica específica.

1.2 HISTORIA DE LOS GRAFICOS POR COMPUTADOR

Los computadores de los años 50 utilizaban dispositivos de hard-copy necesitando usuarios para examinar la salida alfanumérica. El computador Whirlwind construido en 1950 por MIT para investigar la estabilidad y control de los aviones fué probablemente el primer computador que utilizaba una pantalla de tubo de rayos catádicos (CRT) o el de un televisor. Anterior a este, los diseñadores de computadoras no habian pensado conectar este dispositivo de pantalla común a un computador.

El sistema de defensa aérea SAGE de los años 50 convirtió los **blips** del radar en

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

imágenes generadas por computador. Este sistema fue el primero en utilizar un lapiz óptico para seleccionar símbolos en el screen de la pantalla.

A finales de los 50 y comienzos de los 60, surgió la más profunda influencia de los gráficos por computador modernos: la tesis de Ivan Sutherland en 1963. Sutherland desarrolló el sistema de dibujo de líneas Sketchpad, que permitía al usuario dibujar señalando con un lapiz óptico los puntos en el screen. Entre los puntos, los sistemas gráficos podían dibujar líneas y construir polígonos. Los diagramas complejos se generaban repitiendo objetos simples. Estos sistemas gráficos interactivos eran los precursores de los sistemas de computadores modernos. Varios años después abandonaron MIT, Sutherland y Dave Evans cofundaron Evans and Sutherland, una firma dedicada al diseño y fabricación de sistemas de pantallas gráficas de propósito especial.

Los primeros CRTs tenían la capacidad de dibujar una línea recta entre dos puntos cualquiera del screen. Sin embargo, la línea deseparecía rápidamente de la pantalla y tenía que ser redibujada muchas veces por segundo. A comienzo de los 60, esto necesitaba memoria costosa para almacenar los puntos extremos de la línea y hardware costoso para volver a dibujar rápidamente la línea. En 1965, IBM introdujo el primer CRT producido en masa de este tipo.

En 1968, Tektronix introdujo el CRT de tubo de almacenamiento, el cual retenía permanentemente un dibujo hasta que el usuario lo borraba. Estos pantallas eliminaban la necesidad de las costosas memorias y el hardware de redibujo.

Los mediados de los 70 marcaron el comienzo de un periodo de reducción dramática tanto de la memoria como de las unidades lógicas de hardware. Esta reducción condujo a la actual proliferación de las pantallas de barrido de memoria intensiva sobre los cuales pueden crearse imágenes con sobras y colores de una forma interactiva.

Hemos examinado el papel del hardware en el desarrollo de los gráficos por computador. No olvidemos, sin embargo, que el software también jugó una parte crucial en su desarrollo. Ivan Sutherland es el primer dirigente de este campo, habiendo diseñado algunos de los mejores algoritmos y estructuras de datos sobre los que se basa los gráficos por computador. Los trabajos pioneros de Steven Coons (1966) y de Pierre Bezier (1972) con áreas curvadas condujeron la forma de la generación de computadores interactivos de imágenes tridimensionales realistas. En los últimos 10 años, muchas personas han desarrollado algoritmos importantes utilizados en los gráficos por computador. No de modo sorprendente, muchos de ellos estudiaron en la Universidad de Utah donde era profesor Ivan Sutherland.

1.3 VENTAJA DE LOS GRAFICOS INTERACTIVOS

Los gráficos por computador interactivo es el medio mecanizado más importante de producir y reproducir imágenes desde que se inventó la fotografía y la televisión; tambien tiene la ventaja que con el computador podemos hacer figuras de objetos abstractos. Con los gráficos interactivos, se está liberado del aburrimiento y frustación de mirar muchas páginas de textos en listados de impresoras de línea o en terminales alfanuméricas.

Mientras que las imágenes estáticas son una buena forma de comunicar información, las imágenes que varían dinámicamente son mucho mejor. Esto es cierto

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

especialmente cuando se necesita visualizar fenómenos que varían en el tiempo, tanto real (p.e. el giro de un avión en vuelo supersónico o la evolución de una cara humana desde que nace hasta que muere) como abstracta (p.e. el desarrollo de tendencias tales como el uso de la energía nuclear en USA o el movimiento de población, como funciones de tiempo). De este modo, una secuencia dinámica de tramas en la pantalla puede transportar frecuentemente movimiento suave o cambios de formas mejores que una secuencia de cambio lento de tramas individuales, especialmente cuando el usuario puede controlar la animación ajustando su velocidad, la parte total de la escena y la cantidad de detalles mostrados, y otros efectos. Por lo tanto, la mayoría de las tecnologías de los gráficos interactivos trata con técnicas hardware y software para movimientos dinámicos controlado por el usuario (los objetos pueden moverse y dar vueltas con respecto a un observador estacionario) y las actualizaciones dinámicas (el cambio actual de la forma, color u otras propiedades).

En resumen, los gráficos por computador interactivos nos permiten alcanzar la comunicación hombre-máquina utilizando una combinación sensata de texto con figuras estáticas y dinámicas. Esto produce una diferencia significante en nuestra habilidad de comprender datos y visualizar objetos reales o imaginarios. Para hacer mas eficiente la comunicación, los gráficos hacen posible mayor productividad, mayor calidad y resultados o productos mas precisos y menor costo de análisis y diseño.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

1.4 ALGUNOS USOS REPRESENTATIVOS DE LOS GRAFICOS POR COMPUTADOR

Los gráficos por computador se usan hoy dia en diferentes áreas de la indústria, negocios, gobiernos, educación, entretenimiento, y, más recientemente, en el hogar. La lista de las aplicaciones es enorme y aumenta rápidamente. A continuación se muestra unos ejemplos representativos de tales áreas.

- Imágenes interactivas en los negocios, la ciencia y la tecnología. Los gráficos hoy dia son utilizados más frecuentemente para dibujar representaciones en 2 dimensiones o 3 dimensiones de matemáticas, funciones físicas y económicas, histogramas, diagramas de barras y de "tarta", diagramas de inventários y producción, etc. Todos se utilizan para mostrar tendencias y conjuntos de datos de una forma significativa y corta para que aumente el entendimiento de fenómenos complejos.

- Cartografía. Los gráficos por computador se utilizan para la producción de representaciones altamente exactas sobre papel o película de fenómenos geográficos y naturales. Los ejemplos incluyen mapas geográficos, mapas de relieves, mapas de exploración para las perforaciones y la minería, cartas oceanográficas, y mapas de densidad de población.

- Diseño y Trazado Asistido por Ordenador. En el diseño asistido por ordenador (CAD), los gráficos interactivos se usan para diseñar componentes y sistemas de dispositivos mecánicos, eléctricos, electromecánicos y electrónicos. Estos sistemas incluyen estructuras (tales como edificios, plantas químicas y de energía), sistemas ópticos y redes telefónicas y de computadoras.

- Simulación y Animación. Se está popularizando las películas animadas producidas por computador del comportamiento variante en el tiempo de objetos reales o

simulados. Podemos estudiar no solamente figuras matemáticas sino tambien modelos matemáticos para tales fenómenos científicos como el flujo hidraúlico, la relatividad, las reacciones nucleares y químicas, los sistemas y órganos psicológicos, etc.

- Control de Procesos. Mientras un simulador de vuelo permite al usuario interactuar con una simulación de cualquier mundo real o artificial, otras muchas aplicaciones permiten que el usuario interactue con algunos aspectos de su propio mundo real. El estado de la pantalla muestra valores de datos para las refinerías, las plantas de energía y las redes de computadoras de sensores unidos a componentes peligrosos en el sistema; el operador responde entonces a las condiciones excepcionales. Los comandantes militares observan campos de datos (número y posición de vehiculos, movimiento de tropas, bajas, etc.) en pantallas de comandos de control y revizan sus tácticas como necesiten. Los controladores de vuelo en los aeropuertos observan información de identificación y estado generado por computador sólo con los "blips" de los aviones en su radar y de este modo pueden controlar el tráfico más rápido y exacto que con solo el radar de datos.

- Automatización de oficinas y publicaciones electrónicas. El uso de terminales alfanuméricas y gráficas crea y transmite información en las oficinas e incluso está aumentando rápidamente en los hogares. Pueden producirse tanto los documentos impresos tradicionalmente (hardcopy) como los documentos electrónicos (softcopy), los cuales no solo contienen el texto sino también tablas, gráficos y otras informaciones en 2 dimensiones.

- Arte y comercio. El arte y la publicidad por computador tienen el objetivo común de expresar un "mensaje" y atraer la atención del público con imágenes agradables. El computador gráfico abre un mundo enteramente nuevo para el artista, especialmente en la interacción entre el artista y el propio medio. El computador con gráficos ofrece al artista un conjunto de medios de trabajo completamente distinto a cualquier otro anterior. Por ejemplo, existen unos dispositivos que recuerdan los mandos del control de un avión y que permiten desplazar un punto de referencia sobre cualquier parte de la pantalla para introducir dibujos o figuras. Y, ¿qué decir de la posibilidad de emplear un "ratón" para introducir dibujos en el computador?. El "ratón" es la palabra de "argot" que designa un dispositivo semejante a una pequeña rosquilla, pero con la diferencia de que el agujero central lleva un punto de mira. El "raton" se mueve a lo largo del contorno de un dibujo, y este aparece en la pantalla del computador. Existen también lápices luminosos con los que puede dibujar el artista. Se emplea una pluma especial para introducir directamente la información en el computador. Además, es posible realizar efectos especiales increibles, llevados a cabo por el computador a partir de cualquier dibujo que se le introduzca en la pantalla.

1.5 ¿QUIEN USA LOS GRAFICOS POR COMPUTADOR?

Uno de los primeros usos de los gráficos por computador fue como ayuda al diseño. El diseño asistido por computador (CAD) y la fabricación asistida por computador (CAM) se encuentra en áreas de aplicaciones gráficas. Aquí, las pantallas de los computadores suministran un medio para automatizar dibujos de ingeniería, planos de arquitectura, instalaciones de arte comercial, o proceso de fabricación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

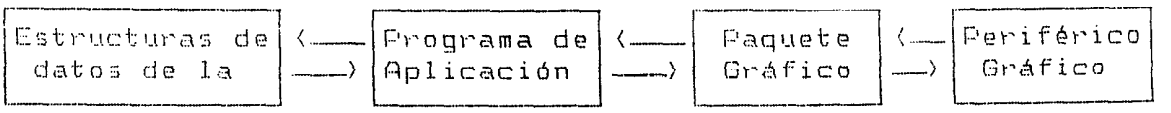
Los ingenieros que diseñan automóbiles, aviones y naves espaciales utilizan técnicas de CAD como ayuda en el trazado de las superficies de diseño.

Los circuitos electrónicos son diseñados con métodos CAD. Comenzando con símbolos gráficos que representan los diferentes componentes, un diseñador electrónico puede crear un circuito en la pantalla añadiendo componentes en cada momento. Con una pantalla de video de un edificio, un diseñador electrónico puede probar diferentes combinaciones para tomas electrónicas o sistemas de aviso de fuego.

Los arquitectos, tambien usan trazados de edificios producidos por métodos CAD como ayuda al diseño. Estos trazados se muestran de forma diferente. Los planos de pisos son util para diseñar habitaciones, colocación de puertas y ventanas, etc.

1.6 MODELO GENERAL DE UNA APLICACION GRAFICA

El modelo de una aplicación gráfica, puede expresarse de la siguiente forma:



Estructuras de datos de la Aplicación: Recogen los datos relevantes del problema, y se van modificando según las acciones o comandos del usuario. No tienen por que ser datos gráficos.

Programas de Aplicación: Son responsables de extraer los atributos necesarios de la estructura de datos para describir, de forma gráfica, al paquete gráfico, el tipo de "vista" que desea de sus datos.

Paquete Gráfico: Es responsable de convertir las descripciones proporcionadas por el programa de aplicación, en una o más imágenes visibles.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

CAPITULO 2

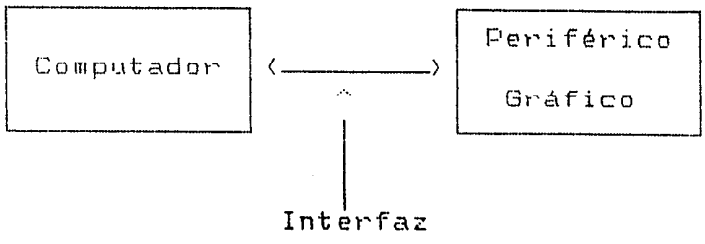
EQUIPOS GRAFICOS

2.1 INTRODUCCION

El propósito de los equipos gráficos es convertir la información gráfica contenida en la memoria del computador en imágenes visibles al usuario.

Si se trata de periféricos de entrada, su propósito es inverso: convertir acciones del usuario sobre una imagen en información gráfica procesable por el computador.

Es importante delimitar el reparto de responsabilidades funcionales entre el computador y el periférico, es decir, determinar el **interfaz** entre ambos:



Actualmente, este interfaz suele consistir de una línea de comunicación de baja velocidad (normalmente una línea serie RS-232-C entre 1200 y 9600 baudios) por la que circulan comandos o primitivas de relativo "alto nivel":

- dibujar una línea entre (X_1,Y_1) y (X_2,Y_2)

- escribir un texto "XXXXXX" con una posición, una orientación, etc.

- dibujar un polígono con los vértices en $(X_1,Y_1),...,(X_n,Y_n)$.

- etc.

Internamente al periférico, se distinguen dos partes diferenciadas:

- El Dispositivo Físico: convierte las señales eléctricas a imágenes.

- El Controlador: convierte las ordenes provenientes del computador, a señales eléctricas apropiadas al dispositivo. Es frecuente que una parte del controlador sea una memoria local al periférico.

La primera clasificación que es posible establecer es:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

- Periféricos de Salida:

    * pantallas de rayos catódicos de diversos tipos

    * trazadores gráficos

    * impresoras gráficas

    * grabadores de películas y de video

    * otras tecnologías

- Periféricos de Entrada:

    * tableta gráfica

    * mesa digitalizadora

    * lápiz sensible a la luz

    * ratón

    * ruedas, palancas, bola,teclas

    * teclado de diverso tipo incluido el teclado convencional

2.2 PANTALLAS DE REFRESCO

Son los periféricos gráficos más extendidos y los primeros históricamente (1950). A pesar de tener algunos inconvenientes tales como tamaño, peso, requisitos de potencia, altos voltages, etc, no han surgido aún ninguna tecnología capaz de desplazarlos ventajosamente, por lo que seguirán utilizandose masivamente en el futuro.

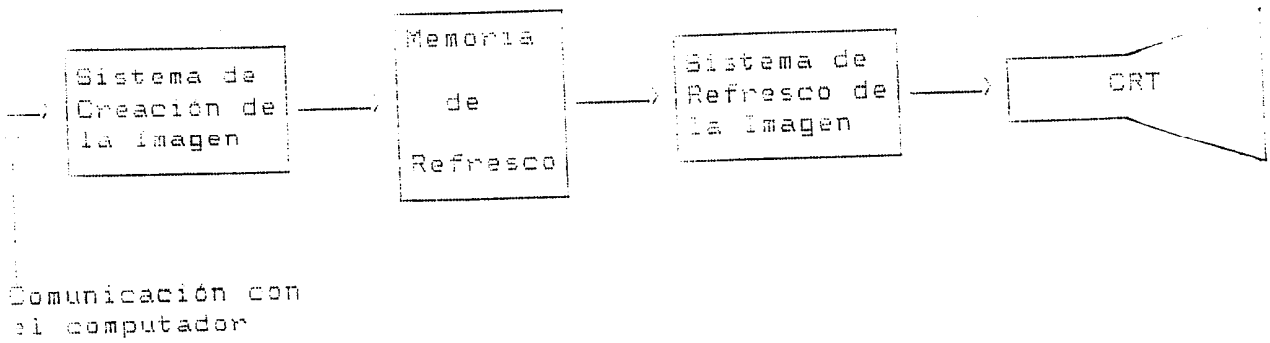
Se basan en un Tubo de Rayos Catódicos (CRT) cuyo haz de electrones "dibuja" la imagen sobre el fósforo que recubre internamente la pantalla.

La imagen ha de ser refrescada continuamente para que parezca estacionaria a los ojos del observador, dado que la luminosidad del fósforo decae exponencialmente en el tiempo, despues de haber sido excitado por el rayo de electrones. Por lo tanto, la imagen se tiene que conservar en una memoria de refresco a partir de la cual se dirige al pixel luminoso en sus ciclos de refresco. Cualquier cambio en el contenido de dicha memoria es reflejado inmediatamente en la imagen visible (el número de ciclos es, al menos, de 30 por segundo).

De esta forma, se consiguen imágenes que cambian dinámicamente en el tiempo, es decir, son modificables por el usuario, a diferencia de las conseguidas con otro tipo de periféricos.

El esquema general de una pantalla de refresco es:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



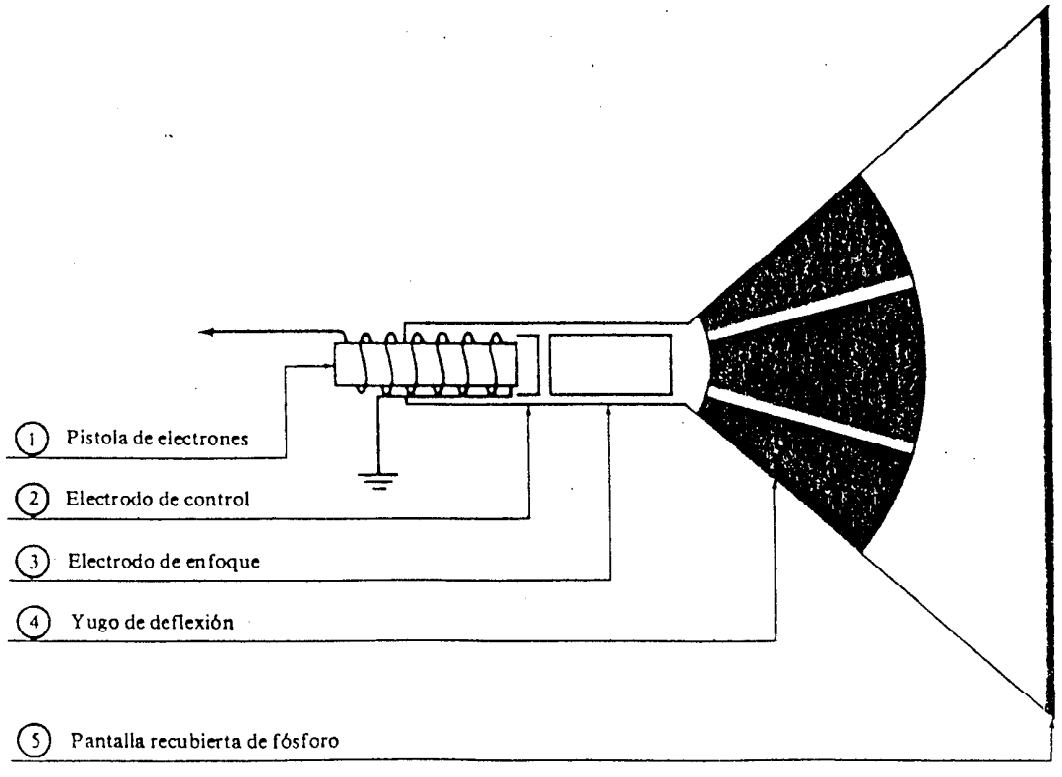
Existen dos tipos de pantalla de refresco, que difieren en el modo en que refrescan la imagen:

- **Pantallas Vectoriales o Caligráficas (Random-Scan)**: son aquellas que para dibujar una imagen en la pantalla siguen una secuencia de dibujo aleatoria.

- **Pantallas de Barrido (Raster-Scan)**: son aquellas que para dibujar una imagen en la pantalla siguen una secuencia fija de dibujo mediante barrido:línea por línea.

EL TUBO DE RAYOS CATODICOS

El tipo más común de dispositivo de salida por computador, de uso actual, capaz de exhibir salida gráfica, es el CRT de rastreo por barrido. En la siguiente figura se ilustra un CRT básico de rastreo por barrido y sus componentes principales:



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

- Pistola de Electrones: Está constituida por una serie de componentes (principalmente un calentador y un cátodo) que juntos provocan que los electrones se concentren al final de la pistola. Después, los electrones se aceleran debido a la aplicación de un campo eléctrico.

- Electrodo de Control (Rejilla de Control): Se utiliza para regular el flujo de electrones. El electrodo de control se encuentra conectado a un amplificador, que a su vez está unido a la circuitería de salida del computador, permitiendo así que este controle cuando se apaga o enciende el haz de electrones. Por lo tanto, permite controlar el número de electrones que forman el rayo, incluso puede suprimirlo por completo. Así, se puede conseguir varias tonalidades de brillo (grises) y desplazar el rayo de un lugar a otro sin dibujar.

- Electrodo de Enfoque (Sistema de Enfoque): Se usa para crear una imagen clara al enfocar los electrones en un haz estrecho. El electrodo de enfoque se utiliza para dicho fin cuando ejerce una fuerza electromagnética sobre los electrones del haz de electrones. De este sistema va a depender la resolución de la pantalla.

- Yugo de Deflexión (Sistema de Deflexión): Sirve para controlar la dirección del haz de electrones. Este yugo crea un campo eléctrico o magnético que doblará el haz de electrones conforme pase a través del campo. En un CRT convencional, el yugo está conectado a un generador de barrido o rastreo. Este generador envía una corriente oscilante tipo diente de sierra que, a su vez, provoca que el yugo de deflexión aplique un campo magnético variante a la ruta del haz de electrones. El potencial de voltage oscilante ocaciona que el haz de electrones se mueva a través de la pantalla del CRT en un patrón regular.

- Pantalla recubierta de Fósforo: Se recubre de fósforo la superficie frontal interior de todo CRT. Esta superficie está recubierta con cristales especiales denominados Fósforo, que tienen una propiedad única que permite a todo el sistema funcionar. Los fósforos brillan cuando inciden en ellos un haz de electrones de alta energía. Continúan brillando durante un periodo de tiempo determinado (el tiempo exacto y el color son únicos para cada tipo de fósforo) después de estar expuestos al haz de elctrones. El brillo que despide el fósforo durante la exposición al haz de electrones se conoce como **fluorescencia**; el brillo continuo emitido después de que el haz se elimina se llama **fosforecencia** y la duración de la fosforecencia recibe el nombre de **persistencia del fósforo**. Todos los fósforos tienen una vida limitada, que es una función tanto de la cantidad de tiempo que está expuesto al haz de elctrones como de la intensidad de dicho haz.

2.2.1 Pantallas Vectoriales o Caligráficas

Como ya se dijo, son aquellas que para dibujar una imagen en la pantalla siguen una secuencia de dibujo aleatoria. Por ello, la memoria de refresco está estructurada como una secuencia lineal de instrucciones gráficas que son interpretadas por un procesador especializado llamado **Procesador de Visualización** (Display Processing Unit" o DPU).

El tipo de instrucciones que ejecuta este procesador son en esencia órdenes que controla el rayo electrónico del tubo (nótese que el rayo electrónico no sigue una secuencia

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

fija en la pantalla, sino aleatoria, en función de la secuencia de instrucciones gráficas en la memoria de refresco):

- cargar coordenada X

- cargar coordenada Y

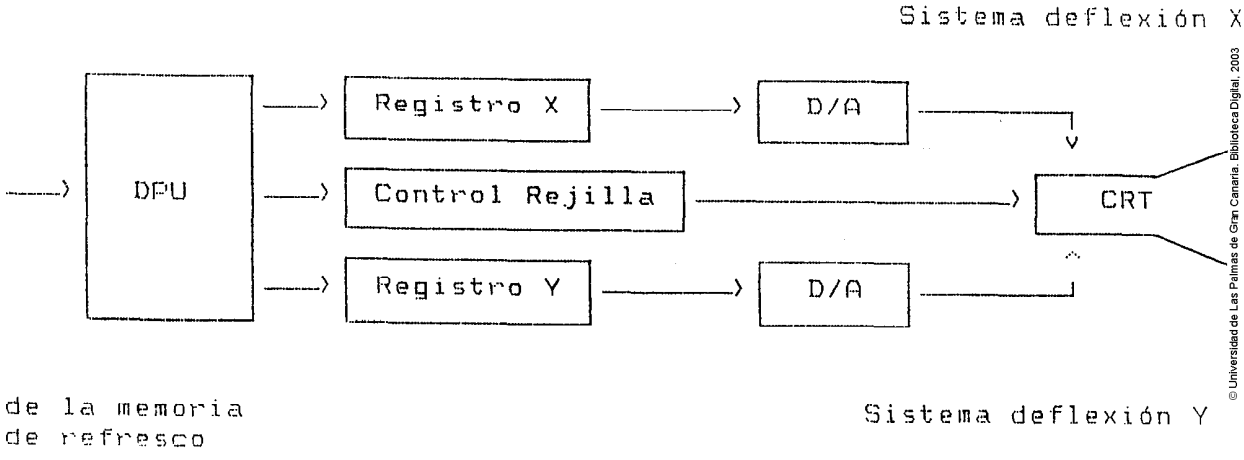
- desplazar rayo sin dibujar, desde posición actual, a X,Y

- desplazar rayo dibujando, desde posición actual, a X,Y

- salto incondicional a una posición de memoria.

Esta última instrucción sirve, en principio, para crear un bucle sin fín en la memoria de refresco, de forma que el DPU repita continuamente la misma imagen. Tambien puede usarse para estructurar el conjunto de instrucciones gráficas en áreas no contiguas de la memoria de forma que sea más fácil su modificación.

El interfaz entre el DPU y el tubo, es de la siguiente forma:



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Un buen sistema de deflexión, necesita un promedio de 5 a 20 microsegundos para deflectar el rayo (tarda más cuanto más largo es el desplazamiento) = = = > 1600 a 6600 vectores en cada ciclo. Un promedio normal, en el que predominan los vectores cortos, puede considerarse de 5000 vectores/ciclo (150000 vectores/seg.). Este sería el tamaño máximo de un dibujo sin que se produzca parpadeo.

Generador de Caracteres

No es eficiente, en cuanto a espacio, que la memoria de refresco contenga instrucciones para dibujar un caracter cada vez que este sea necesario.

Casi todas las pantallas incorporan en ROM o en RAM dichas instrucciones. Cada vez que sea necesario dibujar un caracter, la DPU efectua un "salto a subrutina" a la secuencia correspondiente de instrucciones.

Las instrucciones de la memoria de refresco que provocan dichos saltos, es por ejemplo:

dibujar_cadena_caracteres, "ABCDE....."

A veces es posible especificar otros parámetros tales como:

- tamaño del caracter

- ángulo de orientación del camino.

- orientación de los caracteres respecto al camino.

Los caracteres predefinidos en ROM o RAM, se pueden almacenar de dos formas posibles:

- secuencia de trazos

- matriz de puntos

2.2.2 Pantallas de Barrido

Como ya se dijo, son aquellas que para dibujar una imagen en la pantalla siguen una secuencia fija de dibujo mediante el barrido de línea por línea. Para ello, la memoria de refresco está dividida en N*M celdas, estructuradas como una matriz, cada celda correspondiendo a un punto de la imagen de la pantalla. La imagen está, a su vez, dividida en N*M puntos (N líneas horizontales, cada una con M puntos) y existe una correspondencia biunívoca fija entre celdas de la memoria y puntos de la imagen.

A veces nos referiremos a los puntos de la imagen (y, por extensión, a las celdas de la memoria) como **pixels.**

La información contenida en cada celda es la necesaria para iluminar adecuadamente el punto correspondiente:

- en el caso más sencillo (pantallas monocromáticas) consiste en un código binario:

  * 1 = iluminar punto

  * 0 = no iluminar punto

- a veces es, un índice en un cierto rango (p.e. 0 a 15, 0 a 255) que expresa:

  * tonalidad de gris

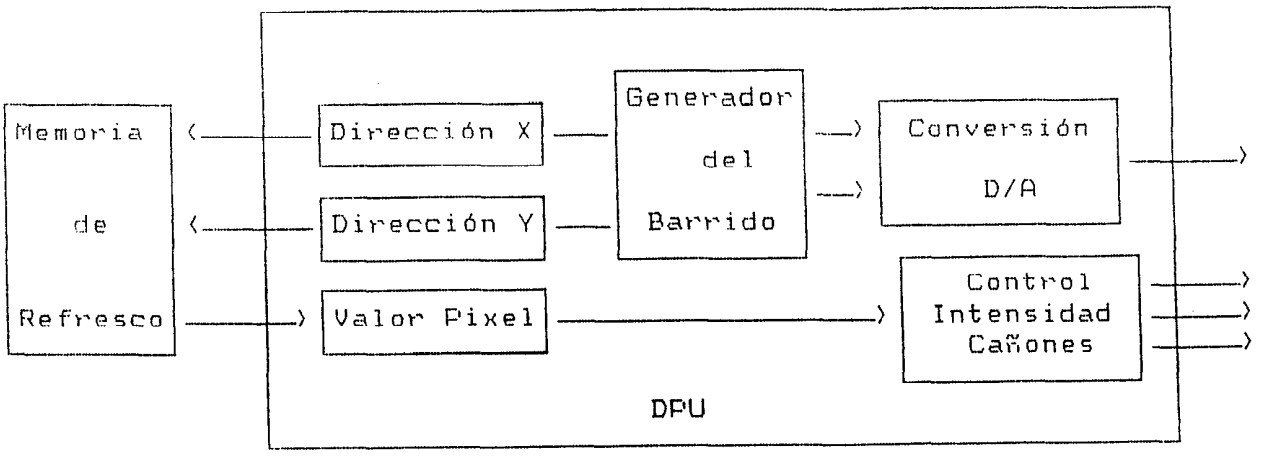
  * indice de color o gris, cuya descripción se encuentra en una memoria auxiliar, estructurada como una tabla de acceso directo, accedida por dicho índice. Esta tabla puede cambiarse por programa

- en el caso más complicado, cada celda son triplas de valores que definen un color

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

como intensidades de los colores básicos: rojo, verde y azul.

Sistema de Refresco de la Imagen



El DPU, aquí llamado procesador de visualización de barrido, recorre la memoria de refresco en una secuencia fija; línea por línea de arriba a abajo y, de izquierda a derecha, dentro de cada línea. Al llegar al final de cada línea, el rayo retrocede sin pintar hasta el comienzo de la siguiente. Al terminar la última, el rayo retrocede al comienzo de la primera. El tiempo de retroceso (1.3 milisegundos) puede utilizarse para producir cambios en la memoria de refresco.

Los cambios podrían también hacerse concurrentemente con el refresco. En ese caso, durante una fracción de segundo, suficiente para distraer al observador, se vería un dibujo semiactualizado.

La frecuencia de los refresco suele ser:

* 30 por segundos (30 Hz): algo de parpadeo

* 30 Hz entrelazado

  1/2 ciclo líneas impares
  1/2 ciclo líneas pares

* 60 Hz no entrelazado: no parpadeo.

Cuanto más alta sea la frecuencia, menos tiempo hay para dibujar cada pixel.

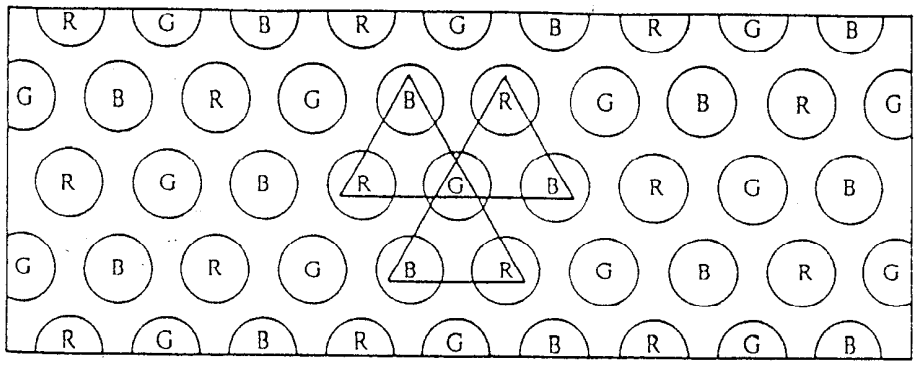
Pantallas de Barrido en Color

La técnica para conseguir color es similar a la empleada en los receptores de TV.

El fósforo que recubre la pantalla internamente, está dispuesto en triadas de puntos

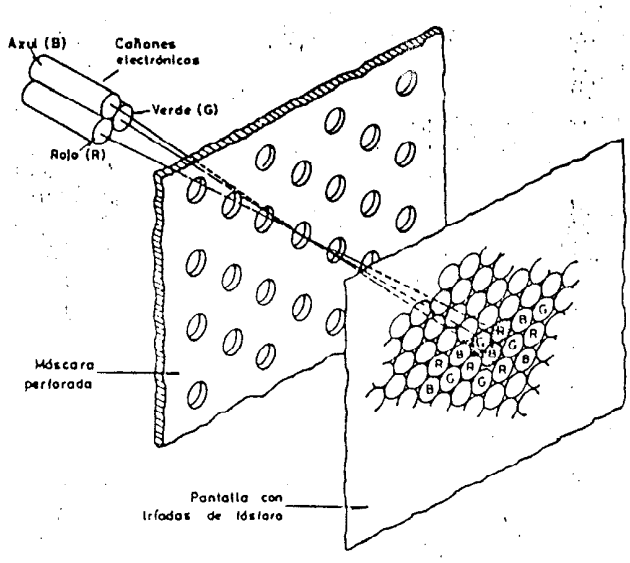
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

de colores rojo, verde y azul:



Se dispone de tres cañones de electrones que se deflectan conjuntamente, cuya intensidad es controlable por separado.

Por medio de una máscara perforada situada algo antes de alcanzar el fósforo, se logra que cada cañón incida exactamente en una de los puntos de la triada. La retina es capaz de "fundir" los tres colores resultantes (uno en la gama del rojo, otro en la del verde y otro en la del azul) obteniendose la sensación de un color mezclado. Variando la intensidad de los tres cañones en todas sus posibilidades, se obtienen amplias gamas de colores diferentes.

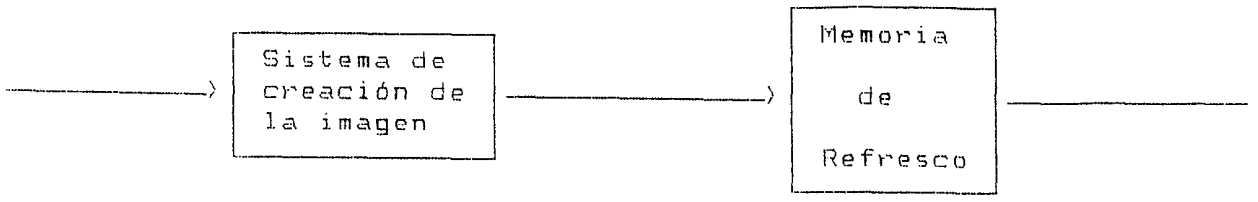


Se consigue menor resolución que en pantallas monocromáticas. Sin embargo, hoy día son posibles resoluciones de 0.25 mm por triada, lo cual permite 1024 puntos en 26 cm de lado. En los receptores de TV la resolución es del orden de 0.60 mm por triada.

Los problemas técnicos de convergencia de los tres rayos al tiempo sobre la triada, están actualmente resueltos.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Sistema de Creación de la Imagen



Las primitivas gráficas provenientes del computador donde se ejecuta la aplicación, normalmente tienen un aspecto de más alto nivel que "iluminar un pixel con un cierto color". Esta primitiva puede existir, pero lo más cómodo para la aplicación son del tipo:

- dibujar línea o conjunto de líneas

- dibujar panel sólido

- dibujar texto

- etc.

Esto quiere decir que el sistema de creación de la imagen ha de convertirlos a puntos antes de escribirlos en la memoria de refresco. Esta conversión (scan_conversion) es normalmente lenta y suele constituir el cuello de botella que impide lograr una alta interactividad en los sistemas de barrido. (Gracias a esta conversión, el interfaz computador-pantalla puede ser muy similar al de una pantalla vectorial).

Por ejemplo, para convertir una línea, hay que calcular qué pixels son los más próximos a la dirección de la línea, e iluminarlos (en el próximo capítulo se verán algoritmos para ello). Si el sistema de creación de la imagen es una CPU convencional (un microprocesador), la velocidad de conversión es de unos pocos microsegundos por pixel.

Con circuitería especial este tiempo puede quedar bien por debajo del microsegundo. Hoy día, se alcanzan, en los equipos de más alto rango, velocidades de hasta 8 nanosegundos/pixel. Esto quiere decir que la conversión a pixels no es ya el cuello de botella en la modificación de la imagen. Se puede recrear toda la imagen en menos tiempo de lo que dura el ciclo de refresco.

La estructura de la memoria de refresco en las pantallas de barrido es causa de algunas ventajas y dificultades que las hace diferentes de las pantallas vectoriales.

La principal ventaja es que puede mostrar paneles sólidos (en blanco y negro o color). La segunda ya ha sido mensionada: no hay límite a la cantidad de información gráfica que pueden mostrar.

Uno de los inconvenientes es el efecto "escalera" en las líneas rectas que no siguen

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

exactamente la horizontal, vertical o diagonal. Existen técnicas para disminuir este efecto.

Los demás inconvenientes tienen que ver con la interactividad (modificación rápida de la imagen):

- el desplazamiento de objetos ha de hacerse borrando y reescribiendo el objeto en n distintas posiciones. Esto lleva consigo

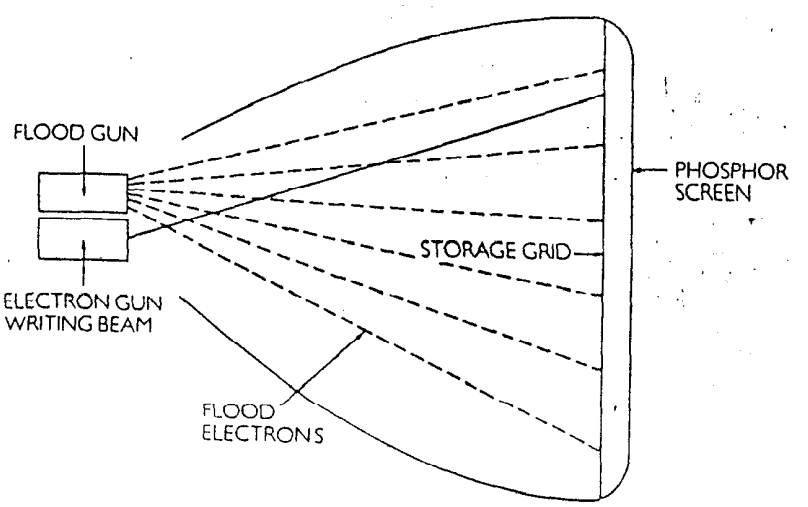
  * lentitud

  * destrucción del dibujo de fondo, salvo que se tomen precauciones.

2.3 PANTALLAS DE ALMACENAMIENTO

También conocidas como DVST (Direct View Storage Tube). Este tipo de pantallas no necesitan refrescar la imagen periódicamente ni, en consecuencia, utilizar memoria de refresco.

La propia pantalla almacena el dibujo en forma de distribución de cargas eléctricas:



La superficie tiene inicialmente una carga negativa. Al incidir el rayo de dibujo, los electrones son desalojados y atraidos por la rejilla colector cargada positivamente.

Al no ser conductor el material fosforecente, permanece el defecto de carga negativa que representa el dibujo.

El cañón de inundación no es enfocado ni reflectado sino que cubre uniformemente la pantalla

- al incidir en zonas no pertenecientes al dibujo (carga negativa) los electrones son repetidos y la pantalla permanece en negro

- al incidir donde hay dibujo (carga más positiva), son atraidos y provocan fosforecencia.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

El borrado se realiza mediante una escritura "total", aplicando un voltage positivo a la superficie de almacenamiento. Seguidamente, se restablece el voltage negativo para iniciar un nuevo dibujo con la pantalla en negro. Durante un instante se produce un "flash" con la pantalla totalmente en blanco. Estos "flashes" pueden llegar a ser molestos si el borrado se realiza frecuentemente.

Disminuyendo la intensidad del rayo de escritura, se pueden conseguir excitaciones del fósforo que no desalojan electrones y, por tanto, no crean dibujos permanentes. Así, es posible combinar una cierta capacidad de dibujo de refresco (con su memoria asociada) para las partes más interactivas, con una gran capacidad de almacenamiento para el resto.

Características a destacar

* permiten ampliar un dibujo, pero no borrar partes selectivamente (salvo si combinan almacenamiento con refresco)

* la misma resolución que la alcanzable por pantallas vectoriales (generalmente 4096 * 4096).

* bajo contraste, debido a los electrones de inundación

* los circuitos de deflexión no necesitan ser tan rápidos como los de una pantalla vectorial

* gran cantidad de información sin problemas de parpadeo

* sólo líneas monocromáticas

* menor coste que las pantallas vectoriales y que las primeras pantallas de barrido, tecnología en desuso

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

2.4 RESUMEN COMPARATIVO DE LOS TIPOS DE PANTALLA

El siguiente cuadro, resume las características más relevantes de cada tipo de pantalla:

	PANTALLAS VECTORIALES	PANTALLAS ALMACENAMIEN.	PANTALLAS BARRIDO
CANTIDAD DE INFORMACION	Limitada (5000 vectores)	Ilimitada	Ilimitada
BORRADO SELECTIVO	Si	No	Si
PANELES SOLIDOS	No	Si	Si
COLOR	Pocos y Fijos	No	Total flexibilidad
RESOLUCION	Alta	Alta	Media
CANTIDAD DE LINEAS	Alta	Bajo Contraste	Efecto "Escalera"

Conviene notar que, pese a las diferencias tecnológicas entre unas pantallas y otras, el interfaz que presentan hacia el computador de la aplicación es muy similar. Al menos en las primitivas referentes a dibujar líneas y caracteres (y por extensión, curvas que, en última instancia, se pueden aproximar con segmentos de rectas). Las diferencias surgirían al tratar de rellenar paneles sólidos. Estos podrían simularse con rayados, al coste de dibujar más información. Ello proporciona la base para introducir un software estandard, entre la aplicación y los periféricos gráficos, que hagan a aquella independiente del tipo concreto utilizado de estos últimos.

2.5 DISPOSITIVOS DE ENTRADA/SALIDA

La mayor parte de los terminales de pantalla se suministran al usuario con un teclado alfanumérico con el cual se comunica con el programa de aplicación. Sin embargo, para muchas aplicaciones, el teclado es inconveniente o inadecuado. Por ejemplo, el usuario puede desear seleccionar un símbolo de entre un conjunto de ellos en el screen para borrarlo. Si cada símbolo está etiquetado, podemos borrarlo indicando su nombre; sin embargo, posicionando el símbolo, se puede borrar y más rapidamente, eliminando el campo que etiqueta al símbolo.

Surge otro problema si el usuario tiene que añadir líneas o símbolos a la imagen de la pantalla. Aunque puede identificar la posición tecleando sus coordenadas, puede hacerlo mejor posicionandolo en la pantalla.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

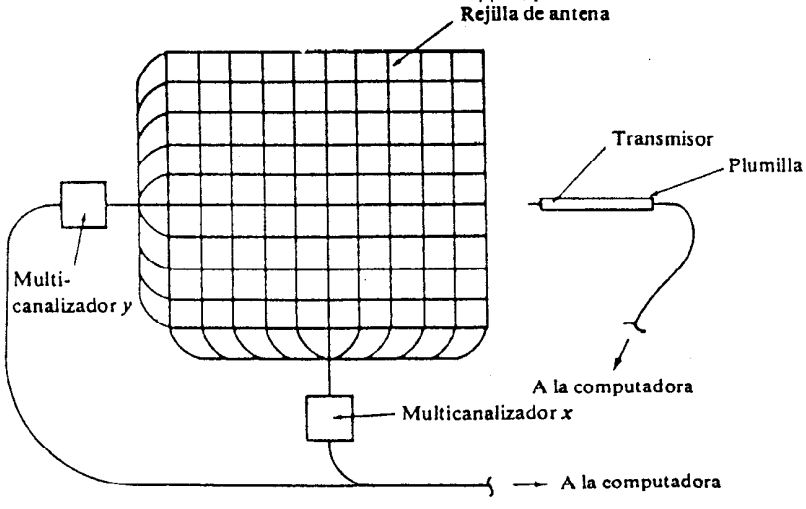
2.5.1 Dispositivos de Entrada

Los dispositivos de entrada gráfica son necesarios por dos razones principales: entrar datos al computador y ayudar en esta manipulación. Generalmente estas dos funciones se combina con facilidades software. A continuación veremos varios dispositivos de entrada:

Tableta Gráfica

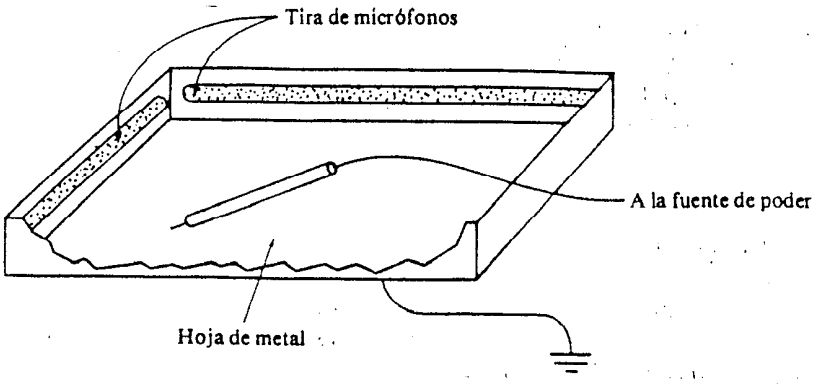
Existen varios tipos de Tabletas Gráficas. El tipo básico consiste de un área lisa con una serie de cables paralelos en las direcciones X e Y. Conceptualmente es muy similar a un pedazo de papel gráfico. Algunos tipos de agujas conectadas a la tableta pueden detectar una coordenada individual X,Y mediante la señal dada por la intersección del cable. Las señales analógicas recibidas de la aguja se procesan mediante lógica de decodificación produciendo posiciones digitales X e Y. Normalmente, estas coordenadas son absolutas ya que el usuario define el origen en la tableta. Normalmente la aguja no toma contacto con la tableta pudiendo estar aproximadamente a 2 centímetros de ella. El software es útil para ayudar en la definición de áreas, curvas, etc.

La pluma y el tablero magnético están compuestos de una rejilla bidimensional de alambre y una plumilla que emite ondas de radio. La rejilla de alambre es una antena matricial que localiza la posición de la plumilla midiendo la intensidad de la señal de radio recibida por cada alambre de la rejilla. Al comparar la intensidad de la señal recibida por cada alambre de la rejilla, la circuitería de demulticanalización permitirá calcular la posición de la plumilla aún cuando se encuentre entre los alambres.



El tablero acústico utiliza una tira de micrófonos localizada alrededor del perímetro del tablero. La plumilla genera una serie de pequeñas chispas cuando se coloca cerca del tablero. Sin embargo, la tableta acústica ha estado perdiendo popularidad ya que es demasiado ruidosa (produce un zumbido) y demasiado suceptible al ruido del ambiente.

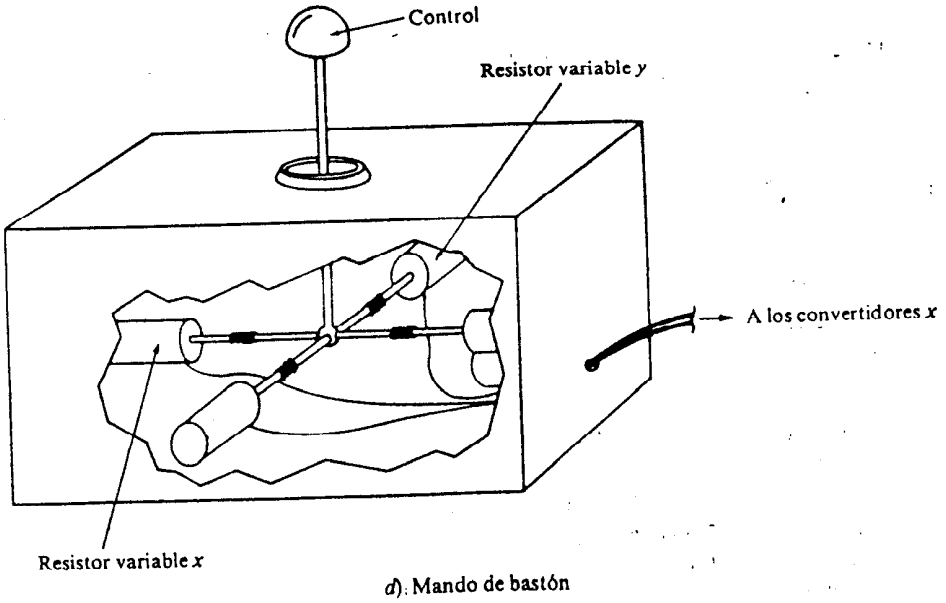
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



Joystick

El Joystick es una palanca que puede moverse a la izquierda, a la derecha, arriba o abajo. Dos potenciómetros recogen el movimiento del Joystick y lo convierten en posiciones X e Y. Esto hace del Joystick una herramienta ideal para controlar un cursor en la pantalla. Además, controlando el grado de movimiento, puede unirse directamente al Joystick la velocidad del movimiento del cursor.

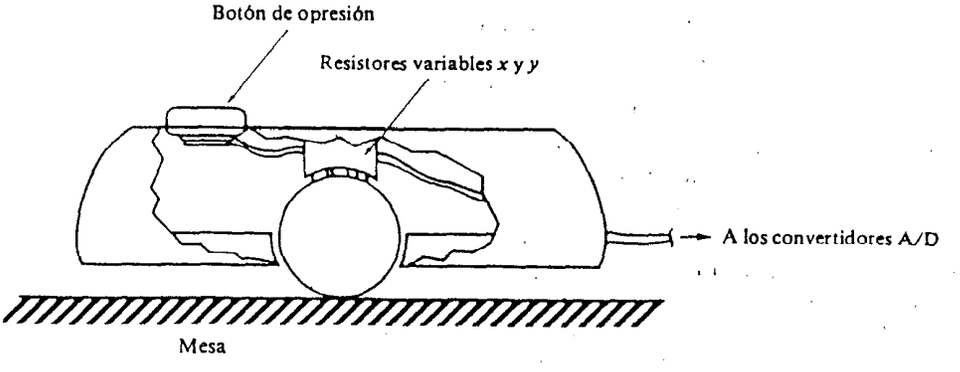
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



d). Mando de bastón

Ratón

El Ratón consiste de una pequeña caja con ruedas que puede moverse con la mano en todos los sentidos fácil y rápidamente. En la parte inferior tiene dos conjuntos de ruedas asociados a unos potenciómetros que producen el movimiento relativo. Esto hace del ratón, al igual que el Joystick, ideal para controlar el cursor. Por tanto, se utiliza, no para entrar datos gráficos, sino para interactuar con el programa. Para ayudar a esto tiene normalmente un conjunto de botones en la parte superior que pueden ser programados por software.

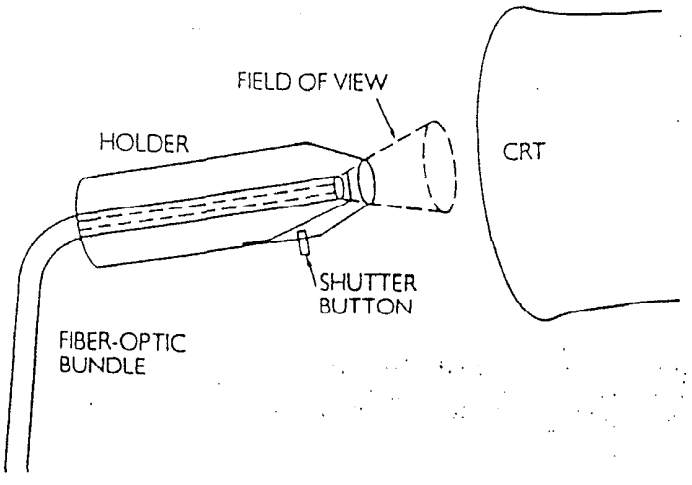


Lápiz Optico

El Lapiz Optico se utiliza principalmente en las pantallas vectoriales o de barrido. Es un pequeño dispositivo que puede detectar la precencia de luz en la pantalla mediante una fotocélula que está colocada en su extremo. Cuando la fotocélula detecta luz, la actividad del procesador de la pantalla se interrumpe y las coordenadas del punto en la pantalla puede devolverse al programa del usuario. Esta acción está acompañada por un usuario que coloca el lapiz óptico en la posición deseada de la pantalla y presionando un interruptor

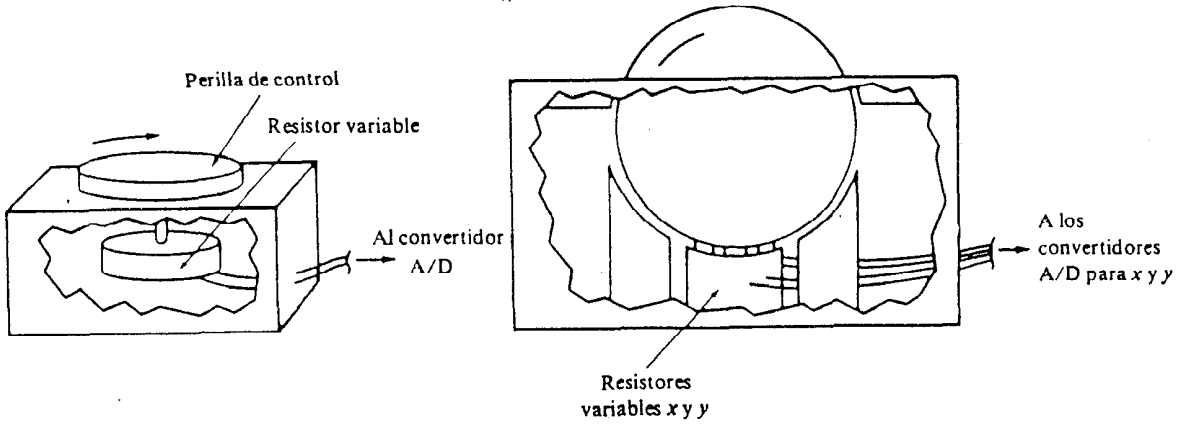
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

mecánico en el extremo del lápiz. En este momento algunos software de aplicación se hacen cargo y realizan una función apropiada tales como dibujar o borrar una línea. La principal desventaja del lapiz óptico es que necesita mantenerse en la pantalla, lo cual es incómodo.



Trackball (Esfera de Control)

El Trackball consiste de una pelota que rota dentro de una caja la cual gira unos potenciómetros produciendo coordenadas digitales. Este controla el cursor de la pantalla en proporción directa a la velocidad de su rodamiento y en la dirección de su rotación. Es dificil realizar movimientos largos y rápidos pero permite posicionamiento preciso del cursor ya que se controla fácilmente la velocidad de rotación.



2.5.2 Dispositivos de Salida

Un dispositivo de salida es una pieza mecánica que produce un "hard copy" de una imagen sobre cualquier papel o película.

Trazadores

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Permiten obtener dibujos permanentes en papel. Básicamente los hay de dos tipos:

- Secuencia Aleatoria de Dibujo: el útil de dibujo, normalmente una plumilla, dibuja trazos sobre el papel en una secuencia no prefijada de antemano. Sólo dibujan líneas.

- Secuencia de Barrido: la imagen gráfica se imprime línea a línea en el papel. Cada línea está formada por puntos muy próximos. Pueden dibujar paneles sólidos y, en muchos casos, permiten colores.

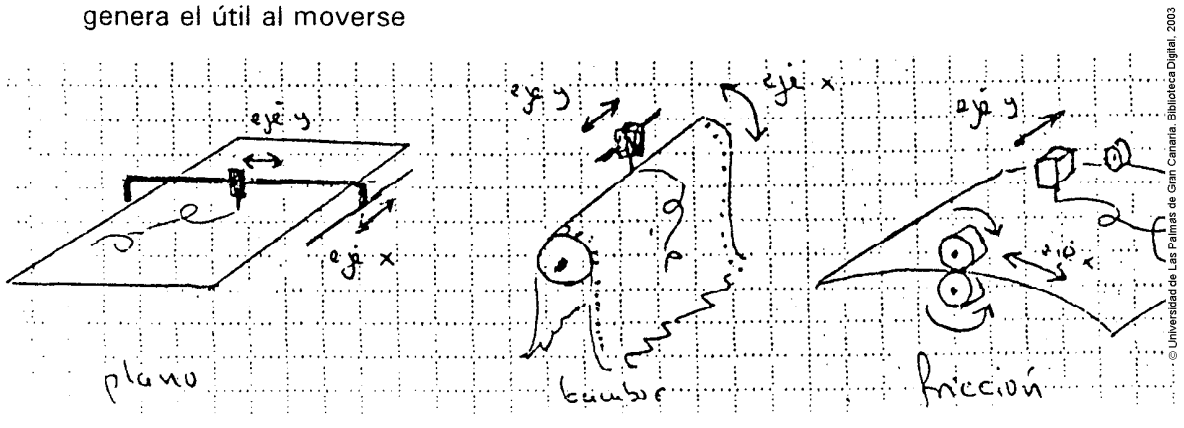
Características de los Trazadores

Existen diversos tipos según el movimiento del útil y el papel.

- Planos: papel fijo. El útil se mueve en los dos ejes X e Y.

- Tambor: Papel performado. Un eje de movimiento lo genera el giro del tambor. El otro lo proporciona el movimiento del útil en sentido horizontal.

- Fricción: el papel se desliza adelante y atrás, entre dos rodillos. El otro eje lo genera el útil al moverse



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

- pueden usar plumillas de varios gruesos y colores, y cambiar de plumillas durante el dibujo.

- tamaño desde DIN A4 hasta varios metros de largo y ancho.

- el tiempo de dibujo depende de:

  * cantidad de información a dibujar

  * tamaño del papel

  * velocidad de desplazamiento del útil-papel

- velocidades alcanzables: 50 cm/seg y aceleraciones de 4G (1G =9.8m/seg^2. Si un automovil pudiera acelerar a 4G, pasaría de 0 a 100 Km/h en 2.8 segundos).

Resolución de hasta 0.025 mm.

- normalmente incorporan un microprocesador que convierte las órdenes provenientes del computador para dibujar segmentos, en comandos incrementales para mover los motores X e Y con la aceleración y velocidad requeridas. Además, proporcionan otras facilidades como:

    * generación de caracteres en diversos tamaños, estilos y orientaciones

    * generación de arcos, círculos, etc.

- el interfaz con el computador, es similar al proporcionado por una pantalla vectorial

- al igual que estos, no puede dibujar paneles sólidos. Solo rayados compuestos de líneas más o menos próximas.

Trazadores de Barrido

Existen varias tecnologias:

- electrostáticos

- xerográficos

- chorros de tinta

- Los electrostáticos depositan cargas eléctricas en el papel, línea a línea, mediante un "peine" de contacto. Despues el papel se impregna con un plovo fino (toner) cargado en sentido contrario. Por último, el "toner" es fijado por calor. Usando "toners" de tres colores, en tres sucesivas pasadas se obtiene una imagen en colores.

- Los xerográficos crean primeramente la imagen en un tambor de selenio, depositando las cargas mediante un rayo laser. Después se imprime en papel con la misma técnica de la fotocopiadora. Repitiendo tres veces el proceso con distintos pigmentos de impresión, se obtienen imágenes en colores.

- Los de chorros de tinta emiten pequeñas gotas de tinta (en tres colores) con gran precisión, mediante una minibomba electromágnetica. Además, de los tres colores básicos suelen disponer de un chorro más para el negro, ya que este es dificil de obtener por sustracción. El papel puede estar dispuesto girando en un tambor, o la cabeza de la impresora desplazarse a lo largo de cada línea.

- dependiendo del tamaño, los más pequeños pueden usarse como impresoras. A veces son utilizados (en particular, los de chorros de tinta) para "volcar" a papel, punto a punto, la imagen de una pantalla de barrido.

- todas ellas comparten las siguientes características:

    * el tiempo de dibujo es independiente de la cantidad de información a dibujar

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

* necesitan una conversión a puntos previa. Si se utilizan para volcar la imagen de una pantalla, la conversión ya está hecha. En el caso general, esta conversión se hace, línea a línea, bien por el procesador de la aplicación o por un procesador especializado incorporado al trazador.

El interfaz entre el computador y los trazadores, sean aleatorios, sean de barrido (siempre que la conversión a puntos no sea parte de la aplicación), es muy similar al comentado para las pantallas. Sin embargo, no existe posibilidad de borrado selectivo, ni total. Un "barrido total" es equivalente a cambiar de papel. Al hablar de estandars gráficos independientes de los periféricos, veremos que las pantallas y los trazadores son indistinguibles desde la aplicación.

Impresoras

Son dispositivos de barrido por lo que comparten las características ya señaladas para los trazadores de barrido. En realidad, es dificil trazar una barrera entre las impresoras y los trazadores de barrido dado que muchas tecnologías, en particular impresión por laser y chorros de tinta, se aplican a ambos. Un mismo dispositivo puede funcionar a la vez como impresora y trazador.

Las que emplean cinta entintada y matriz de puntos suelen ser impresoras con "posibilidades gráficas", normalmente en blanco y negro.

En algún caso, existen impresoras con cinta tricolor que pueden generar una cierta gama de colores mediante el efecto de "integración espacial" (tramas de puntos de distintos colores, con mayor o menor intensidad para conseguir el tono deseado).

Normalmente son densidades de hasta 165 puntos/pulgadas (1.500 puntos por línea de impresión) y velocidades de un minuto por imagen gráfica de una página.

Cuando funcionan en modo alfanumérico permiten gran versatilidad: varios anchos de caracter, negritas, subrayado, subíndice, acentos, etc.

La conversión a puntos ha de hacerse en el computador de la aplicación, bién convirtiendo cada primitiva a puntos sobre una matriz bidimensional, bien almacenando cada primitiva y calculando después, línea a línea, qué se ha de imprimir.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2003

CAPITULO 3

ALGORITMOS DEL EQUIPO

3.1 INTRODUCCION

En este capítulo se presentan, esencialmente, algoritmos que tienen que ver con la conversión a puntos de primitivas gráficas para dibujar líneas y círculos.

Estos algoritmos se encuentran normalmente implementados en el procesador interno de las pantallas de barrido y en algunos trazadores, por lo que el programador de aplicaciones gráficas no tendrá, en general, que programarlos.

No obstante, es útil conocerlos, al menos por las siguientes razones:

* pueden utilizarse para convertir impresoras gráficas en periféricos gráficos cómodos de usar.

* ayudan a comprender mejor los problemas de eficiencia asociados a la conversión a pixels.

* son la base para implementaciones en hardware.

* son imprescindibles para diseñar un periférico de barrido.

3.2 ALGORITMOS DE DIBUJO DE LINEAS

Ya que una pantalla de tubo de rayos catódicos (CRT) puede considerarse como una matriz de pixels en la cual cada uno de ellos puede estar iluminado, no es posible dibujar directamente una línea recta desde un punto a otro. El proceso de determinar que pixels suministrará una mejor aproximación a la línea deseada se conoce como **rasterization**. Para líneas horizontales, verticales y de 45 grados los elementos de raster son obvios. Para cualquier otra orientación la elección es más dificil. Esto se muestra en la figura 3.1.

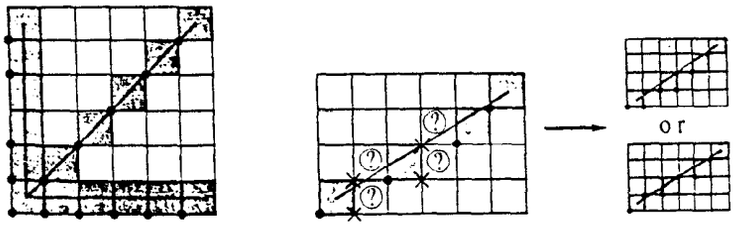


Figura 3.1

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Antes de discutir los algoritmos de dibujos de líneas es util considerar la necesidad de tales algoritmos, por ejemplo, cuales son las características deseables para estas líneas. Ciertamente, las líneas rectas aparecerán como líneas rectas y comienzan y acaban correctamente. Por lo tanto, las líneas mostradas deberían tener luminosidad constante en toda su longitud, independientemente de la longitud y orientación de la línea. Finalmente, las líneas se mostrarán rápidamente. La mayoria de las pantallas impiden la generación de una línea completamente recta excepto para casos especiales. No es posible para una línea empezar y acabar precisamente en lugares específicos. Sin embargo, con pantallas de buena resolución son posibles aproximaciones aceptables.

La luminosidad será constante sólo para líneas verticales, horizontales y de 45 grados. Para todas las otras orientaciones, la rasterization producirá una luminosidad desigual.

La mayoria de los algoritmos de dibujos de líneas utilizan métodos incrementales para simplificar los cálculos. A continuación se analizará distintos algorítmos de dibujos de líneas.

Algoritmo elemental

La ecuación de la recta es:

$Y = mX + B$ (siempre que $m <> oo$)

Sustituyendo los valores de los extremos de la línea (X_1,Y_1), (X_2,Y_2) se obtiene:

$Y_1 = mX_1 + B$

$Y_2 = mX_2 + B$

$$m = (Y_2-Y_1)/(X_2-X_1) = DY/DX$$

$$B = Y_1 - mX_1$$

Supongamos que $0 <= m <= 1$, es decir, la recta está en el primer octante del plano (más tarde generalizaremos el algoritmo). En ese caso, X_2-X_1 será siempre mayor que Y_2-Y_1.

El algoritmo incrementa **X** de uno en uno y, para cada valor de **X**, calcula el valor entero de **Y** más próximo a la **Y** real por donde debería de pasar la recta.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Algoritmo Linea1 (X_1,Y_1,X_2,Y_2:Integer)

```
var m,B,Y:real
        X:integer

m = real(Y2-Y1)/real(X2-X1)
B = real(Y1)-m*real(X1)

para X = X1 hasta X2 hacer

   Y = m * real(X) + B
   punto(X,round(Y))

fin para
```

- punto: ilumina un pixel en la memoria de refresco.

- real: convierte un entero a su real equivalente.

- round: convierte un real a su entero más próximo.

Algoritmo Incremental DDA (Digital Differential Analyzer)

No es necesario calcular, en cada paso, el valor de **Y** utilizando una multiplicación real como hace el algoritmo "Linea 1". Se puede calcular a partir de la **Y** anterior teniendo en cuenta que:

$m = DY/DX,\ DX = 1 \ ===> \ Y_{i+1} = Y_i + m$

Algoritmo Linea2 (X_1,Y_1,X_2,Y_2:integer)

```
var m,Y:real
    X:integer

m = real(Y2-Y1)/real(X2-X1)
Y = real (Y1)
X = X1

mientras X < = X2 hacer

      punto(X,round(Y))
      X = X + 1
      Y = Y + m

fin mientras
```

Si hubiera que tratar líneas en el segundo octante ($Y_2-Y_1 > X_2-X_1$) se incrementaría la **Y** de uno en uno, y se redondearía el valor de **X** resultante. La generalización a los ocho octantes es la siguiente:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Algoritmo Linea3 (X_1,Y_1,X_2,Y_2:integer)

var X,Y,incX,incY:real
i,long:integer

si abs(X_2-X_1) > = abs(Y_2-Y_1) **entonces**
long = abs(X_2-X_1)
si no
long = abs(Y_2-Y_1)
fin si

incX = real(X_2-X_1)/real(long) {+-1 si long = abs(X_2-X_1)}
incY = real(Y_2-Y_1)/real(long) {+-1 si long = abs(Y_2-Y_1)}
X = real(X_1)
Y = real(Y_1)

para i = 0 **hasta** long **hacer**

punto(round(X),round(Y))
X = X + incX
Y = Y + incY

fin para

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

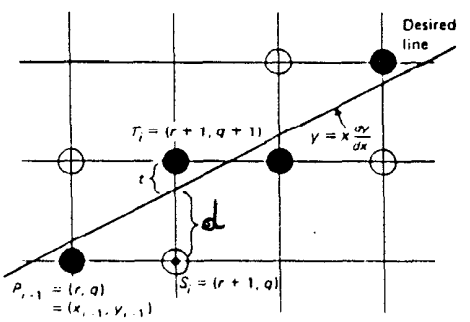
Algoritmo de Bresenham

Los algoritmos anteriores, a pesar de ser incrementales y no necesitar operaciones de multiplicación, trabajan con aritmética real. La operación de redondeo es especialmente costosa.

El algoritmo de Bresenham (J.E.Bresenham, 1965), desarrollado especialmente para trazadores, utiliza exclusivamente aritmética entera.

La primera versión del mismo utiliza todavía una variable real. Supondremos, de nuevo, que la recta a dibujar se encuentra en el primer octante (pendiente entre 0 y 1).

La X se va incrementando de uno en uno. Para cada valor de X siempre habrán dos pixels candidatos a iluminar. Se iluminará aquel que quede más cerca de la línea teorica.



La decisión se toma en base a que la distancia **d** de la recta al pixel es < = 1/2 en valor absoluto:

$$-1/2 <= d < 1/2$$

En el siguiente pixel X_{i+1}, la coordenada Y ha crecido en la pendiente m:

$$Y_{i+1} = Y_i + m$$

por tanto, la distancia d ha crecido en la misma cantidad:

$$d_{i+1} = d_i + m$$

Si la nueva distancia sigue comprendida entre -1/2 y +1/2, se siguen iluminando pixels de la misma fila. Si supera (o iguala) 1/2, se ilumina el pixel de la fila superior y se actualiza la distancia restándole 1:

$$Y = Y + 1 \qquad d = d - 1$$

De este modo, **d** lleva cuenta del error en más o en menos, que se comete al iluminar un pixel con respecto a la línea teórica.

Para simplificar la compresión del valor de **d**, la variable utilizada es, en realidad, e = d - 1/2. Así:

$$-1/2 <= d < 1/2 \quad <==> \quad -1 <= e < 0$$

y

$$d >= 1/2 \quad <==> \quad e >= 0$$

con lo cual, basta con comprobar el signo de e.

Algoritmo Linea4 (X_1,Y_1,X_2,Y_2:integer)

```
var m,e:real
    X,Y:integer

m = real(Y2-Y1)/real(X2-X1)
X = X1
Y = Y1
e = -0.5

mientras X <= X2 hacer
        punto(X,Y)
        X = X + 1
        e = e + m
        si e >= 0 entonces
                        Y = Y + 1
                        e = e - 1
        fin si

fin mientras
```


© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

La última transformación consiste en sustituir la **e** real por una **e** entera **ee**, relacionadas de la siguiente forma:

$$ee = 2 * e *(X_2 - X_1)$$

lo cual lleva a las siguientes equivalencias:

$e = -0.5 \quad <===> \quad ee = -(X_2 - X_1)$
$e = e + m \quad <===> \quad ee = ee + 2 * (Y_2 - Y_1)$
$e >= 0 \quad <===> \quad ee >= 0$
$e = e - 1 \quad <===> \quad ee = ee - 2 * (X_2 - X_1)$

Algoritmo Linea 5 (X_1,Y_1,X_2,Y_2:integer)

var incX,inc2X,inc2Y,X,Y,ee:integer

incX = $X_2 - X_1$
inc2X = 2 * incX
inc2Y = 2 * ($Y_2 - Y_1$)
X = X_1
Y = Y_1
ee = -incX

mientras X < = X_2 **hacer**

    punto(X,Y)
    X = X + 1
    ee = ee + inc2Y

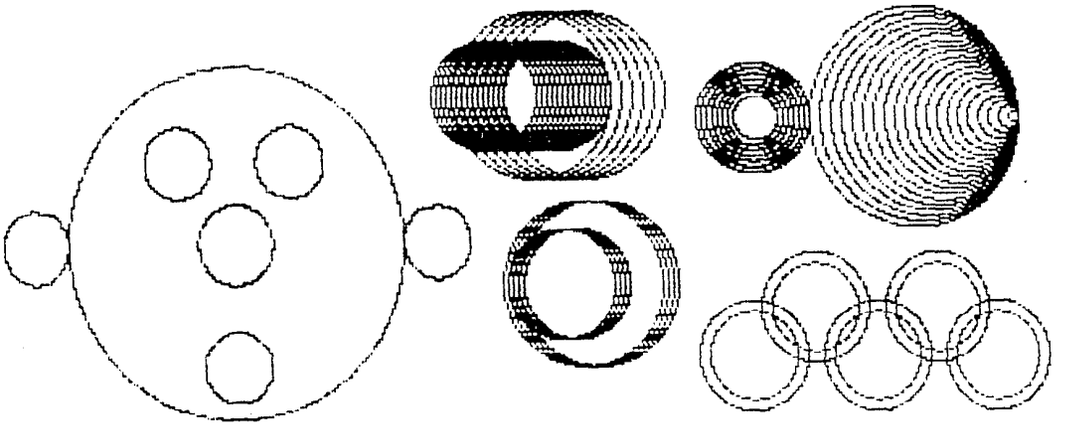
    si ee > = 0 **hacer**
        Y = Y + 1
        ee = ee - inc2X
    fin si

fin mientras

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3.3 DIBUJO DE CIRCULOS

Los círculos son probablemente las curvas más utilizadas en los gráficos elementales. Generalmente, se utilizan como bloques de construcción para generar imágenes artísticas (Figura 3.2). A continuación se verá varios algoritmos de dibujos de círculos.



(a) (b)

Figura 3.2

Representación Implícita

Un círculo se representa por las coordenadas de su centro (X_c,Y_c) y su radio R (Figura 3.3). La ecuación más familiar de un círculo es:

$$(X - X_c)^2 + (Y - Y_c)^2 = R^2 \qquad (3.1)$$

Si el centro del círculo está situado en el origen (0,0), la ecuación 3.1 se reduce a:

$$X^2 + Y^2 = R^2$$

Resolviendo la ecuación 3.1 para Y:

$$Y = Y_c +- \sqrt{R^2 - (X - X_c)^2}$$

Para dibujar un círculo, se incrementa los valores de X una unidad desde -R a +R y se utiliza la ecaución 3.1 para resolver los dos valores de Y en cada paso. Por supuesto, es necesario convertir a enteros los valores antes de dibujar.

Este método de dibujar un círculo no es eficiente por varias razones. Primero, no se toma la ventaja de la simetría del círculo. Si hemos calculado un punto de un círculo, veremos que otros siete puntos son conocidos inmediatamente. Un segundo problema es la cantidad de tiempo de proceso requerido para realizar rápidamente las operaciones de

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

cuadratura y raiz cuadrada.

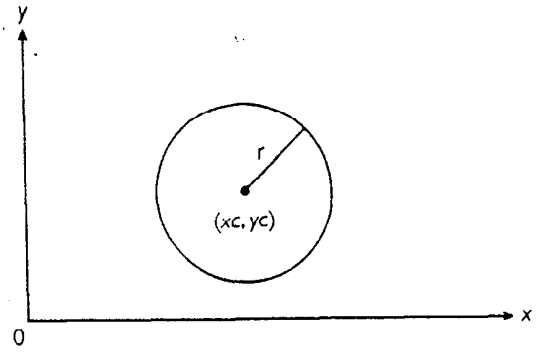


Figura 3.3

El tercer problema no está tan claro. Si dibujamos un cuadrante de un círculo con centro en el origen, obtenemos la figura 3.4. De esta figura vemos que, aunque los valores de X son igualmente espaciados (difieren por una unidad), los de Y no lo son. El círculo es denso y plano cerca del eje Y, y tiene huecos grandes y está pendiente cerca del eje X.

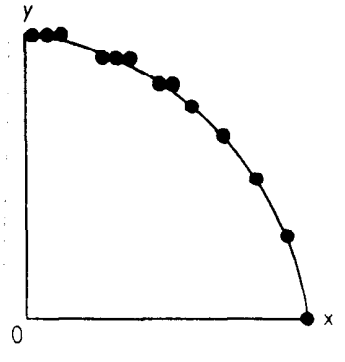


Figura 3.4

Supongamos, por claridad, que el centro del círculo está en el origen. Un cambio en el valor de Y puede estar relacionado con un cambio en valores de X por la ecuación (diferencial) $dY = (-X/Y)dX$. El cambio en X, o dX, se fija en una unidad. A medida que X se acerca a cero, la parte derecha de esta ecuación llega a ser muy pequeña, y por lo tanto el cambio en el valor de Y, o dY llega a ser muy pequeño. Estos valores consecutivos cerca del eje Y serán aproximadamente iguales. Cuando estos valores de Y se redondea a los valores enteros más cercanos, el valor entero es el mismo. Esto tiene como resultado los grupos de valores de Y igual en la figura 3.4. Sin embargo, cuando X tiende al radio **R** los valores de Y se aproximan a cero (ya que $X^2 + Y^2 = R^2$) y dY llega a ser muy grande.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Representación Polar Paramétrica

Un método de eliminar el problema de dibujar puntos no uniformemente espaciados alrededor del círculo es utilizar la representación polar de un círculo:

$$X = X_c + R\ Cos(\theta)$$

$$Y = Y_c + R\ Sen(\theta)$$

donde θ se mide en radianes desde 0 a 2π. Ya que la longitud del arco es igual a **RXθ** y **R**, el radio, es constante, los incrementos iguales de θ, dθ, tiene como resultado un espaciado igual (longitud del arco) entre puntos dibujados sucesivamentes (Figura 3.5).

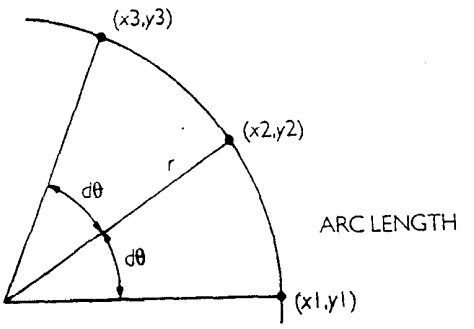


Figura 3.5

Dibujo Incremental

Uno de los problemas de utilizar la representación polar para dibujar un círculo es que, en la mayoría de los microcomputadores, el cálculo repetido de los valores para Cos(θ) y Sen(θ) consume una cantidad significante de tiempo de procesamiento. Podemos aumentar la velocidad de computación utilizando un método incremental que calcule los puntos de un círculo de las coordenadas del punto calculado anteriormente. Esta técnica necesita solamente un cálculo inicial del seno y del coseno. De nuevo para claridad de la explicación, el centro del círculo se coloca en el origen.

Los dos puntos consecutivos (X_1,Y_1) y (X_2,Y_2) de un círculo se relacionan por:

$$X_1 = R\ Cos(\theta)$$

$$Y_1 = R\ Sen(\theta) \qquad (3.2)$$

$$X_2 = R\ Cos(\theta + d\theta)$$

$$Y_2 = R\ Sen(\theta + d\theta)$$

donde **dθ** es un tamaño de paso angular fijado (Figura 3.5)

Usando trigonometría, obtenemos

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$X_2 = R\,Cos(\theta)Cos(d\theta) - R\,Sen(\theta)Sen(d\theta)$$

$$Y_2 = R\,Sen(\theta)Cos(d\theta) + R\,Cos(\theta)Sen(d\theta)$$

Sustituyendo X_1 e Y_1 de las ecuaciones 3.2 tenemos

$$X_2 = X_1Cos(d\theta) - Y_1Sen(d\theta)$$

$$Y_2 = Y_1Cos(d\theta) + X_1Sen(d\theta)$$

Estas son las ecuaciones incrementales que queremos para generar un círculo. Si comenzamos el círculo en $X_1 = 0$, $Y_1 = R$ y los incrementos del ángulo fijado, podemos computarizar todos los puntos en el círculo calculando Cos ($d\theta$) solamente una vez.

Antes de utilizar estas ecuaciones para dibujar un círculo, discutiremos la simetría. Si un punto (a,b) pertenece al círculo $X^2 + Y^2 = R^2$ centrado en el origen, tenemos otros siete puntos: (-a,b), (a,-b), (-a,-b), (b,a), (-b,a), (b,-a), (-b,-a), como ilustra la figura 3.6. (Para demostrar esto, sustituya los ocho puntos en la ecuación del círculo). Tomemos ventaja de esta simetría calculando solamente uno de cada ocho valores de un círculo, con un intervalo angular de 45 grados. Si el primer punto es el (0,R), los cálculos se terminan cuando Y = X.

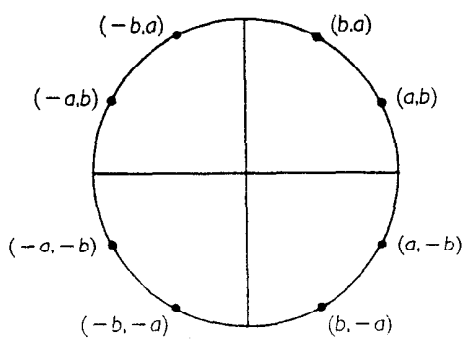


Figura 3.6

Para encontrar los puntos simétricos de un círculo centrado en (X_c,Y_c), se añade X_c a la primera coordenada e Y_c a la segunda coordenada para cada uno de los ocho puntos.

El siguiente algoritmo, Círculo, calcula los puntos de un círculo centrado sobre el origen y entonces añade X_c a los valores de X e Y_c a los valores de Y, moviendo el centro a (X_c,Y_c). Para hacer el círculo de forma circular, los valores de Y se multiplican por un ratio antes de dibujarlo. Ya que se utilizan coordenadas enteras del screen, el radio es una distancia entera desde el centro del círculo (por lo tanto se utiliza la función **real**). Se desea obtener un espaciado de una unidad aproximadamente entre los puntos adyacentes. Esto implica que la longitud del arco, R X $d\theta$, será una unidad; por lo tanto, $d\theta$ será 1/radio. Antes de llamar a este algoritmo, examinaremos que el círculo esté dentro de los límites del screen utilizando el procedimiento de llamada. Esto se realiza comparando X_c +- R e

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Y_c +- R con las coordenadas máximas y mínimas de los ejes horizontal y vertical, respectivamente.

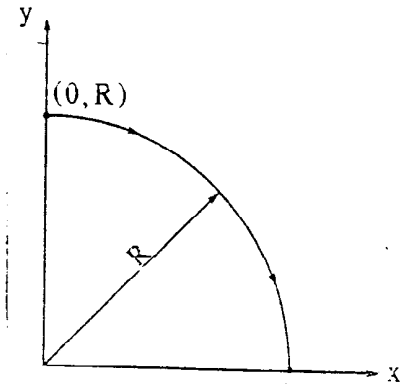
```
Algoritmo Círculo(
    Xc,Yc              {centro del círculo}
    radio : integer)     {radio del círculo}

var
    dtheta,            {incremento del ángulo}
    ct,st,             {seno y coseno dθ}
    X,Y,               {puntos computados
    xtemp : real       {almacena los valores antiguos de X}

ar = 1.33
θ = 1/radio
ct = cos(θ)
st = sin(θ)

X = 0
Y = radio

mientras Y > = X hacer

    punto(round(Xc + X),round(Yc + Y*ar))
    punto(round(Xc - X),round(Yc + Y*ar))
    punto(round(Xc + X),round(Yc - Y*ar))
    punto(round(Xc - X),round(Yc - Y*ar))
    punto(round(Xc + Y),round(Yc + X*ar))
    punto(round(Xc - Y),round(Yc + X*ar))
    punto(round(Xc + Y),round(Yc - X*ar))
    punto(round(Xc - Y),round(Yc - X*ar))
    xtemp = X
    X = (X*ct - Y*st)
    Y = (Y*ct + xtemp*st)

fin mientras
```

3.4 ALGORITMO DE BRESENHAM DE DIBUJO DE CIRCULOS

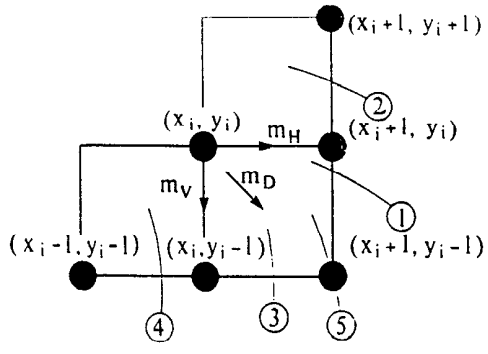
El algoritmo genera un cuadrante de circunferencia. Por simetría respecto a los ejes **Y** y **X** se puede obtener toda la circunferencia.

Supondremos el origen en el centro de la circunferencia (si no, bastaría con una traslación). El centro se supone un pixel exacto, y el radio R, un número entero. Se generará el cuadrante en el sentido de las agujas del reloj.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



La **Y** es función monótona decreciente de la **X** a lo largo del cuadrante. Si (X_i,Y_i) es el último pixel dibujado, el siguiente solo puede ser uno de los tres:



movimiento horizontal M_h --> (X_i+1,Y_i)
movimiento diagonal M_d --> (X_i+1,Y_i-1)
movimiento vertical M_v --> (X_i,Y_i-1)

El círculo real seguirá una de las trayectorias señaladas de 1 a 5.

Las siguientes magnitudes expresan el cuadrado del error que se cometería al aproximar el círculo teórico mediante uno de dichos pixels. Si la cantidad es positiva, el pixel sería exterior al círculo. Si es negativa, el pixel es interior:

$$D_h = (X_i + 1)^2 + Y_i^2 - R^2$$
$$D_d = (X_i + 1)^2 + (Y_i-1)^2 - R^2$$
$$D_v = X_i^2 + (Y_i-1)^2 - R^2$$

Las 5 trayectorias posibles del círculo teórico se relacionan del siguiente modo con el valor de D_d:

$D_d < 0$: casos 1 y 2. La mejor aproximación es el pixel M_h o M_d.
$D_d > 0$: casos 3 y 4. La mejor aproximación es el pixel M_v o M_d.
$D_d = 0$: caso 5. La aproximación exacta es el pixel M_d.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

caso 1

Hay que ver que distancia es más grande en valor absoluto:

si $|D_h| > |D_d|$: la mejor aproximación es M_d.

si $|D_h| <= |D_d|$: la mejor aproximación es M_h.

(el caso $|D_h| = |D_d|$ se asigna arbitrariamente a M_h)

caso 2

La mejor aproximación es, si duda, M_h. Note que:

en caso 1 : $D_d < 0$ and $D_h > 0$
en caso 2 : $D_d < 0$ and $D_h <= 0$

Para decidir ambas cuestiones con un solo computo, se calcula la magnitud:

$d = D_h + D_d = 2 D_d + 2Y_i - 1$

- si $d <= 0$: o bien se trata del caso 2, o bien se trata del caso 1 con $|D_h| <= |D_d|$. En cualquier caso, el pixel a elegir es M_h.

- si $d > 0$:se ha de tratar del caso 1, con $|D_h| > |D_d|$. El pixel a elegir es M_d.

caso 3

Hay que comparar las distancias $|D_d|$ y $|D_v|$:

- si $|D_d| <= |D_v|$: La mejor aproximación es M_d.

- si $|D_d| > |D_v|$: La mejor aproximación es M_v.

(el caso $|D_d| = |D_v|$ se asigna arbitrariamente a M_d)

caso 4

La mejor aproximación es M_v. Ahora los signos de D_d y D_v son:

caso 3 : $D_d > 0$ and $D_v < 0$
caso 4 : $D_d > 0$ and $D_v >= 0$

Igual que antes, para decidir en que caso estamos se calcula la magnitud:

$d' = D_d + D_v = 2 D_d - 2 X_i - 1$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

- si d' > 0 :o bien se trata del caso 4, o bien, del caso 3 con $|D_d| > |D_v|$. En ambas situaciones, el pixel a elegir es M_v.

- si d' < = 0 :se trata del caso 3 con $|D_d| <= |D_v|$. el pixel a elegir es M_d.

En resumen, para decidir cual de los tres pixel es el más aproximado, basta con calcular D_d y, en ese caso, d o d', que están expresadas en términos de D_d.

No es necesario calcular D_d en cada punto (X_i,Y_i) (lo llamaremos ahora Di). Se puede calcular incrementalmente a partir de Di en el punto anterior:

Si D_i,X_i,Y_i son los valores en el pixel en curso, los siguientes D_{i+1},X_{i+1},Y_{i+1} dependen de éstos de distintas formas, según nos hayamos movido horizontalmente, verticalmente, o en diagonal.

1) Movimiento Horizontal

$$X_{i+1} = X_i + 1$$

$$Y_{i+1} = Y_i$$

$$D_{i+1} = (X_{i+1} + 1)^2 + (Y_{i+1} - 1)^2 - R^2 =$$

$$= (X_{i+1})^2 + 2X_{i+1} + 1 + (Y_i - 1)^2 - R^2 =$$

$$= (X_i + 1)^2 + 2(X_i + 1) + 1 + (Y_i - 1)^2 - R^2 =$$

$$= D_i + 2X_i + 3$$

2) Movimiento Vertical (no se desarrolla)

$$X_{i+1} = X_i$$

$$Y_{i+1} = Y_i - 1$$

$$D_{i+1} = D_i - 2Y_i + 3$$

3) Movimiento Diagonal (no se desarrolla)

$$X_{i+1} = X_i + 1$$

$$Y_{i+1} = Y_i - 1$$

$$D_{i+1} = Di + 2X_i - 2Y_i + 6$$

Los valores iniciales, dibujando el cuadrante en sentido de las agujas del reloj, son:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$X_i = 0$

$Y_i = R$

$D_i = (0 + 1)^2 + (R - 1)^2 - R^2 = 2 - 2R$

De este modo, se consigue un algoritmo con aritmética exclusivamente entera, donde las únicas multiplicaciones son por 2 (en realidad, son sumas, o desplazamientos a la izquierda). Es decir, se trata de un algoritmo extremadamente eficiente.

```
Algoritmo Circunferencia (R:integer)

var X,Y,inc,d:integer

X = 0
Y = R
inc = 2 - 2*R

mientras Y >= 0 hacer

    punto(X,Y)
    opcion
     inc = 0 : diag(X,Y,inc)

     inc < 0 : d = 2 * inc + 2 * Y - 1
     opcion
        d <= 0 : horiz(X,Y,inc)

        d > 0  : diag(X,Y,inc)
     fopcion

     inc > 0 : d = 2 * inc - 2 * X - 1
     opcion
        d > 0  : vert(X,Y,inc)

        d <= 0 : diag(X,Y,inc)
     fopcion
    fopcion

fin mientras
```

Las acciones "horiz", "vert" y "diag" actualizan los valores de X, Y, e inc, de acuerdo con las fórmulas de recurrencia explicadas en (1), (2) y (3) respectivamente.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3.5 ELIPSES

Una elipse es una variación de un círculo. Alargando un círculo en una dirección se produce una elipse. Examinaremos solamente las elipses que están alargadas en la dirección X o Y. Las ecuaciones polares para este tipo de elipses centradas en (X_c,Y_c) son:

$$X = X_c + a * Cos(\theta)$$
$$Y = Y_c + b * Sen(\theta) \qquad (3.3)$$

donde el ángulo θ toma valores entre 0 y 2 π radianes(figura 3.7).

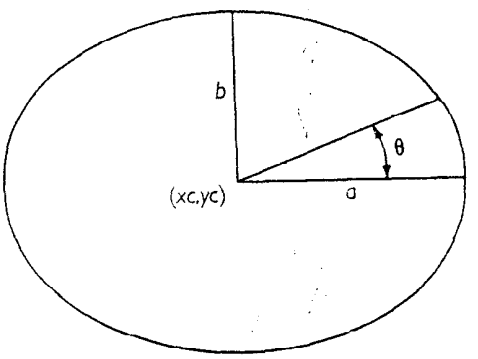


Figura 3.7

Los valores de a y b afectan a la forma de la elipse. Si b>a, la elipse es mayor en la dirección Y. Si b>a, la elipse es mayor en la dirección X. La elipse puede dibujarse utilizando una simetría de cuatro puntos: si (c,d) pertenece a la elipse, también pertenecen a ella los puntos (-c,d), (c,-d), (-c,-d).

Las siguientes ecuaciones incrementales para una elipse se derivan de las ecuaciones 3.3:

$$X_2 = X_1 Cos(d\theta) - (a/b)Y_1 Sen(d\theta)$$
$$Y_2 = Y_1 Cos(d\theta) - (b/a)X_1 Sen(d\theta) \qquad (3.4)$$

El algoritmo Círculo se puede modificar para dibujar una elipse mediante la inicialización de X a cero e Y a b y reemplazar las ecuaciones del círculo por las ecuaciones 3.4. De hecho, el algoritmo Circulo puede dibujar una elipse simplemente cambiando el ratio.

3.6 ESPIRALES

Una espiral es otra variación de un círculo. Las espirales pueden dibujarse mediante el incremento gradual de los radios cuando se dibuja un círculo. Un Procedimiento Espiral realiza la espiral de la figura 3.8 utilizando las ecuaciones paramétricas del círculo. Esto se

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

hace incrementando el radio mientras se incrementa el ángulo polar. La figura 3.9 muestra otra espiral; una con un incremento radial constante de dos unidades y un incremento angular de 1.5 radianes.

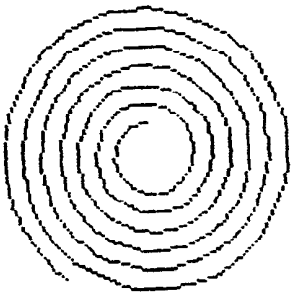


Figura 3.8

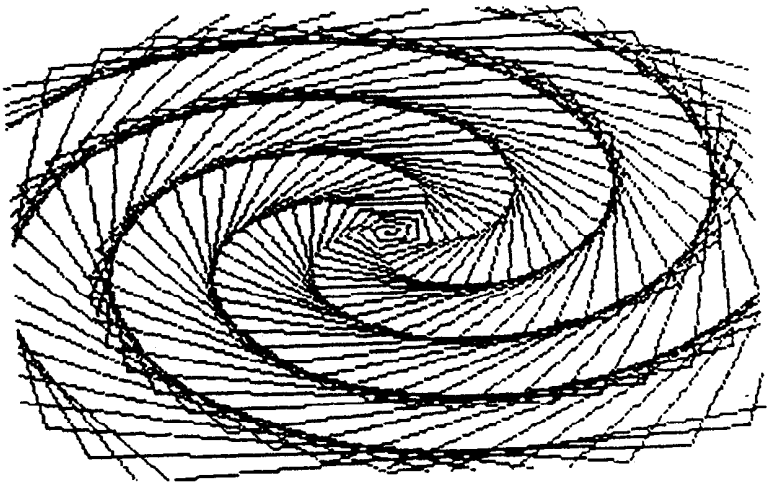


Figura 3.9

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3.7 RELLENADO DE POLIGONOS

3.7.1 Introducción

La mayor parte de los objetos complejos que son generados por computador pueden representarse por uno o más polígonos. Incluso si un objeto no es poligonal en su forma natural, puede utilizarse una forma poligonal de él. Por ejemplo, un círculo se representa como un polígono con muchas caras. Mientras se dibuja un polígono, podemos utilizar las características de nuestro sistema de pantalla para rellenar el espacio interior del polígono. Si se iluminan todos los pixels interiores a los límites del poligono, se realiza un relleno sólido. También es posible iluminar los pixels en una línea si y otra no, etc. Incluso podemos llenarlo con un conjunto arbitrario predefinido. Los diferentes tipos de relleno pueden aumentar el realismo de un objeto de un polígono definido, o las características de diferentes puntos de varios polígonos. La figura 3.10 muestra varios rellenados de polígonos.

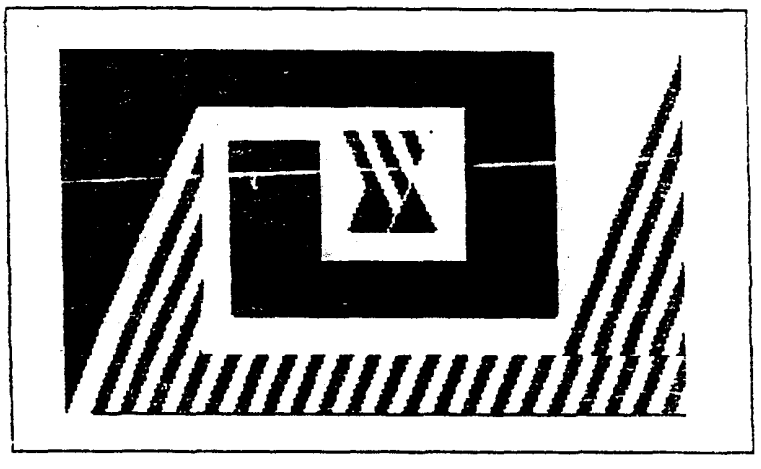


Figura 3.10

A continuación se examinarán varias técnicas y algoritmos para el rellenado de polígonos. Comenzaremos con una discusión de las diferentes representaciones de los polígonos.

3.7.2 Representación de Poligonos

Quizás, la forma más natural de representar un polígono es enumerando sus n vértices en una lista ordenada: $P=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_n,Y_n)\}$. El polígono puede construirse situandose en el primer vértice, (X_1,Y_1), y emitir comandos de dibujos a los restantes n-1 vértices. Para cerrar el polígono, se necesita un último comando de dibujo desde el último vértice al primero (Figura 3.11).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

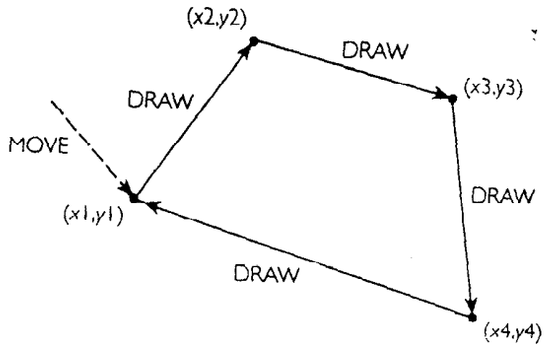


Figura 3.11

El problema con esta representación es que si queremos trasladar el polígono, es necesario aplicar la transformación de traslación a cada uno de los vértices para obtener el polígono trasladado. Para los objetos definidos con muchos polígonos con muchos vértices, puede ser un proceso que consume un tiempo considerable. Un método para reducir el tiempo de computación es representar el polígono mediante la localización absoluta de su primer vértice (X_1,Y_1) y representar los siguientes vértices como posiciones relativas del vértice anterior. Esto nos permite trasladar el polígono simplemente cambiando las coordenadas del primer vértice. Veamos un ejemplo utilizando esta idea.

El polígono P de la figura 3.12 está representado en **coordenadas absolutas** por:

P = {(-1,-1),(-1,1),(3,1),(3,-1)}.

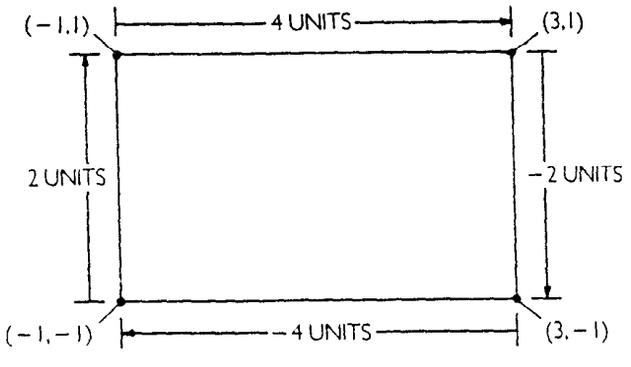


Figura 3.12

Para trasladar P tres unidades en la dirección X y cinco unidades en la dirección Y, es necesario sumar 3 a cada una de las coordenadas X y 5 a cada una de las coordenadas Y de P. Esto necesita un total de ocho sumas. Sin embargo, si representamos el mismo polígono P en **coordenadas relativas**:

P = {(-1,-1),(0,2),(4,0),(0,-2)}

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

donde el primer par es el vértice inicial y los otros pares son distancias indicadas, podemos trasladar el polígono por la misma cantidad simplemente sumando tres unidades a la coordenada X y cinco unidades a la coordenada Y de la primera coordenada (-1,-1). Este método necesita solamente dos sumas.

Los polígonos también pueden representarse listando sus aristas y vértices. Esta operación hace más facil determinar las propiedades geométricas como por ejemplo que polígonos comparten una arista común. Esta representación necesita más almacenamiento pero reduce el tiempo de computación gastado en determinar las propiedades geométricas.

Para simplificar la discusión de rellenado de polígonos, los representaremos utilizando sus coordenadas absolutas.

3.7.3 Frame Buffer Filling

Rellenado por Línea de Barrido (SCAN-LINE FILLING)

Un método relativamente fácil de rellenar un polígono es tener primero los límites de este dibujado en un buffer de trama. En la mayor parte de los casos el polígono se ha entrado interactivamente y está dibujado en la pantalla. De esta forma, solo los pixels en el buffer de trama que están iluminados son aquellos que definen el polígono. El algoritmo de relleno continua de la misma forma que la línea de barrido comenzando por el pixel más a la izquierda en cada una de las líneas de barrido. Mientras se barre de izquierda a derecha, se utiliza la función booleana **screenbit**(X,Y) (verdadera si el pixel (X,Y) está iluminado) para determinar si ha sido encontrado una arista analizando si el pixel está a ON. Si lo está, los pixels siguientes en esa línea de barrido se iluminan hasta que se encuentre la próxima arista. Esta arista se encuentra de la misma forma que anteriormente. Para aumentar la velocidad de relleno, los primeros algoritmos determinan los valores de X tanto del borde izquierdo como del derecho de la línea de barrido y dibujan entonces una línea entre ellos, rellenando en ese segmento. Esto se repite hasta que se alcanze el último pixel de esa línea de barrido. (Figura 3.13).

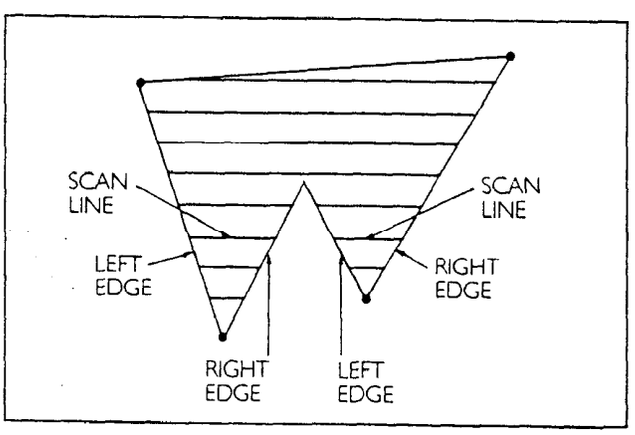


Figura 3.13

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

El siguiente algoritmo, Buffer_Fill implementa este rellenado.

```
Algoritmo Buffer_Fill

var
      X,Y, Xstart:integer

Ymax = 511
Xmax = 511

para Y = 0 hasta Ymax hacer

  X = 0

1: mientras not screenbit(X,Y) and (X<Xmax) hacer
     X = X + 1
  fin mientras

  si X < Xmax entonces
     X = X + 1
     Xstart = X

     mientras not screenbit(X,Y) hacer
        X = X + 1
     fin mientras

     Linea(Xstart,Y,X,Y)
     X = X + 1

   fin si

if X < Xmax goto 1

fin para
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Un problema con este algoritmo es el test repetido necesario para determinar el interior del polígono utilizando la función booleana **Screenbit**. Para buffers de tramas grandes que contienen a un polígono pequeño, todo el tiempo se gasta testeando los pixels que no están encendidos. Podriamos hacer el algoritmo más eficiente limitando los valores X e Y a los valores máximos y mínimos que alcanza el polígono.

Otro problema con el procedimiento Buffer_Fill es que no realiza correctamente el relleno con todo tipo de polígonos. Por ejemplo, como se ilustra en la Figura 3.14, el algoritmo dibuja incorrectamente una línea entre los dos vértices inferiores A y B.

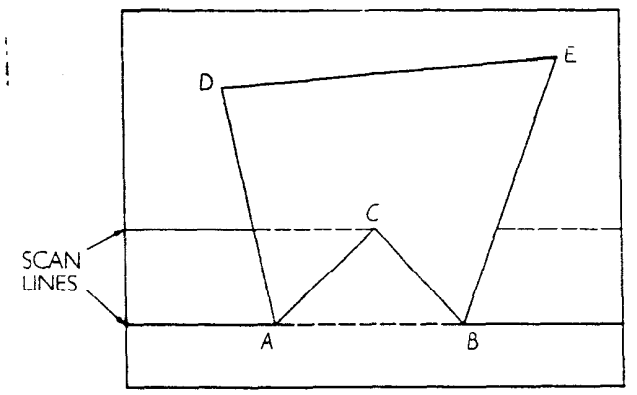


Figura 3.14

Rellenado por Inundación (FLOOD FILLING)

El rellenado correcto del polígono reside en el buffer de trama, utilizando flood filling, se selecciona cualquier pixel interior al polígono y se envia al procedimiento recursivo Flood_Filling. Cada vez que un pixel interior se envia al Flood_Filling (**screenbit**(X,Y) es falso), ese pixel se ilumina y, a través de sus propias llamadas, se realiza la misma prueba sobre cada uno de sus cuatro pixels vecinos. Ya que el límite del polígono está en el buffer de trama, todos los pixels límites (X,Y) tienen **screenbit**(X,Y) igual a verdadero.

Mientras que este procedimiento es fácil de cifrar, la profundidad de la recursión produce generalmente "stack overflow" en los microcomputadores con memoria limitada. Un método de tratar con este problema es crear una variable entera global, "depth" con valor igual al número que se llama al procedimiento de recursividad. Si "depth" llega a ser mayor que el tamaño del stack, 500 en el procedimiento, el procedimiento se termina antes de que el sistema aborte. En este momento, el usuario debe elegir un nuevo pixel interior no iluminado y pasarlo al Flood_Fill. Este proceso debe repetirse hasta que se rellene todo el polígono

Algoritmo Flood_Fill (X,Y:**integer**)

```
si not screenbit(X,Y) and depth < 500 entonces
    depth  = depth + 1
    punto(X,Y)
    Flood_Fill(X + 1,Y)
    Flood_Fill(X-1,Y)
    Flood_Fill(X,Y + 1)
    Flood_Fill(X,Y-1)
    depth  = depth - 1
fin si
```

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3.7.4 Rellenado Convexo

Un polígono es Convexo si para 2 puntos cualquiera situados dentro del polígono, la línea que nos une también está dentro del polígono. Un polígono es Convexo Horizontal, si para dos puntos cualquiera del polígono que tengan el mismo valor de **Y**, la línea horizontal que los une también pertenece al polígono. Todos los polígonos convexos son también convexos horizontal. La figura 3.15 muestra distintos polígonos: convexo, convexo horizontal y no convexo. El algoritmo de relleno que veremos dentro de este apartado, trabaja solamente con polígonos convexo y convexos horizontales.

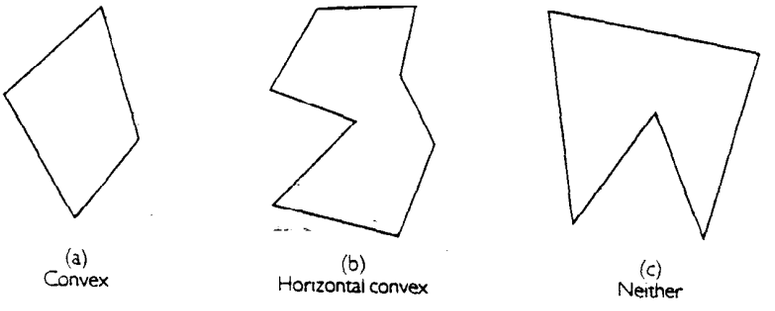
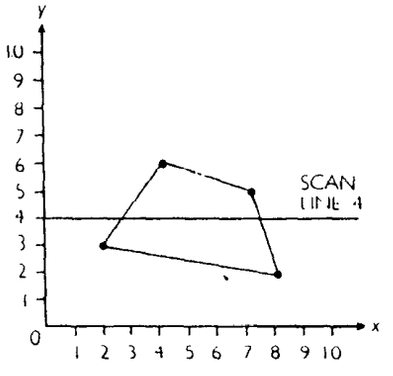


Figura 3.15

Según la definición, un polígono de este tipo tiene como máximo dos aristas en la línea de barrido. Vamos a descartar todas las aristas horizontales, las cuales veremos que no afectan el rellenado.

Para almacenar los valores de **X** máximo y mínimo de las aristas para cada línea de barrido, se utiliza un tabla de líneas de barrido. El valor **X** de la arista izquierda se coloca en el valor mínimo, mientras que el valor **X** de la arista derecha se coloca en el valor máximo; esto se hace para cada línea de barrido. La tabla se inicializa de tal forma que todas las entradas mínimas se colocan en la **X** máxima del screen y todas las entradas máximas se colocan en la **X** mínima del screen. En la figura 3.16 se muestra un polígono y su tabla de línea de barrido asociada, donde $\mathbf{X}_{max} = 10$.



SCAN LINE	XMIN	XMAX
•		
•		
•		
6	4	4
5	3	7
4	3	7
3	2	8
2	8	8
1	10	0
0	10	0

Figura 3.16

© Universidad de Las Palmas de Gran Canaria Biblioteca Digital, 2003

El primer paso en el algoritmo de relleno es dibujar el polígono. El siguiente paso será determinar los valores de **X** máximo y mínimo para cada línea de barrido asociada. Veamos como se puede calcular estos valores.

Si existe una arista no horizontal desde (X_1,Y_1) a (X_2,Y_2), la inversa de su pendiente será:

$$1/m = (X_2-X_1)/(Y_2-Y_1)$$

Si $Y_1 < Y_2$, el valor de **Y** se incrementa una unidad cuando subamos a la siguiente línea de barrido. El valor de **X** lo conocemos cuando **Y** iguale a $\mathbf{Y_1}$, es decir, sea $\mathbf{X_1}$.Para obtener el valor de **X** (X_{next}) perteneciente a la arista de la siguiente línea de barrido, $\mathbf{Y_1 + 1}$, se observa que:

$$m = (Y_1 + 1 - Y_1)/(X_{next}-X_1) = 1/(X_{next}-X_1)$$

ó

$$X_{next} = X_1 + 1/m$$

Esta función presenta alguna dificultad. Los valores de **X** son enteros mientras que la inversa de la pendiente, 1/m, es un valor real. Podríamos intentar corregir esto utilizando una función de redondeo:

$$X_{next} = \text{round}(X_1 + 1/m)$$

Esta corrección también sería incorrecta. Si 1/m es menor que 0.5, el valor X_{next} nunca se incrementa y siempre sería igual al valor inicial de X. Nuestro método de resolver este problema es mantener y actualizar un valor de **X** real. Para realizar esto, los valores de 1/m pueden acumularse, y la función round sólo se utiliza para dibujar el punto.

Generalizando este planteamiento, concluimos que la intersección **X** de la arista con la próxima línea de barrido se obtiene de la intersección **X** de la línea de barrido anterior y la pendiente de la arista:

$$X_{next} = X_{anterior} + 1/m$$

Este proceso comienza en $\mathbf{Y_1 + 1}$ y continua hasta que se alcance el extremo de la arista, $\mathbf{Y = Y_2}$. Si $\mathbf{Y_2 < Y_1}$ se utiliza un proceso similar decrementando el valor de **Y** en una unidad.

El Algoritmo Calcula_X determina el valor de la intersección **X** de una arista con una línea de barrido.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

```
Algoritmo Calcula_X

variables: xval : array [0..ymax] of integer
           xreal,m_inverse,temp : real
           Y,Ymax : integer


Ymax = 511
m_inverse = (X2-X1)/(Y2-Y1)

si Y1 > Y2 entonces

        temp = Y1
        Y1 = Y2
        Y2 = temp
        temp = X1
        X1 = X2
        X2 = temp

fin si

Xval[Y1] = X1
Xval[Y2] = X2
Xreal = X1
Y = Y1 + 1

mientras Y < Y2 hacer

        Xreal = Xreal + m_inverse
        Xval[Y] = round(Xreal)
        Y = Y + 1

fin mientras
```

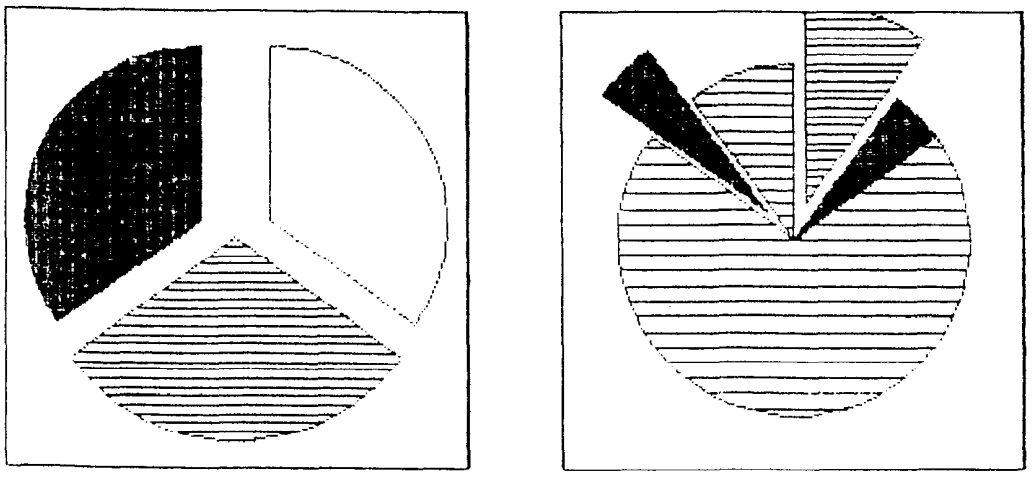
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Cuando se calcula un valor **X** de la arista, se compara con los valores mínimos y máximos para esa entrada **Y** en la tabla de líneas de barrido. Si el valor **X** es menor que el mínimo o mayor que el máximo, se inserta el valor **X** de la arista en la entrada apropiada.

Después de que cada una de las aristas se procesen y se coloquen los valores apropiados en la tabla de línea de barrido, el polígono está preparado para rellenarse. Esto se realiza dibujando líneas sobre la línea de barrido desde X_{min} a X_{max} siempre que $X_{min} < X_{max}$. Observe que $X_{min} > X_{max}$ sólo cuando esa línea está inicializada y, por lo tanto, una arista del polígono nunca será intersectada por esa línea de barrido. El algoritmo Convex_Fill realiza este paso final.

```
Algoritmo Convex_Fill

variables : Y : Integer

para Y = 0 hasta Ymax hacer

        con scan_table[Y] hacer

          si Xmin < Xmax entonces

            mover(Xmin,Y)
            dibujar(Xmax,Y)

          fin si

fin para
```

Se puede rellenar un polígono dibujando líneas alternadas reemplazando el bucle **para** por un **mientras** e incrementando el valor de **Y** por dos. Este algoritmo se utilizó para rellenar el diagrama de tarta de la figura 3.17a. Cada sector de la tarta es un polígono convexo horizontal y cada una de ellos puede rellenarse utilizando este algoritmo. Observe que el diagrama de tarta de la figura 3.17b no puede rellenarse utilizando este algoritmo ya que el trozo mayor no satisface el criterio de convexo.



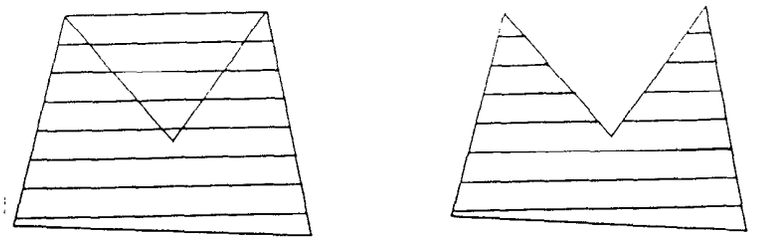
(a) (b)

Figura 3.17

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3.7.5 Rellenado de Polígonos Generales

La técnica de rellenado convexo trabaja bien para polígonos que tienen más de dos aristas sobre una línea de barrido. La figura 3.18a muestra los resultados incorrectos cuando este algoritmo se aplica a un polígono arbitrario. El gran problema es que el segmento de línea rellenado continua a través de los límites del polígono mientras se dibuja el valor **X** máximo.



(a) (b)

Figura 3.18

La figura 3.18b muestra que se rellenaría los pixels sobre una línea de barrido entre pares de aristas que son intersectadas por una línea de barrido. Un algoritmo que realiza esto tendrá los siguientes pasos:

1.- Computar los valores de la coordenada **X** de las intersecciones de la línea de barrido actual con todas las aristas.

2.- Ordenar estas intersecciones con las aristas por valores de **X** crecientes.

3.- Agrupar las intersecciones de los bordes por pares.

4.- Rellenar los pixels sobre la línea de barrido entre pares de valores de **X**.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Tabla de Intersección de Aristas

Una forma sencilla de computar los valores de **X** de los puntos de intersección de una línea de barrido y una arista es sustituir el valor actual de **Y** en la ecuación del segmento de la línea que representa la arista y resolverla para **X**. Un primer asunto es determinar primero que aristas han intersectado la línea de barrido actual. Un segundo asunto es que el cálculo de la ecuación del segmento de línea que define una arista puede ser muy largo en cuanto a tiempo. Para polígonos con muchas aristas, este cálculo puede retrasar el rellenado considerablemente. Existe una forma eficiente para realizar las intersecciones de las aristas que es similar a la técnica de obtener los valores de las aristas X_{min} y X_{max} para polígonos convexos.

Si **m** es la pendiente de una arista, entonces si X_{previo} es el valor de la intersección **X** (valor real) de la línea de barrido **Y** con la arista, entonces:

$$X_{next} = X_{previo} + 1/m \qquad (1)$$

es la intersección de la primera línea de barrido **Y + 1** con esa arista (figura 3.19). Por lo tanto, si conocemos la intersección de la línea de barrido más inferior con la arista, podemos utilizar la ecuación (1) para computar las intersecciones de las aristas de esas líneas de barrido. Esto indica que rellenaremos el polígono de abajo a arriba, moviendonos una línea de barrido hacia arriba en cada momento.

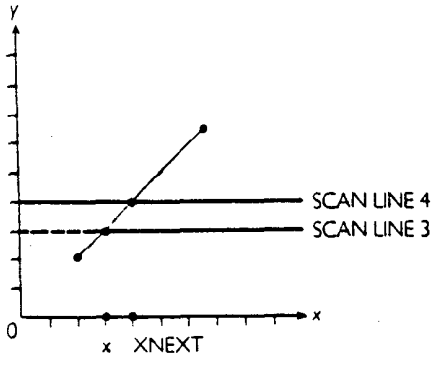


Figura 3.19

Todavía necesitamos un medio eficiente de determinar que aristas intersectan la línea de barrido. Para realizar esto, creamos una tabla de todas las aristas del polígono. La estructura de datos para esta tabla es un array de Registros, un Registro para cada línea de barrido. Cada Registro contiene aquellas aristas cuyos valores más pequeños de **Y** comienzan en una línea de barrido representada por ese Registro. Un Registro es un puntero que apunta a la lista de aristas. Se utiliza un puntero **nil** para Registros que no tienen asociado listas de aristas. La creación de esta estructura de datos se llama un Registro. La figura 3.21 muestra la estructura de datos de la tabla de aristas para el polígono de la figura 3.20

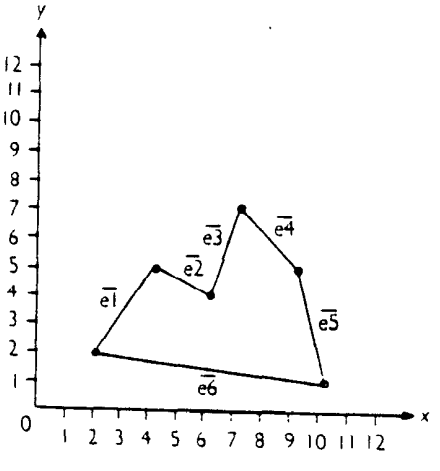


Figura 3.20

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2003

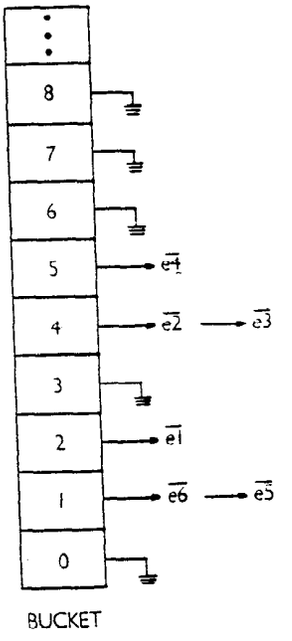


Figura 3.21

Una vez construida esta tabla, el algoritmo de rellenado la utiliza para determinar que aristas intersectan la línea de barrido actual y cual es la coordenada **X** de cada intersección. La coordenada **Y** de una intersección de la arista será el valor de la línea de barrido actual. Para facilitar este proceso, cada arista introducida en la tabla de aristas se representa por un registro. El primer campo contiene el valor de **Y** más superior de esa arista, es decir, el valor **Y** de la última línea de barrido intersectada. Llamemosle $\mathbf{Y_{top}}$. El segundo campo contiene el valor de la coordenada **X** del valor de **Y** más pequeño para esa arista, denotado por $\mathbf{X_{min}}$. El siguiente campo es la inversa de la pendiente de la arista, **m_inverse**. El último campo, **next**, es un puntero a la próxima arista en el mismo Registro **Y**. Las aristas dentro de un Registro son ordenadas por el campo $\mathbf{X_{min}}$ en orden creciente. La variable $\mathbf{X_{min}}$ es de tipo real ya que cambia según la ecuación 1.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Consideraciones de los Vértices

La tabla de aristas creada es adecuada para líneas de barrido que no intersectan un vértice. Las intersecciones de los vértices deben ser tratados separadamente y la tabla de aristas se modifica como corresponda. La figura 3.22 muestra este problema. La línea de barrido $Y = 2$ intersecta el polígono en las aristas e_1,e_2 y e_4 con valores de intersección **X** correspondiente 2,2 y 4. Agrupando estos pixels en pares incorrectamente resultará la iluminación del pixel (2,2) y aquellos pixels sobre la línea de barrido 2 desde $X = 4$ hasta el final del screen. Sin embargo, la línea de barrido $Y = 4$ intersecta el polígono en las aristas e_2 y e_3 con valores de intersección de **X** 4 y 4. Agrupando en pares, sólo un pixel (4,4) es correctamente iluminado.

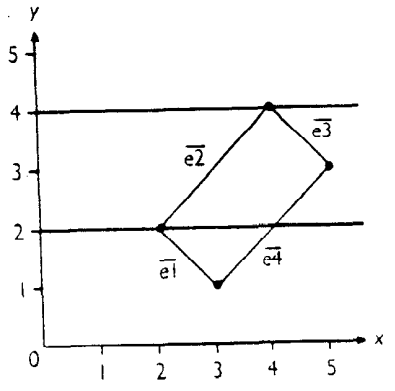


Figura 3.22

Para obtener los resultados correctos examinemos la figura 3.23, con la actual línea de barrido intersectando los vértices 1,2,3 y 4. Esto representa los distintos casos que debemos examinar.

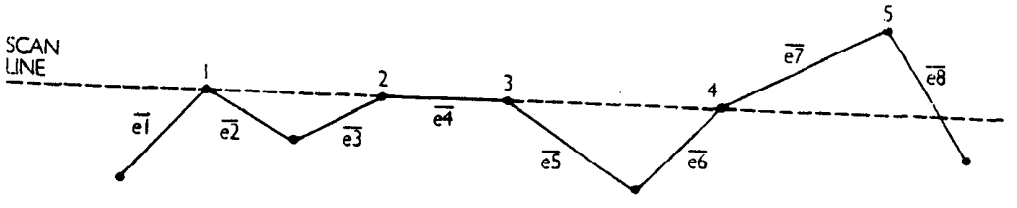


Figura 3.23

El vértice 1 conecta una arista de pendiente creciente (e_1) a una arista de pendiente decreciente (e_2). Un vértice con esta propiedad se llama un extremo. La arista que cumple un extremo puede ser insertada en la tabla de aristas sin procesamiento adicional. Es decir, ambas aristas son contadas resultando dos intersecciones con la línea de barrido.

Los vértices 2 y 3 son los puntos extremos de una arista horizontal e_4 teniendo pendiente cero. La línea de barrido actual intersecta la arista e_4 para diferentes valores de **X**. Aunque la arista e_4 sea eliminada del procesamiento, los vértices 2 y 3 son utilizados todavía por las aristas e_3 y e_5.

El vértice 4 no es un extremo local, y ese vértice debe contarse sólo en la tabla de aristas para la línea de barrido actual. (Es decir, el vértice 4 no debería incluirse en las aristas e_6 y e_7). Si lo contamos dos veces, el par de coordenadas **X** no sería rellenado entre el vértice 4 y la arista e_8. Para asegurar que el vértice 4 sólo se cuenta una vez, la arista e_7 se ordena por una unidad (pixel) en la dirección **Y**. El valor de la coordenada **X** del nuevo vértice 4'se calcula utilizando la ecuación (1). Por lo tanto, nunca nos encontraremos con

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

un vértice que no sea un extremo. (Figura 3.24).

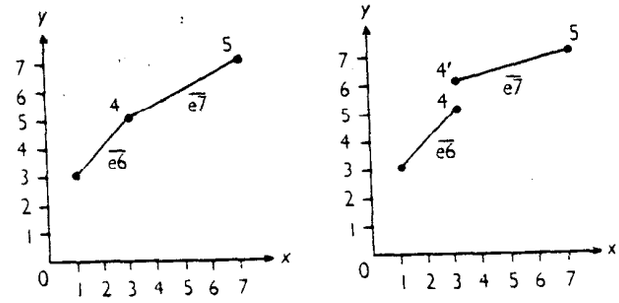


Figura 3.24

Algoritmo

Una vez construida la tabla, el algoritmo procede de la siguiente forma:

1.- Empieza con el valor **Y** más pequeño de la tabla de aristas cuya cubeta no esté vacía.

2.- Inicializa la cubeta de la línea de barrido a **nil**.

3.- Mientras el valor de la línea de barrido en curso **Y** no sea mayor que la línea superior del polígono, hacer:

a) Para el valor **Y** actual, fusionar la tabla de aristas con la lista de las líneas de barrido, manteniendo el orden creciente con respecto a los valores de $\mathbf{X}_{min}$.

b) Rellenar los pixels entre los pares de valores redondeados de $\mathbf{X}_{min}$ en la lista de la línea de barrido.

c) Eliminar las aristas de la lista de líneas de barrido cuyo valor **Y** máximo, $\mathbf{Y}_{top}$, sea igual al valor de la línea de barrido en curso.

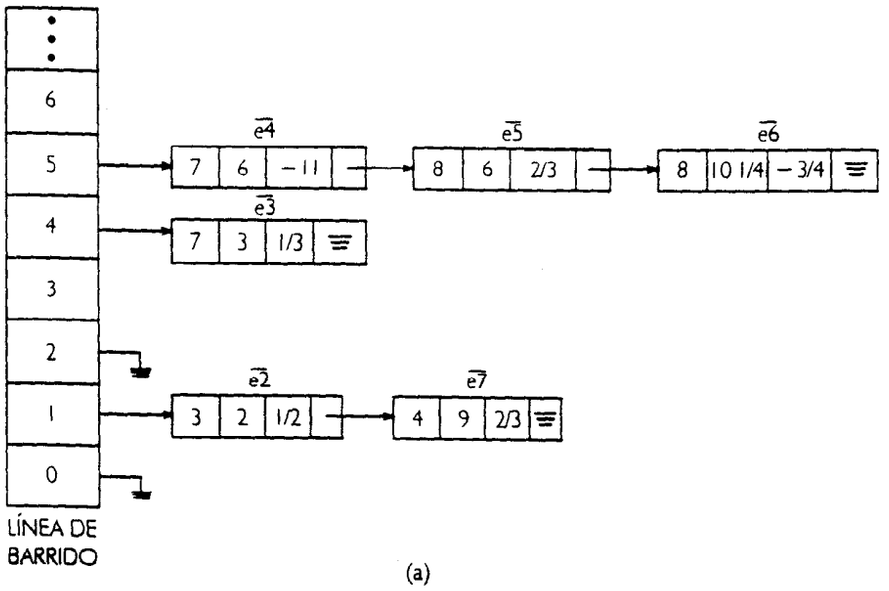
d) Incrementar en **m_inverse** el el valor de $\mathbf{X}_{min}$ para las aristas restantes en la lista de lñineas de barrido.

e) Reordenar la lista de lñineas de barrido en orden creciente con respecto a los valores de $\mathbf{X}_{min}$.

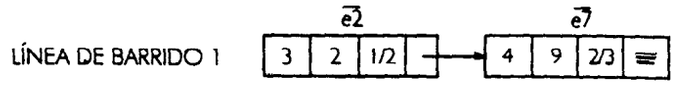
f) Incrementar en 1 el valor **Y** de la línea de barrido en curso para obtener la siguiente lñinea de barrido.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

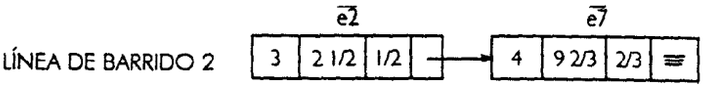
En la figura 3.25 se muestra un ejemplo siguiendo los pasos del algoritmo anterior.



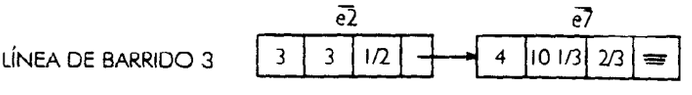
(a)



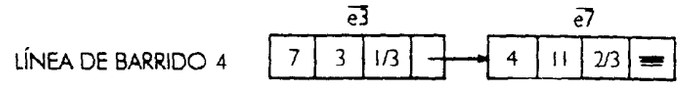
(b)



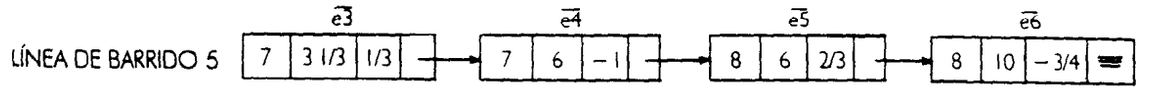
(c)



(d)



(e)



(f)

Figura 3.25

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3.7.6 Relleno con Patrones

En los métodos anteriores se iluminaban todos los pixels contenidos dentro de los límites de un polígono. Sin embargo, algunas aplicaciones requieren que se iluminen sólo aquellos pixels que correspondan a un patrón (figura 3.26). Tal extensión del relleno de polígonos también se puede usar para conferir textura a los polígonos.

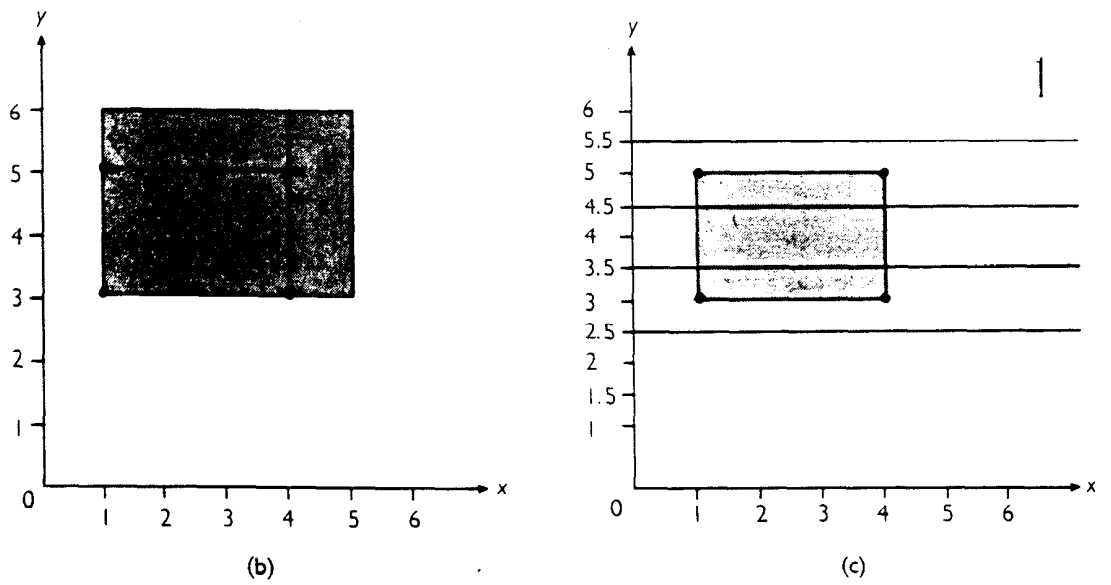


Figura 3.26

La figura 3.27a ilustra el patrón usado para rellenar el polígono de la figura 3.27b. Los pixels iluminados corresponden a valores verdaderos en el arreglo.

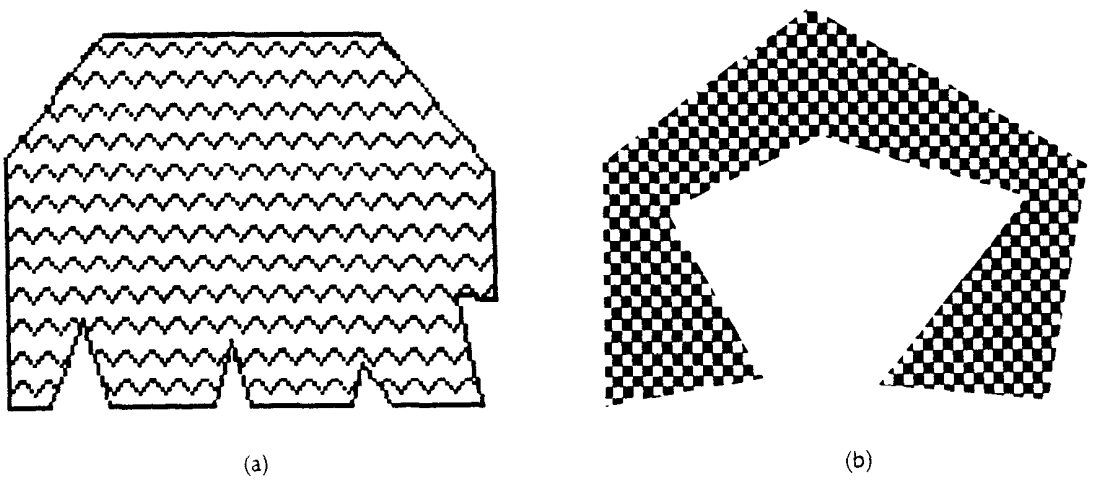


Figura 3.27

Conceptualmente, estos patrones son colocados uno tras otro para dar el efecto de

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

repetición que se ilustra en la figura 3.27b. Cada pixel contenido en el polígono es probado con su entrada correspondiente en el patrón repetido. Un pixel se ilumina sólo si su entrada correspondiente en el patrón tiene el valor booleano verdadero.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2003

CAPITULO 4

ALGORITMOS DEL INTERFAZ

4.1 RECORTE (CIIPPING)

El proceso del Recorte consiste en recortar las líneas que se encuentran fuera de una ventana dada mostrando solamente aquellas líneas que están dentro de ella. En el Recorte se examina cada una de las líneas del screen para determinar si está completamente dentro o fuera de la ventana. Si está dentro, la línea se dibuja; si está fuera, no se dibuja. Si cruza el límite, se debe determinar el punto de intersección y dibujar solamente la parte que está situada dentro. (Ver figura 4.1).

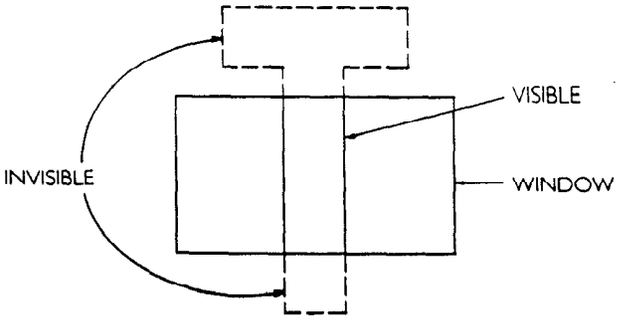


Figura 4.1

4.1.1 Recorte de Líneas en 2-D

La figura 4.2 muestra una escena en dos dimensiones y una ventana de recorte regular. Los límites en dos dimensiones está definido por Left (L), Right (R), Top (T) y Bottom (B). Una ventana de recorte regular es rectangular, con sus límites alineados con los del espacio de la figura o con la pantalla. El propósito de un algoritmo de recorte es determinar que puntos, líneas o partes de líneas están dentro de la ventana. Estos puntos, líneas o partes de líneas se dejan en la pantalla. Todas las demás se descartan.

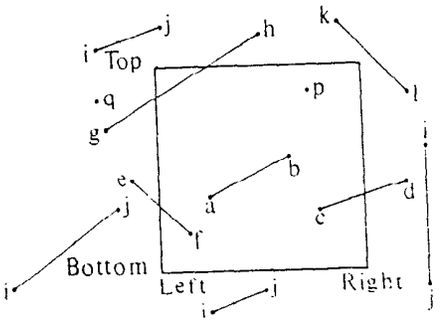


Figura 4.2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Cuando se tienen que recortar muchas líneas o puntos de una figura, es de gran interés la eficiencia de los algoritmos de recorte. En muchos casos, la gran mayoria de puntos o líneas son interiores o exteriores a la ventana. Por lo tanto, en la figura 4.2 es importante ser capaz de aceptar rápidamente una línea **ab** o un punto **p** o rechazar una línea **ij** o un punto **q**.

Los puntos son interiores a la ventana siempre que

$$X_l <= X <= X_r \quad y \quad Y_b <= Y <= Y_t$$

El signo igual indica que los puntos en los bordes de la ventana son incuidos dentro de la misma.

Las líneas son interiores a la ventana y por lo tanto visibles si ambos puntos extremos son interiores a la ventana, por ejemplo la línea **ab** en la figura 4.2. Sin embargo, si ambos puntos extremos de la línea son exteriores a la ventana, la línea no es necesariamente exterior a la ventana, por ejemplo , la línea **gh** en la figura 4.2. Si ambos puntos extremos de la línea están o completamente a la derecha, a la izquierda, abajo o arriba de la ventana, entonces la línea es completamente exterior a la ventana y por lo tanto invisible. Esta prueba eliminará todas las líneas etiquetadas **ij** en la figura 4.2. No eliminará las líneas **gh**, la cual es parcialmente visible, o la línea **kl**, la cual es totalmente invisible.

Si **a** y **b** son lon puntos extremos de una línea, entonces un algoritmo para identificar líneas completamente visibles y la mayoría de las líneas invisibles podria ser el siguiente:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Algoritmo de visibilidad simple

a y b son los puntos extremos de la línea con componentes X e Y.

Para cada línea, chequea si es totalmente visible. Si cualquiera de las coordenadas de los puntos extremos está fuera de la ventana, la línea no es totalmente visible.

si $X_a < X_l$ **or** $X_a > X_r$ **entonces** 1 **fin si**
si $X_b < X_l$ **or** $X_b > X_r$ **entonces** 1 **fin si**
si $Y_a < Y_b$ **or** $Y_a > Y_t$ **entonces** 1 **fin si**
si $Y_b < Y_b$ **or** $Y_b > Y_t$ **entonces** 1 **fin si**

La línea es totalmente visible.

Dibujar_Linea

go to 3

Chequeo para las líneas totalmente invisibles. Si ambos puntos extremos están a la izquierda, derecha, abajo o arriba de la ventana, la línea es trivialmente invisible.

1 **si** $X_a < X_l$ **and** $X_b < X_l$ **entonces** 2 **fin si**
si $X_a > X_r$ **and** $X_b > X_r$ **entonces** 2 **fin si**
si $Y_a > Y_t$ **and** $Y_b > Y_t$ **entonces** 2 **fin si**
si $Y_a < Y_b$ **and** $Y_b < Y_b$ **entonces** 2 **fin si**

La línea es parcialmente visible o la diagonal a través de la esquina invisible determina las intersecciones.

2 la línea es invisible.

3 próxima línea

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Aquí X_l, X_r, Y_t, Y_b son las coordenadas **X** e **Y**, respectivamente de los bordes izquierdo, derecho, superior e inferior de la ventana. El orden en el cual se realizan las pruebas para determinar la visibilidad o la invisibilidad es indiferente. Algunas líneas necesitarán los cuatro chequeos antes de aceptarse como totalmente visible o recortada o totalmente invisible. Otras necesitarán solamente un chequeo. Tambien es indiferente si se realiza primero el chequeo para las líneas totalmente invisibles o las líneas totalmente visibles. Sin embargo, los cálculos de intersección de la línea con el borde de la ventana es caro en cuanto al tiempo de computación y será realizado más tarde.

Las pruebas para las líneas totalmente visibles y para las regiones dadas anteriormente pueden formalizarse utilizando una técnica debida a Dan Cohen e Ivan Sutherland. La técnica utiliza un código de cuatro bits para indicar cual de las nueve regiones contiene el punto extremo de una línea. Los códigos de cuatro bits se muestra en la figura 4.3. El primer bit es el de más a la derecha. Los bits se colocan a 1 según el siguiente esquema:

Primer bit : si el punto extremo está a la izquierda de la ventana.

Segundo bi : si el punto extremo está a la derecha de la ventana.

Tercer bit : si el punto extremo está debajo de la ventana.

Cuarto bit : si el punto extremo está encima de la ventana.

En cualquier otro caso, el bit se coloca a cero. Es evidente que, si ambos códigos de los puntos extremos son ceros, entonces ambos extremos de la línea están dentro de la ventana y la línea es visible.

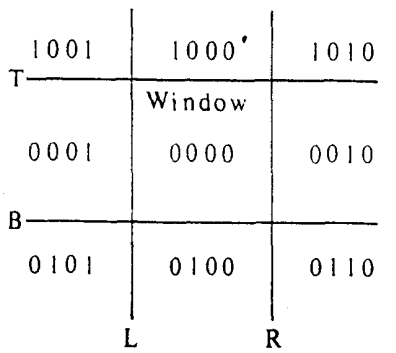


Figura 4.3

Los códigos de los puntos extremos tambien pueden utilizarse para rechazar las líneas totalmente invisibles. Considere la siguiente tabla:

Verdadero y Falso	——> Falso 1 AND 0	——> 0
Falso y Verdadero	——> Falso 0 AND 1	——> 0
Falso y Falso	——> Falso 0 AND 0	——> 0
Verdadero y Verdader	—> Verd. 1 AND 1	——> 1

la cual es equivalente al operador lógico **AND**. Si la intersección lógica bit a bit de los dos códigos de los puntos extremos no es cero, entonces la línea es totalmente invisible y puede ser rechazada. Los distintos ejemplos mostrados en la Tabla 4.1 ayuda a clarificar esto. Note en la Tabla 4.1 que, cuando la intersección lógica es distinta de cero, la línea es totalmente invisible. Sin embargo, cuando la intersección lógica es cero, la línea puede ser totalmente o parcialmente visible o totalmente invisible. Es por esta razón que es necesario chequear ambos códigos de los puntos extremos separadamente para determinar la total visibilidad.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Line	End point codes		Logical intersection	Comments
ab	0000	0000	0000	Totally visible
ij	0010	0110	0010	Totally invisible
ij	1001	1000	1000	Totally invisible
ij	0101	0001	0001	Totally invisible
ij	0100	0100	0100	Totally invisible
cd	0000	0010	0000	Partially visible
ef	0001	0000	0000	Partially visible
gh	0001	1000	0000	Partially visible
kl	1000	0010	0000	Totally invisible

Tabla 4.1

Puede implementarse fácilmente el chequeo de los códigos de los puntos extremos cuando se dispone de las rutinas de manipulación de bit. Una posible implementación software que no utiliza las rutinas de manipulación de bit se muestra en los siguientes algoritmos.

Primero se determina si las líneas son totalmente visibles o totalmente invisibles, entonces se pasan a la rutina de intersección aquellas líneas parcialmente visibles, para la cuales la intersección lógica de los códigos de los puntos extremos es cero. Esta rutina debe tambien, por supuesto, identificar las líneas totalmente invisibles que le son pasadas.

La intersección entre dos líneas puede determinarse paramétricamente o no. Explícitamente la ecuación de la línea infinita a través de $P_1(X_1,Y_1)$ y $P_2(X_2,Y_2)$ es:

$$Y = m(X - X_1) + Y_1 \quad o \quad Y = m(X - X_2) + Y_2$$

donde

$$m = (Y_2 - Y_1)/(X_2 - X_1)$$

es la pendiente de la línea. Las intersecciones con los bordes de la ventana estan dados por

Left : X_l, $Y = m(X_l - X_1) + Y_1$ $m <> oo$

Right : X_r, $Y = m(X_r - X_1) + Y_1$ $m <> oo$

Top : Y_t, $X = X_1 + (1/m)(Y_t - Y_1)$ $m <> 0$

Bottom : Y_b, $X = X_1 + (1/m)(Y_b - Y_1)$ $m <> 0$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

El ejemplo 4.1 muestra que el método explícito permite el rechazo de las intersecciones inadecuadas comparando simplemente los valores de la intersección con los bordes de la ventana.

EJEMPLO 4.1

Considere la ventana y las líneas mostradas en la figura 4.4. Para la línea desde $P_1(-3/2,1/6)$ a $P_2(1/2,3/2)$ la pendiente es

$$m = (Y_2-Y_1)/(X_2-X_1) = (3/2-1/6)/(1/2-(-3/2)) = 2/3$$

y las intersecciones con los bordes de la ventana son

Left: $X = -1 \quad Y = (2/3)[-1-(-3/2)]+1/6 = 1/2$

Right: $X = 1 \quad Y = (2/3)[1-(-3/2)]+1/6 = 11/6$

la cual es mayor que Y_t y por tanto rechazada.

Top: $Y = 1 \quad X = -3/2+(3/2)[1-1/6] = -1/4$

Bottom: $Y = -1 \quad X = -3/2+(3/2)+-1-(1/6)] = -13/4$

la cual es menor que X_l y por tanto rechazada.

Similarmente para la línea desde $P_3(-3/2,-1)$ a $P_4(3/2,2)$

$$m = (Y_2-Y_1)/(X_2-X_1) = (2-(-1))/(3/2-(-3/2)) = 1$$

y las intersecciones con los bordes de la ventana son

Left: $X = -1 \quad Y = (1)[-1-(-3/2)]+(-1) = -1/2$

Right: $X = 1 \quad Y = (1)[1-(-3/2)]+(-1) = 3/2$

la cual es mayor que Y_t y por tanto rechazada.

Top: $Y = 1 \quad X = -3/2+(1)[1-(-1)] = 1/2$

Bottom: $Y = -1 \quad X = -3/2+(1)[-1-(-1)] = -3/2$

la cual es menor que X_l y por tanto rechazada.

En el desarrollo de la estructura de un algoritmo de recorte eficiente se puede considerar algunos casos especiales. Volviendo a la discusión anterior, si la pendiente de la línea es infinita, es paralela a los bordes izquierdo y derecho, y solo necesita chequearse para la intersección los bordes superior e inferior de la ventana. De la misma forma, si la pendiente es cero, la línea es paralela a los bordes superior e inferior de la ventana y necesita chequearse para la intersección solamente los bordes izquierdo y derecho de la ventana. Finalmente, si uno de los códigos de los puntos extremos es cero, ese punto

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

extremo es interior a la ventana y solo puede ocurrir una intersección. A continuación se verá una implementación.

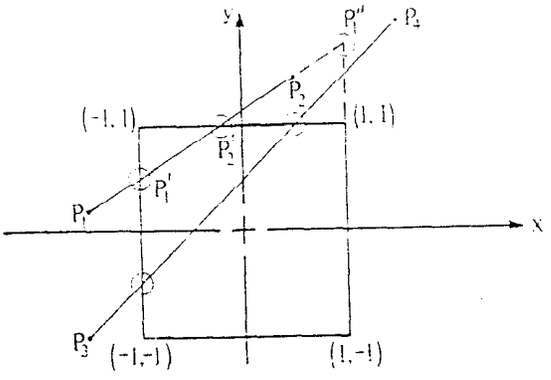


Figura 4.4

Algoritmo de Recorte de Sutherland-Cohen de subdivisión de la línea

El algoritmo anterior es similar a uno desarrollado por Dan Cohen e Ivan Sutherland. El algoritmo anterior recorta la línea sucesivamente con cada uno de los bordes de la ventana y examina el punto de intersección resultante para ver si está dentro de la ventana. Esto se realiza primero para la línea P_1 P_2 para proporcionar P_1' P_2 y entonces para la línea P_2 P_1' para proporcionar P_2' P_1', la línea recortada.

El algoritmo de Sutherland-Cohen también divide la línea en un borde de la ventana. Por contra no la chequea para ver si el punto de intersección está dentro de la ventana ya que intenta aceptar o rechazar los dos segmentos resultantes usando los códigos de los puntos extremos de la línea. Volviendo con la línea P_1 P_2 de la figura 4.4 revela inmediatamente una dificultad con esta simple técnica. Si P_1 P_2 se recorta con el borde izquierdo de la ventana, los dos nuevos segmentos son P_1 P_1' y P_1' P_2. Los códigos de los puntos extremos para ambos segmentos indican que ambos pueden ser parcialmente visibles. Por lo tanto, ninguno de ellos pueden rechazarse como invisibles o aceptados como visibles. De forma general, el algoritmo de Sutherland-Cohen es:

Para cada borde de la ventana:

Para la línea P_1 P_2, determina si la línea es totalmente visible o puede ser rechazada como invisible.

Si P_1 está fuera de la ventana continua; en cualquier otro caso, intercambia P_1 y P_2.

Reemplaza P_1 con la intersección de P_1 P_2 y el borde de la ventana.

El ejemplo 4.2 ilustra el algoritmo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Ejemplo 4.2

Consideremos la línea P_1 P_2 recortada con la ventana mostrada en la figura 4.4. Los códigos de los puntos extremos para $P_1(-3/2,1/6)$ y $P_2(1/2,3/2)$ son (0001) y (1000) respectivamente. La línea no es ni totalmente visible ni trivialmente invisible. P_1 está fuera de la ventana.

La intersección con el borde izquierdo de la ventana ($X = -1$) es $P_1'(-1,1/2)$. Reemplaza P_1 con P_1' produciendo la nueva línea $P_1(-1,1/2)$ a $P_2(1/2,3/2)$.

Los códigos de los puntos extremos para P_1 y P_2 son ahora (0000) y (1000), respectivamente. La línea no es ni totalmente visible ni trivialmente invisible.

P_1 está dentro de la ventana. Intercambia P_1 y P_2 para producir la nueva línea $P_1(1/2,3/2)$ a $P_2(-1,1/2)$. También intercambia los códigos de los puntos extremos.

La intersección con el borde derecho de la ventana($X = 1$) es $P''(1,11/6)$. Reemplazando P_1 con P'' produce la nueva línea $P_1(1,11/6)$ a $P_2(-1,1/2)$

Los códigos de los puntos extremos para P_1 y P_2 son ahora (1000) y (0000), respectivamente. La línea nos es ni totalmente visible ni trivialmente invisible.

P_1 está fuera de la ventana.

La intersección con el borde superior de la ventana ($Y = 1$) es $P_2'(-1/4,1)$. Reemplazando P_1 con P_2' produce la nueva línea $P_1(-1/4,1)$ a $P_2(-1,1/2)$.

Los códigos de los puntos extremos para P_1 y P_2 son (0000) y (0000), respectivamente. La línea es totalmente visible.

El procedimiento se completa.

Dibujar la línea.

Algoritmo de Subdivisión del Punto Medio

El algoritmo anterior requiere el cálculo de la intersección de la línea con el borde de la ventana. El cálculo directo puede evitarse realizando una búsqueda binaria para la intersección dividiendo la línea en sus puntos medios. El algoritmo, que es un caso especial del algoritmo de Sutherland-Cohen, fue propuesto por Sproull y Sutherland para la implementación en hardware. La implementación en software del algoritmo es más lenta que utilizando el cálculo directo de la intersección de la línea con el borde de la ventana como se vio anteriormente. La implementación en hardware es más rápida y eficiente debido a la arquitectura paralela que puede utilizarse y la suma y división por 2 es más rápida en hardware. En hardware, la división por 2 puede realizarse moviendo cada bit a la derecha. Por ejemplo, la representación binaria de 4 bits del número decimal 6 es 0110. Moviendo cada bit a la derecha produce el 0011 que es el número decimal 3 = 6/2.

El algoritmo utiliza los códigos de los puntos extremos de la línea y las pruebas asociadas para identificar inmediatamente las líneas totalmente visibles, por ejemplo, la línea **a** de la figura 4.5, y las líneas trivialmente invisibles, por ejemplo la línea **b** de la figura

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

4.5. Las líneas que no pueden identificarse inmediatamente utilizando estas pruebas, por ejemplo las líneas **c** a **g** de la figra 4.5, son subdivididas en dos partes iguales. Entonces, las pruebas se aplican para cada mitad hasta que se encuentre la intersección con el borde de la ventana o la longitud de los segmentos divididos es infinitesimal, por ejemplo, la línea **f** de la figura 4.5. Entonces se determina la visibilidad de los puntos. El resultado es realizar una búsqueda logarítmica para el punto de intersección. El número máximo de subdivisiones es igual a la precisión de la representación de los puntos extremos de la línea. Para ilustrar la técnica considere las líneas **c** y **f** de la figura 4.5. Aunque la línea **f** no es visible, atravieza la esquina y no puede ser trivialmente rechazada. La subdivisión en el punto medio P_m1 permite que la mitad P_m1 P_2 sea trivialmente rechazada. La mitad P_m1 P_1 atravieza de nuevo la esquina y no pueden ser trivialmente rechazada. Además, la subdivisión en P_m2 permite rechazar P_m2 P_1 como invisible. La subdivisión de los restantes P_m1 P_m2 continua hasta que se encuentre la intersección de la línea con la extensión del borde de la ventana de la derecha dentro de una precisión especificada. Entonces se examina este punto y se observa que es invisible. Por lo tanto, toda la línea es invisible.

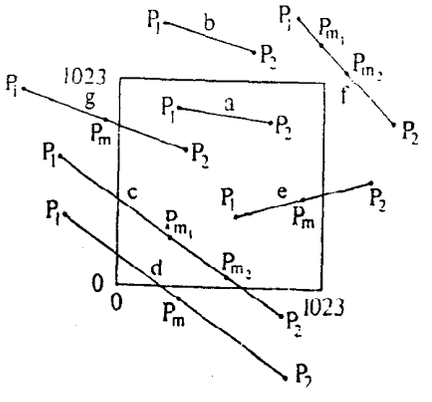


Figura 4.5

A partir de los códigos de los puntos extremos de la línea **c** de la figura 4.5, ni es totalmente visible ni puede ser trivialmente rechazada como invisible. La subdivisión en el punto medio P_m1 produce algunos resultados para ambas mitades. Colocando aparte el segmento P_m1 P_1 para su posterior consideración, el segmento P_m1 P_2 se subdivide en P_m2. Ahora el segmento P_m1 P_m2 es totalmente visible, y el segmento P_m2 P_2 parcialmente visible. El segmento P_m1 P_m2 podría dibujarse. Sin embargo, esto resultaría en la porción visible de la línea que se está dibujando como una serie de cortos segmentos. En su lugar el punto P_m2 se recuerda como el punto visible más lejos actual de P_1. La subdivisión del segmento P_m2 P_2 continua. Cada uno de los puntos medios visibles que se encuentran se declara el punto visible más lejos actual de P_1, hasta que la intersección con el borde inferior de la ventana se determine con alguna precisión determinada. Entonces esta intersección se declara el punto visible más lejos de P_1. A continuación el segmento P_m1 P_1 se examina de la misma forma. Para la línea **c** de la figura 4.5 el punto visible más lejos de P_2 es la intersección con el borde de la ventana de la izquierda. Entonces la parte visible de la línea P_1 P_2 se dibuja entre las dos intersecciones.

Para las líneas **c** y **d** de la figura 4.5 el algoritmo de subdivisión del punto medio realiza dos búsquedas logarítmicas para los puntos visibles más lejanos de los extremos de la línea.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

El algoritmo puede formalizarse en tres pasos. Para cada uno de los puntos extremos:

si el punto extremo es visible, entonces es el punto visible más lejano. El proceso se completa. Si no, continua.

Si la línea es trivialmente invisible, no se genera salida. El proceso se completa. Si no, continua.

Calcula el punto visible más lejano dividiendo la línea P_1 P_2 en su punto medio P_m. Aplicar las pruebas sobre los dos segmentos P_1 P_m y P_m P_2. Si P_m P_2 se rechaza trivialmente como invisible, el punto medio es una sobreestimación del punto visible más lejano. Continua con P_1 P_m. En cualquier otro caso, el punto medio es una subestimación del punto visible más lejano. Continua con P_2 P_m. Si el segmento llega a ser tan corto que el punto medio corresponde a la precisión se la máquina o a los puntos extremos, evalua la visibilidad del punto y se completa el proceso.

Un ejemplo específico ilustra mejor el algoritmo.

Ejemplo 4.3. Subdivisión del Punto Medio

Considere la ventana en las coordenadas del screen mostrada en la figura 4.5 que tiene los bordes izquierdo, derecho, inferior y superior a 0, 1023, 0, 1023, respectivamente. La línea **c** tiene los puntos extremos P_1(-307,631) y P_2(820,-136) en coordenadas del screen. Los códigos de los puntos extremos para P_1 es (0001) y para P_2 es (0100). Ambos códigos de los puntos extremos son distintos de cero, así que la línea no es totalmente visible. La intersección lógica de los códigos de los puntos extremos es (0000). La línea no puede ser rechazada trivialmente como invisible. Miremos las intersecciones.

El punto medio es:

$$X_m = (X_2 + X_1)/2 = (820 - 307)/2 = 256.5 = 256$$

$$Y_m = (Y_2 + Y_1)/2 = (-136 + 631)/2 = 247.5 = 247$$

El código del punto extremo para el punto medio es (0000). Los segmentos P_1 P_m y P_2 P_m no son ni totalmente visible ni trivialmente invisible. Colocando aparte el segmento P_2 P_m y continuando con P_1 P_m el proceso de subdivisión continua como muestra la tabla 4.2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

P_1	P_2	P_m	Comment
−307, 631	820, −136	256, 247	Save P_mP_2, continue P_1P_m
−307, 631	256, 247	−26, 439	Continue P_mP_2
−26, 439	256, 247	115, 343	Continue P_1P_m
−26, 439	115, 343	44, 391	Continue P_1P_m
−26, 439	44, 391	9, 415	Continue P_1P_m
−26, 439	9, 415	−9, 427	Continue P_mP_2
−9, 427	9, 415	0, 421	Success
256, 247	820, −136	538, 55	Recall saved P_mP_2, continue P_mP_2
538, 55	820, −136	679, −41	Continue P_1P_m
538, 55	679, −41	608, 7	Continue P_mP_2
608, 7	679, −41	643, −17	Continue P_1P_m
608, 7	643, −17	625, −5	Continue P_1P_m
608, 7	625, −5	616, 1	Continue P_mP_2
616, 1	625, −5	620, −2	Continue P_1P_m
616, 1	620, −2	618, −1	Continue P_1P_m
616, 1	618, −1	617, 0	Success

Tabla 4.2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

4.1.2 Recorte de Polígonos

El recorte de líneas es aceptable cuando la salida constituye un conjunto de líneas desconectadas. Sin embargo, existen situaciones en las cuales un polígono recortado con una ventana, el resultado sería uno o más polígonos. Por ejemplo, la figura 4.6a-4.6b muestran dos polígonos y el resultado de recortar dichos polígonos. Observe que aquellas líneas del polígono recortado que coincide con los límites de la ventana no aparecerían si se utilizara el recorte de líneas.

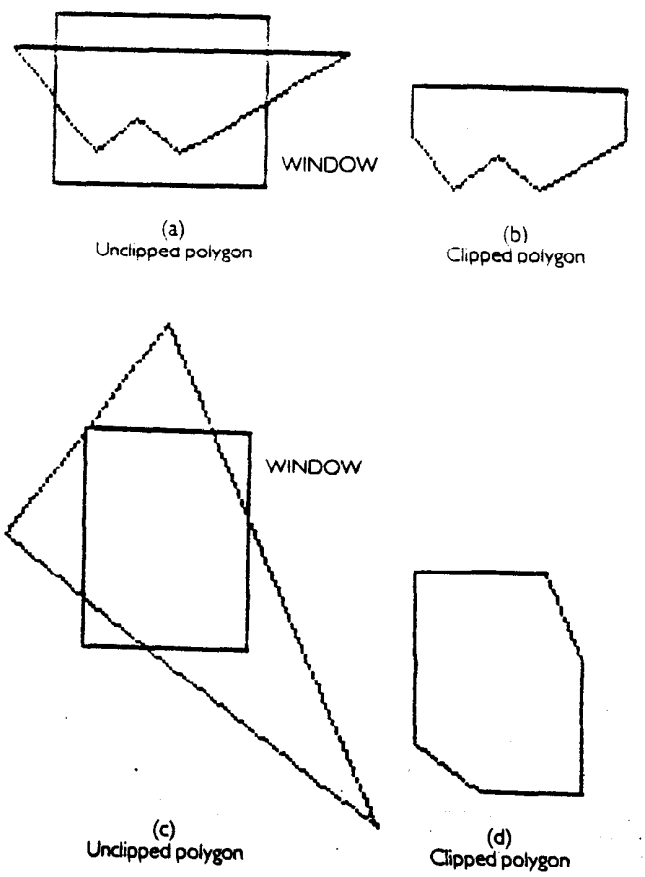


Figura 4.6

El algoritmo de recorte de polígonos de Sutherland-Hodgman recorta un polígono con cada uno de los bordes de la ventana. Específicamente, para cada uno de los bordes de la ventana se introduce una lista de vértices y se saca una lista de nuevos vértices, los cuales se someten al algoritmo para recortar con el próximo borde de la ventana. La lista de entrada es una secuencia de vértices consecutivos del polígono obtenido del recorte con el borde anterior. El primer borde de la ventana tiene como entrada en conjunto de vértices originales. Después de recortarlos con el último borde, la lista de salida consiste de un conjunto de vértices que determinan el polígono recortado.

El algoritmo testea un par de vértices consecutivos con un borde de la ventana. Existen cuatro casos posibles, mostrados en la figura 4.7, donde dicho testeo se realiza con el borde izquierdo. El polígono del ejemplo tiene vértices (v_1,v_2,v_3,v_4). El caso 1 tiene el primer y segundo vértice, v_1 y v_2, dentro de la ventana y el segundo vértice, v_2, se envía a la lista de salida. El caso 2 tiene el primer vértice, v_2, dentro de la ventana y el segundo vértice, v_3, fuera de la ventana. El punto de intersección, i_1, del lado del polígono junto con los vértices y el borde se añaden a la lista de salida. El caso 3 tiene ambos vértices, v_3 y v_4, fuera de la ventana y no se saca ningún punto. El caso 4 tiene el primer vértice, v_4, fuera y el segundo vértice, v_1, dentro de la ventana. El punto de intersección, i_2, y el segundo vértice, v_1, se añaden a la lista de salida. El resultado de este recorte con el borde izquierdo es la tranformación de la lista de entrada [v_1,v_2,v_3,v_4] a la lista de salida [v_2,i_1,i_2,v_1]. La figura 4.8 muestra los sucesivos recortes con los cuatro límites de la ventana.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

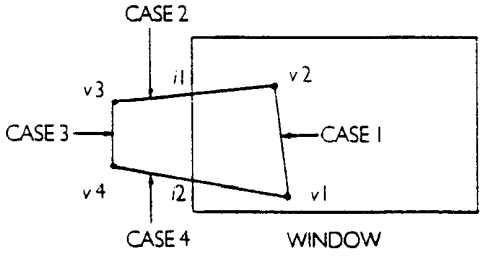


Figura 4.7

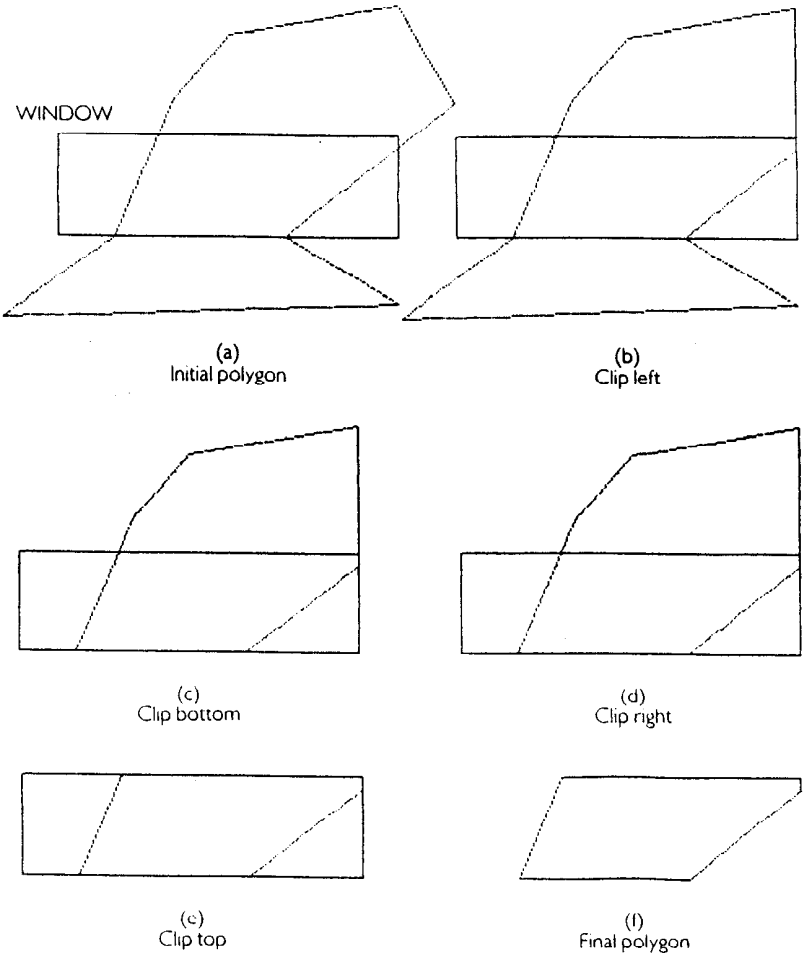


Figura 4.8

4.1.3 Genaralización del Recorte de líneas para Regiones Convexas

Los métodos de recorte vistos hasta ahora suponían que la ventana de recorte era un polígono rectangular. Para muchas aplicaciones la ventana de recorte no será un polígono

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

rectangular por lo que se han de aplicar otros algoritmos. Uno de ellos fué desarrollado por Cyrus-Beck para el recorte de cualquier región convexa arbitraria.

Antes de exponer el algoritmo vamos a considerar el recorte de una línea definida de forma paramétrica. La ecuación paramétrica del segmento de línea $P_1\ P_2$ es:

$$P(t) = P_1 + (P_2 - P_1)t \quad 0 <= t <= 1 \qquad (4.1)$$

donde t es el parámetro. Restringiendo el rango de t a $0<=t<=1$ hacemos que el segmento de línea no sea una línea infinita. La descripción paramétrica de una línea es independiente de cualquier sistema de coordenadas. Estos atributos la hacen particularmente eficaz para la determinación de la intersección entre una línea y el borde de un polígono convex arbitrario.

Para el sistema de coordenadas cartesianas, la ecuación (4.1) produce un par de ecuaciones paramétricas, una para cada coordenada:

$$x(t) = x_1 + (x_2 - x_1)t \quad 0 <= t <= 1 \qquad (4.2a)$$

$$y(t) = y_1 + (y_2 - y_1)t \quad 0 <= t <= 1 \qquad (4.2b)$$

Para una ventana de recorte rectangular, una de las coordenadas de la intersección con cada borde es conocida. Sólo es necesario calcular la otra. De la ecuación (4.1) el valor del parámetro t para cualquier punto sobre el segmento de línea es:

$$t = (P(t) - P_1) / (P_2 - P_1)$$

De la ecuación (4.2) el valor específico de t correspondiente a la intersección con el borde de la ventana es:

Para el borde izquierdo : $t = (xl-x_1)/(x_2-x_1)$ $0<=t<=1$
Para el borde derecho : $t = (xr-x_1)/(x_2-x_1)$ $0<=t<=1$
Para el borde superior : $t = (y_t-y_1)/(y_2-y_1)$ $0<=t<=1$
Para el borde inferior : $t = (y_b-y_1)/(y_2-y_1)$ $0<=t<=1$

donde xl, xr, y_b, y_t son las coordenadas de los bordes izquierdo, derecho, inferior y superior respectivamente. Si las soluciones de esas ecuaciones dan valores de t fuera del rango $0<=t<=1$, entonces esas soluciones serán descartadas ya que representan a los puntos más allá de los límites del segmento de línea.

Ejemplo: Línea Parcialmente Visible.

Consideremos la línea parcialmente visible $P_1(-3/2,-3/4)$ $P_2(3/2,1/2)$ recortada con la ventana $(xl,xr,y_b,y_t) = (-1,1,-1,1)$ como se muestra en la figura 4.9.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

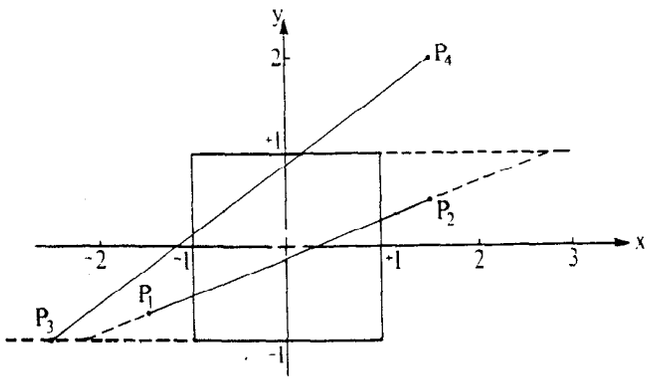


Figura 4.9

Para el borde izquierdo : $t=(xl-x_1)/(x_2-x_1) = 1/6$
Para el borde derecho : $t=(xr-x_1)/(x_2-x_1) = 5/6$
Para el borde inferior : $t=(y_b-y_1)/(y_2-y_1) = -1/5$
es <0 y es rechazada.
Para el borde superior : $t=(y_t-y_1)/(y_2-y_1) = 7/5$
es >1 y es rechazada

La porción visible de la línea es entonces $1/6 <= t <= 5/6$.

Las coordenadas x e y de los puntos de intersección son obtenidas de las ecuaciones paramétricas. En particular, para $t=1/6$ la ecuación (4.2) produce:

$$x(1/6) = -3/2 + [3/2 - (-3/2)](1/6) = -1$$
$$y(1/6) = -3/4 + [1/2 - (-3/4)](1/6) = -13/14$$

Similarmente para $t=5/6$ tenemos:

$$[x(5/6)\ y(5/6)] = [-3/2-3/4] + [3/2-(-3/2)\ \ 1/2-(-3/4)](5/6) = [1\ \ 7/24]$$

De este ejemplo parece que la técnica de recorte es sencilla y eficaz. Sin embargo, los siguientes ejemplos mostrarán algunas dificultades que se presentan.

Ejemplo: Línea Parcialmente Visible.

Consideremos la línea $P_3(-5/2,-1)$ $P_4(3/2,2)$ también mostrada en la figura 4.9 y de nuevo recortada con la ventana (-1,1,-1,1). Aquí los puntos de intersección estan dados por los valores paramétricos:

$t_i = 3/8$ $t_f = 7/8$ $t_b = 0$ $t_t = 2/3$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

los cuales están todos dentro del rango $0 <= t <= 1$.

Es ya conocido que, si un segmento línea intersecta a un polígono convexo, no lo puede hacer en más de dos puntos. Por lo tanto, solo se necesitan dos de los cuatro valores paramétricos hallados en el ejemplo anterior. Reordenando estos cuatro valores de menor a mayor tenemos t_b, t_l, t_t, t_r. Examinando la figura 4.9 se ve que los valores requeridos son $t_l = 3/8$ y $t_t = 2/3$ que producen la intersección en los puntos (-1,1/8) y (1/6,1), respectivamente. Esos valores son el máximo mínimo y el mínimo máximo de los valores paramétricos t. La determinación formal de esos valores es un simple problema clásico en la programación lineal.

Como en cualquier algoritmo de recorte, es importante la habilidad para identificar rápidamente y separar las líneas totalmente visibles y totalmente invisibles. En los próximos ejemplos se indican otras dificultades.

Ejemplo: Líneas Totalmente Visibles.

Consideremos la línea totalmente visible $P_1(-1/2,1/2)$ $P_2(1/2,-1/2)$ de nuevo recortada con la ventana (-1,1,-1,1) (Figura 4.10). Los valores paramétricos hallados son:

$t_l = -1/2 \quad t_r = 3/2 \quad t_b = 3/2 \quad t_t = -1/2$

Todos estos valores están fuera del rango $0 <= t <= 1$.

Del ejemplo anterior parece que se ha encontrado una técnica para identificar las líneas totalmente visibles. Sin embargo, el próximo ejemplo ilustra que este no es el caso.

Ejemplo: Líneas Totalmente Invisibles.

Consideremos la línea invisible $P_3(3/2,-1/2)$ $P_4(2,1/2)$ también mostrada en la figura 4.10. La ventana es de nuevo (-1,1,-1,1). Aquí los valores paramétricos para la intersección con los bordes de la ventana son:

$t_l = -5 \quad t_r = -1 \quad t_b = -1/2 \quad t_t = 3/2$

De nuevo todos los valores están fuera del rango $0 <= t <= 1$.

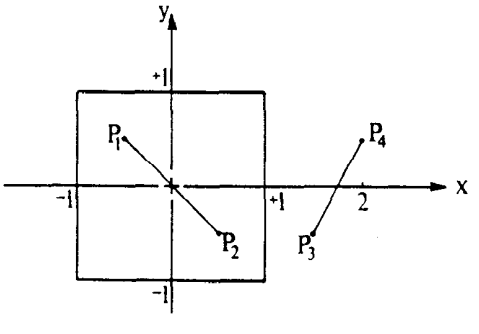


Figura 4.10

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2003

Algoritmo de Cyrus-Beck.

Para desarrollar un algoritmo de recorte eficaz es necesario encontrar una técnica segura para determinar si un punto sobre la línea está dentro, sobre, o fuera de la ventana. El algoritmo de Cyrus-Beck usa el vector normal para conseguirlo.

Consideremos una región de recorte convexa R. Aunque no limitamos a que R sea una región 2D, lo supondremos en los ejemplos presentados en este apartado. Un vector normal para cualquier punto a sobre los límites de R está dado por el producto escalar de vectores:

$$n*(b-a) >= 0$$

donde b es cualquier otro punto sobre los límites de R. Para ver esto, recordamos que el producto escalar de dos vectores V_1 y V_2 está dado por:

$$V_1 * V_2 = |V_1| * |V_2| * \cos\theta$$

donde θ es el menor de los ángulos formados por V_1 y V_2. Note que si $\theta = \pi/2$ entonces $\cos\theta = 0$ y $V_1 * V_2 = 0$; esto es, cuando el producto escalar de dos vectores es cero, los dos vectores son perpendiculares. La figura 4.11 muestra una región convexa R, donde están los vectores normales de entrada n_i y de salida n_o del punto a, junto con otros vectores que van del punto a a otros puntos sobre los límites de R. El ángulo entre n_i y cualquiera de esos vectores está siempre entre el intervalo $-\pi/2 <= \theta <= \pi/2$. En este intervalo el coseno es siempre positivo. Por lo tanto, el producto escalar es siempre positivo, como planteamos antes. Sin embargo, el ángulo entre n_o y cualquiera de esos vectores es siempre $\pi-\theta$ y el coseno$(\pi-\theta)$ = -cosenoθ es siempre negativo. Para ilustrar mejor esto, consideremos el siguiente ejemplo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

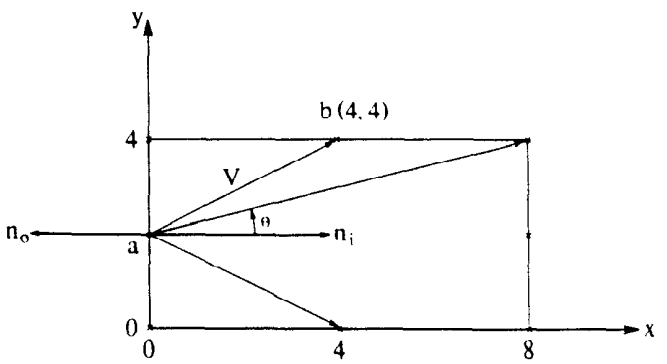


Figura 4.11

Ejemplo: Vectores Normales de Entrada y Salida.

Consideremos la región rectangular en la figura 4.11. Aquí los vectores normales son $n_i = i$ y $n_o = -i$, donde i es el vector unitario en la dirección de X. La tabla 4.3 muestra los valores del producto escalar de ambos vectores normales y los vectores que van desde el punto a a varios puntos b sobre los bordes de R. Como ejemplo particular: En el punto a

el vector normal de entrada es $n_i = i$. Sea el vector a(0,2)-b(4,4), que es igual a b-a = 4i + 2j; entonces el producto escalar es: $n_i*(b-a) = i*(4i + 2j) = 4$.

a	b	$\mathbf{n}_i \cdot (\mathbf{b} - a)$	$\mathbf{n}_o \cdot (\mathbf{b} - \mathbf{a})$
(0, 2)	(0, 4)	0	0
	(4, 4)	4	−4
	(8, 4)	8	−8
	(8, 2)	8	−8
	(8, 0)	8	−8
	(4, 0)	4	−4
	(0, 0)	0	0

Tabla 4.3

Los valores cero en la Tabla 4.3 indican que los vectores normales son perpendiculares.

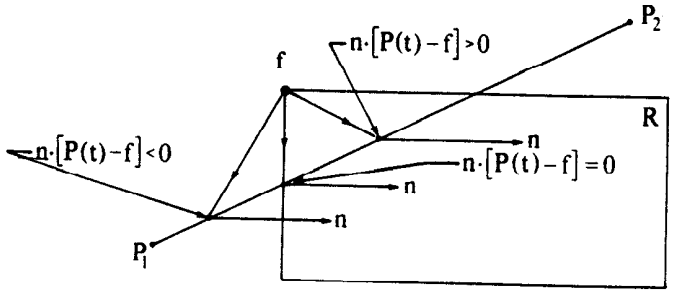


Figura 4.12

Volviendo a la determinación de la intersección de una línea y una ventana, consideramos de nuevo la representación de la línea P_1,P_2:

$$P(t) = P_1 + (P_2 - P_1)t \qquad 0 <= t <= 1$$

Si f es un punto en el límite de la región convexa R y n es un vector normal de entrada para uno de sus bordes, entonces para cualquier valor particular de t, esto es, cualquier punto de la línea P_1P_2,

$$n*[P(t)-f] < 0$$

implica que el vector P(t)-f apunta hacia el exterior de R.

$$n*[P(t)-f] = 0$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

implica que P(t)-f apunta paralelamente hacia el plano que contiene a f y perpendicular a la normal.

$$n*[P(t)-f] > 0$$

implica que P(t)-f apunta hacia el interior de R como se ilustra en la figura 4.12. Junto con estas condiciones mostramos que, si una región convexa R está cerrada, una línea infinita que intersecte a la región lo va hacer por dos puntos. Es más, esos dos puntos no están sobre el mismo borde. Entonces,

$$n*[P(t)-f] = 0$$

tiene una única solución. Si el punto f está en el borde para el cual n es el vector normal de entrada, entonces aquel punto t sobre la línea P(t) que satisfaga esta condición, será la intersección de la línea y el plano del borde.

Ejemplo: Recorte por Cyrus-Beck. Líneas Parcialmente Visibles.

Consideremos la línea $P_1(-1,1)$ y $P_2(9,3)$ recortada por la región rectangular mostrada en la figura 4.13. La ecuación de la línea P_1P_2 es $y=0.2(x+6)$ que intersecta la ventrana en (0,1.2) y (8,2.8). La representación paramétrica de la línea P_1P_2 es:

$$P(t)=P_1+(P_2-P_1)t=[-1\ \ 1]+[10\ \ 2]t=(10t-1)i+(2t+1)j \quad 0<=t<=1$$

donde i,j son los vectores unitarios de las direcciones x,y respectivamente. Los cuatro vectores normales de entrada son:

$$\begin{aligned} &\text{Izquierdon}_l = i \\ &\text{Derecho} \ : \ n_r = -i \\ &\text{Inferior} \ : \ n_b = j \\ &\text{Superior} \ : \ n_t = -j \end{aligned}$$

Eligiendo f(0,0) para el borde izquierdo, tenemos:

$$P(t)-f=(10t-1)i+(2t+1)j \quad y \quad n_l*[P(t)-f]=10t-1=0$$

$$--> \ t=1/10$$

es la intersección de la línea y el borde izquierdo de la ventana de recorte. Por lo tanto,

$$P(1/10)=[-1\ \ 1]+[10\ \ 2](1/10)=[0\ \ 1.2]$$

que coincide con el calculado explicitamente.

Eligiendo f(8,4) para el borde derecho, tenemos:

$$P(t)-f=(10t-9)i+(2t-3)j \quad y \quad n_r*[P(t)-f]=-(10t-9)=0$$

$$--> \ t=9/10$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

como el punto de intersección de la línea y el borde derecho.

Específicamente

$$P(9/10) = [-1\ \ 1] + [10\ \ 2](9/10) = [8\ \ 2.8]$$

que es tambien igual al calculado explicitamente.

Usando f(0,0) para el borde inferior, tenemos:

$$n_b * [P(t) - f] = -(2t + 1) = 0 \quad \text{--> } t = -1/2$$

que está fuera del rango $0 <= t <= 1$ y por lo tanto rechazada.

Usando f(8,4) para el borde superior, tenemos:

$$n_t * [P(t) - f] = -(2t - 3) = 0 \quad \text{--> } t = 3/2$$

que está tambien fuera del rango $0 <= t <= 1$ y por lo tanto rechazada. El intervalo visible de la línea P_1P_2 recortada por la región rectangular de la Figura 4.13 es $1/10 <= t <= 9/10$ o (0,1.2)-(8,2.8).

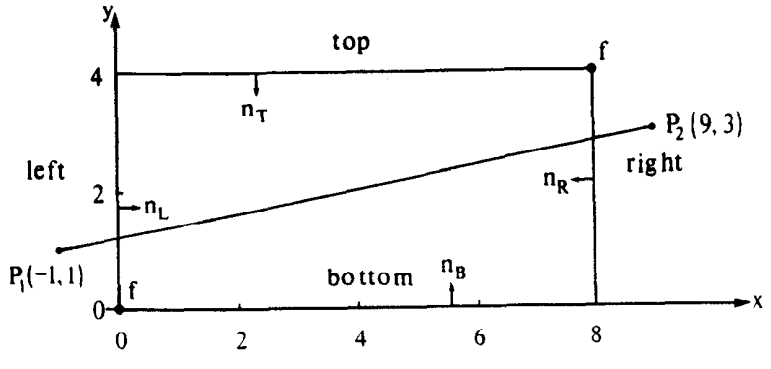


Figura 4.13

Este ejemplo muestra que los puntos de intersección pueden encontrarse fácilmente. La identificación de las líneas totalmente visibles y totalmente invisibles es ilustrada por los siguientes ejemplos.

Ejemplo: Cyrus-Beck. Líneas Totalmente Visibles.

Consideremos la línea $P_1(1,1)$ y $P_2(7,3)$ recortada por la ventana rectangular mostrada en la Figura 4.14. La representación paramétrica de la línea P_1P_2 es:

$$P(t) = [1\ \ 1] + [6,2]t$$

Los resultados, usando los vectores normales de entrada y los puntos límites del

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

ejemplo anterior se dan en la tabla 4.4.

Edge	n	f	P(t)−f	n·[P(t)−f]	t
Left	i	(0, 0)	(1 + 6t) i + (1 + 2t) j	1 + 6t	−1/6
Right	− i	(8, 4)	(−7 + 6t) i + (−3 + 2t) j	7 − 6t	7/6
Bottom	j	(0, 0)	(1 + 6t) i + (1 + 2t) j	1 + 2t	−1/2
Top	− j	(8, 4)	(−7 + 6t) i + (−3 + 2t) j	3 − 2t	3/2

Tabla 4.4

Todos los valores de intersección para t están fuera del rango 0< =t< =1. La línea entera es visible.

Los próximos dos ejemplos consideran dos tipos de líneas invisibles. Una línea está totalmente a la izquierda de la ventana y puede ser declarada invisible usando los códigos de los puntos extremos visto anteriormente. La segunda cruza la esquina de la ventana por fuera. No puede ser declarada invisible usando los códigos de los puntos extremos.

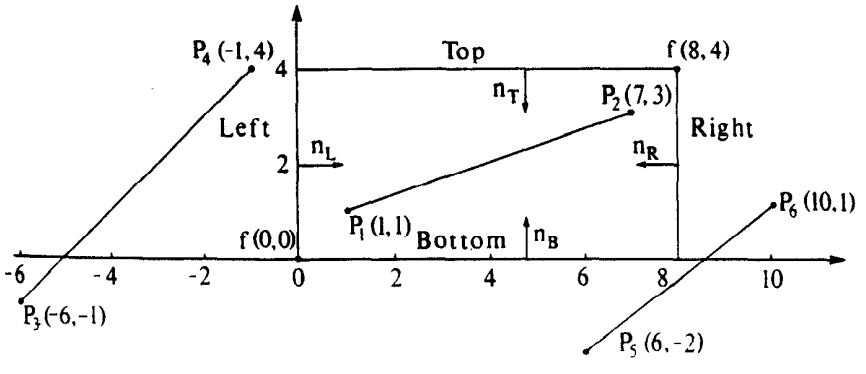


Figura 4.14

Ejemplo: Cyrus-Beck. Línea Trivialmente Invisible.

Consideremos la línea $P_3(-6,-1)$ y $P_4(-1,4)$ recortada por la ventana rectangular mostrada en la Figura 4.14. La línea es invisible. La representación paramétrica es:

$$P(t) = [-6\ -1] + [5\ 5]t$$

Los resultados usando los vectores normales y los puntos límites de los ejemplo anteriores, se dan en la Tabla 4.5.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Edge	n	f	P(t)−f	n·[P(t)−f]	t
Left	i	(0, 0)	$(-6+5t)\mathbf{i}+(-1+5t)\mathbf{j}$	$-6+5t$	$t = 6/5$
Right	−i	(8, 4)	$(-14+5t)\mathbf{i}+(-5+5t)\mathbf{j}$	$-(-14+5t)$	$t = 14/5$
Bottom	j	(0, 0)	$(-6+5t)\mathbf{i}+(-1+5t)\mathbf{j}$	$-1+5t$	$t = 1/5$
Top	−j	(8, 4)	$(-14+5t)\mathbf{i}+(-5+5t)\mathbf{j}$	$-(-5+5t)$	$t = 1$

Tabla 4.5

Examinando los resultados en la Tabla 4.5 vemos que los valores de intersección para los bordes izquierdo y derecho están ambos fuera del rango $0 <= t <= 1$, al contrario que para el superior e inferior. Basándonos en esto podemos asumir inicialmente que la línea es visible en el intervalo $1/5 <= t <= 1$. Sin embargo, un examen más profundo muestra que la ventana está completamente a la derecha de la línea. Por lo tanto la línea es invisible.

Si en este ejemplo intercambiamos los puntos P_3 y P_4, entonces los resultados mostrarían que la ventana estaba completamente a la izquierda de la línea. La dirección de la línea es importante para concluir sobre la invisibilidad de la línea. El próximo ejemplo aclarará esta cuestión.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Ejemplo. Cyrus-Beck. Línea Invisible no Trivial.

Sea la línea $P_5(6,-2)$ y $P_6(10,1)$ de nuevo recortada en la ventana rectangular de la Figura 4.14. La representación paramétrica es:

$$P(t) = [6\ -2] + [4\ 3]t$$

Usando los vectores normales de entrada y los puntos límites de los ejemplos anteriores se dan los resultados mostrados en la Tabla 4.6.

Edge	n	f	P(t)−f	n·[P(t)−f]	t
Left	i	(0, 0)	$(6+4t)\mathbf{i}+(-2+3t)\mathbf{j}$	$6+4t$	$t = -3/2$
Right	−i	(8, 4)	$(-2+4t)\mathbf{i}+(-6+3t)\mathbf{j}$	$-(-2+4t)$	$t = 1/2$
Bottom	j	(0, 0)	$(6+4t)\mathbf{i}+(-2+3t)\mathbf{j}$	$-2+3t$	$t = 2/3$
Top	−j	(8, 4)	$(-2+4t)\mathbf{i}+(-6+3t)\mathbf{j}$	$-(-6+3t)$	$t = 2$

Tabla 4.6

Los resultados muestran que para los bordes izquierdo y superior los valores de la intersección están fuera del rango. Mientras que para los bordes derecho e inferior están dentro. Pero, considerando la dirección de la línea P_5 a P_6, no es posible que ésta intersecte el borde derecho en $t = 1/2$, antes intersectaría el borde inferior en $t = 2/3$ y atravesaría la ventana. Luego la línea es invisible.

De estos ejemplos parece claro que la línea visible aparentemente puede ser identificada de manera correcta considerando también su dirección.

4.2 TRANSFORMACIONES GEOMETRICAS EN 2D

4.2.1 Traslación

Los puntos en el plano X-Y pueden trasladarse a nuevos puntos añadiendo la cantidad a desplazar a las coordenadas de los puntos. Para cada uno de los puntos P(X,Y) que se están trasladando Dx unidades en el eje X y Dy unidades en el eje Y al nuevo punto P'(X',Y'), podemos escribir:

$$X' = X + Dx \qquad Y' = Y + Dy \tag{4.3}$$

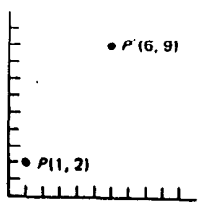


Figura 4.15

Esto se ilustra en la figura 4.15, en la cual el punto (1,2) se traslada (5,7) convirtiendose en el punto (6,9). Definiendo los siguientes vectores filas como

$$P = [X\ Y] \quad P' = [X'\ Y'] \quad T = [Dx\ Dy]$$

podemos volver a escribir la ecuación (4.1) en forma de vector como

$$[X'\ Y'] = [X\ Y] + [Dx\ Dy] \qquad \ldots \tag{4.4}$$

y más concisamente como

$$P' = P + T \tag{4.5}$$

Un objeto puede transladarse aplicando (4.3) a cada punto del objeto. Y_a que cada línea de un objeto está compuesta de un número infinito de puntos, este proceso se haría infinitamente largo. Afortunadamente, todos los puntos de una línea pueden trasladarse desplazando solamente sus puntos finales y dibujando una nueva línea entre los puntos trasladados. Esto es cierto para escalar y rotar. La figura 4.16 muestra el efecto de trasladar el perfil de una casa con (3,-4).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

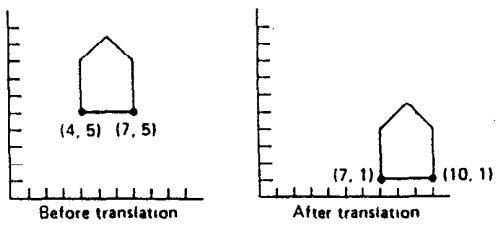


Figura 4.16

4.2.2 Escalado

Los puntos pueden ser escalados por Sx en el eje X y por Sy en el eje Y dentro de los nuevos puntos mediante la multiplicación:

$$X' = X * Sx \qquad Y' = Y * Sy \tag{4.6}$$

Definiendo S como:

$$S = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

podemos escribir en forma de matriz

$$[X' \ Y'] = [X \ Y] * \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \qquad (4.7)$$

ó

$$P' = P * S \tag{4.8}$$

En la figura 4.17 el punto (6,6) está escalado por 1/2 en X y 1/3 en Y. La figura 4.18 muestra el perfil de una casa escalado por 1/2 en X y 1/4 en Y. Observe que el escalado es con respecto al origen: la casa está más pequeña y más cercana al origen. Si el escalado fuese mayor que uno, la casa sería más grande y estaría más alejada del origen. Las proporciones de la casa también han cambiado: se ha utilizado un escalado diferencial, con Sx < > Sy. Con un escalado uniforme (Sx = Sy) no afecta a las proporciones de la casa.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

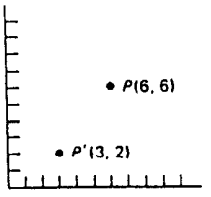


Figura 4.17

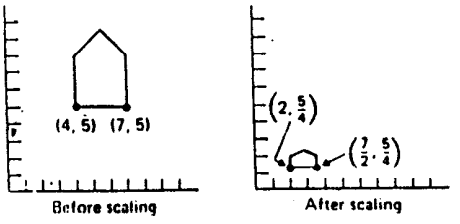


Figura 4.18

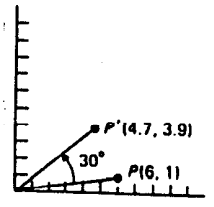


Figura 4.19

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

4.2.3 Rotación

Los puntos pueden rotarse sobre el origen a través de un ángulo θ, como muestra la figura 4.19 para el punto P(6,1) y un ángulo $\theta = 30°$. La rotación se define matemáticamente como:

$$X' = X \cos\theta - Y \operatorname{Sen}\theta$$
$$Y' = X \operatorname{Sen}\theta + Y \cos\theta \qquad (4.9)$$

En forma de matriz tenemos:

$$[X' \; Y'] = [X \; Y] \cdot \begin{bmatrix} \cos\theta & \operatorname{Sen}\theta \\ -\operatorname{Sen}\theta & \cos\theta \end{bmatrix} \qquad (4.10)$$

ó

$$P' = P * R \qquad (4.11)$$

donde R representa la matriz de rotación en (4.10). La figura 4.20 muestra un cuadrado rotado 45°. La rotación está realizada sobre el origen.

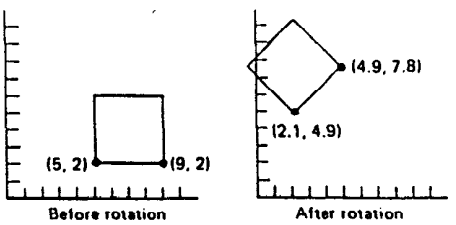


Figura 4.20

Los ángulos positivos se miden en sentido opuesto a las agujas del reloj desde X hacia Y. Para ángulos negativos (en el sentido de las agujas del reloj), pueden utilizarse $\text{Cos}(-\theta) = \text{Cos}\theta$ y $\text{Sen}(-\theta) = -\text{Sen}(\theta)$ para modificar (4.9) y (4.10).

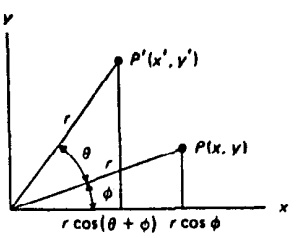


Figura 4.21

La ecuación (4.9) se deriva facilmente de la figura 4.21, en la cual una rotación con θ transforma P(X,Y) en P'(X',Y'). A medida que la rotación está sobre el origen, las distancias desde el origen a P y a P' son iguales y están etiquetadas "r" en la figura. Mediante trigonometría, tenemos que:

$$X = r\,\text{Cos}\theta \qquad Y = r\,\text{Sen}\theta \qquad (4.12)$$

y

$$X' = r\,\text{Cos}(\theta + \theta) = r\,\text{Cos}\theta\,\text{Cos}\theta - r\,\text{Sen}\theta\,\text{Sen}\theta$$
$$Y' = r\,\text{Sen}(\theta + \theta) = r\,\text{Cos}\theta\,\text{Sen}\theta + r\,\text{Sen}\theta\,\text{Cos}\,\theta \qquad (4.13)$$

Ahora sustituyendo (4.12) en (4.13) obtenemos (4.9).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

4.2.4 Afilamiento

Una transformación de afilamiento produce una distorción de un objeto, o bien, una imagen completa. Se examinan dos tipos de afilamiento: afilamiento **y** y afilamiento **x**. Un afilamiento y transforma el punto (x,y) al nuevo punto (x´,y´), donde:

$$x' = x$$

$$y' = Afy * x + y \quad Afy <> 0$$

Un afilamiento **y** desplaza una línea vertical hacia arriba o hacia abajo, dependiendo del signo del factor de afilamiento Afy.

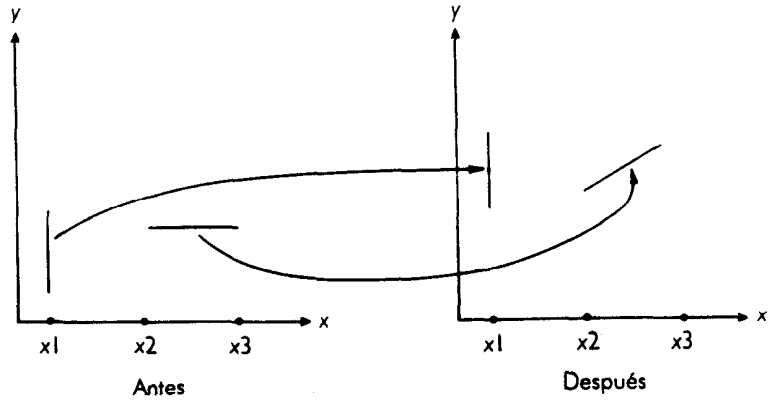


Figura 4.22

Una línea horizontal se distorciona hasta formar una línea inclinada con pendiente Afy. Combinando esas dos observaciones, se aprecia que el afilamiento **y** distorciona un rectángulo hasta darle la forma de un paralelogramo.

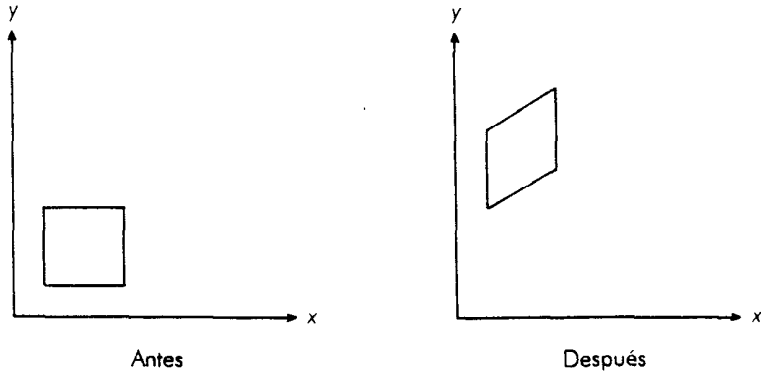


Figura 4.23

Un afilamiento **x** tiene el efecto opuesto. El punto (x,y) se transforma en el punto (x´,y´), donde:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$x' = x + Afx * y \quad Afx <> 0$$

$$y' = y$$

Las líneas verticales se vuelven inclinadas con pendiente Afx, en tanto que las horizontales se corren a la derecha o a la izquierda, dependiendo del signo de Afx.

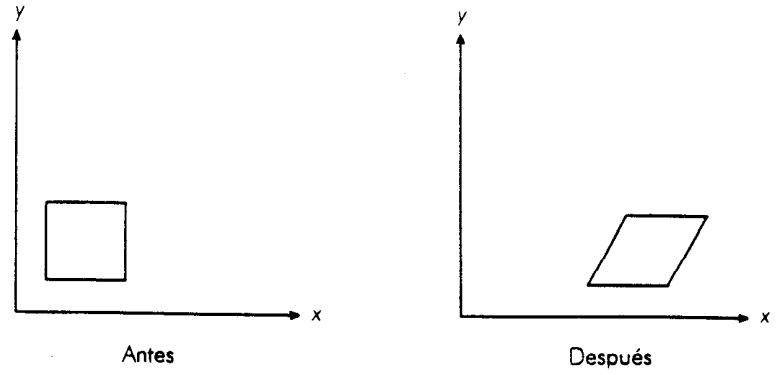


Figura 4.24

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

4.2.5 Representación en Coordenadas Homogéneas y Matrices de las Transformaciones de 2D

La representación de matrices para trasladar, escalar y rotar son, respectivamente:

$$P' = P + T \qquad (4.5)$$

$$P' = P * S \qquad (4.8)$$

$$P' = P * R \qquad (4.11)$$

Desafortunadamente, la traslación se trata de forma diferente (como una suma) que el escalado y la rotación (multiplicación). Sería interezante tratar las tres de una forma homogénea, para que puedan combinarse facilmente las tres transformaciones básicas, como se hará posteriormente.

Si expresáramos los puntos en Coordenadas Homogéneas, las tres transformaciones podrían tratarse como multiplicaciones. Las Coordenadas Homogéneas se desarrollan en geometría y han sido aplicadas en gráficos. Un gran número de paquetes de subrutinas gráficas trabajan con coordenadas y tranformaciones homogéneas. En algunos casos, el programa de aplicación los usa directamente pasando parámetros al paquete gráfico, mientras que en otro se aplican solamente dentro del paquete y no son visibles al programador.

En Coordenadas Homogéneas, el punto P(X,Y) se representa como P(W*X,W*Y,W) para cualquier escala de factores $W <> 0$. Entonces, dada una representación en coordenadas homogéneas para un punto P(X,Y,W), se puede encontrar la representación en coordenadas cartesianas de 2 dimensiones para el punto como $x = X/W$ e $y = Y/W$. En

nuestro caso, W será 1 y de este modo nunca será necesaria la división.

Ahora, los puntos son vectores fila de 3 elementos. De este modo, las matrices de transformación, que multiplican un punto del vector produciendo otro punto del vector, deben ser de 3X3. En forma de matriz de 3X3 para coordenadas homogéneas, las ecuaciones (4.3) están representadas como:

$$[X'\ Y'\ 1] = [X\ Y\ 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix} \qquad (4.14)$$

Expresado de otra forma:

$$P' = P * T(Dx,Dy) \qquad (4.15)$$

donde

$$T(Dx\ Dy) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix} \qquad (4.16)$$

¿Qué ocurre si un punto P se traslada con (Dx_1,Dy_1) a P' y a continuación se traslada con (Dx_2,Dy_2) a P''?. El resultado esperado intuitivamente es una traslación neta de (Dx_1+Dx_2,Dy_1+Dy_2). Vamos a ver cual sería el resultado:

$$P' = P * T(Dx_1,Dy_1) \qquad (4.17)$$

$$P'' = P' * T(Dx_2,Dy_2) \qquad (4.18)$$

Sustituyendo (4.17) en (4.18) obtenemos:

$$P'' = (P * T(Dx_1,Dy_1)) * T(Dx_2,Dy_2) = P * (T(Dx_1,Dy_1) * T(Dx_2,Dy_2)) \qquad (4.19)$$

El producto de matrices $T(Dx_1,Dy_1) * T(Dx_2,Dy_2)$ es:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_1 & Dy_1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_2 & Dy_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_1+Dx_2 & Dy_1+Dy_2 & 1 \end{bmatrix} \quad (4.20)$$

Efectivamente, la traslación es $(Dx_1 + Dx_2, Dy_1 + Dy_2)$.

De igual forma, las ecuaciones de escalado (4.6) se representan en forma de matriz como:

$$[X' \; Y' \; 1] = [X \; Y \; 1] * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.21)$$

Definiendo

$$S(Sx, Sy) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.22)$$

tenemos

$$P' = P * S(Sx,Sy) \quad (4.23)$$

De igual forma que las traslaciones sucesivas son aditivas, suponemos que escalados sucesivos serán multiplicativos. Dado:

$$P' = P * S(Sx_1,Sy_1) \quad (4.24)$$

$$P'' = P' * S(Sx_2,Sy_2) \quad (4.25)$$

sustituyendo (4.24) en (4.25) obtenemos:

$$P'' = (P * S(Sx_1,Sy_1)) * S(Sx_2,Sy_2) = P(S(Sx_1,Sy_1) * S(Sx_2,Sy_2)) \quad (4.26)$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

El producto de las matrices $S(Sx_1,Sy_1) * S(Sx_2,Sy_2)$ es:

$$\begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Sx_1 * Sx_2 & 0 & 0 \\ 0 & Sy_1 * Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.27)$$

Por lo tanto el escalado sucesivo es multiplicativo.

Finalmente, las ecuaciones de rotación (4.3) pueden representarse como:

$$[X' \; Y' \; 1] = [X \; Y \; 1] * \begin{bmatrix} Cos\theta & Sen\theta & 0 \\ -Sen\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.28)$$

Definiendo:

$$R(\theta) = \begin{bmatrix} Cos\theta & Sen\theta & 0 \\ -Sen\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.29)$$

tenemos:

$P' = P * R(\theta)$ (4.30)

4.2.6 Composición de Transformaciones de 2D

La idea de composición fue introducida en la sección anterior. Se va a mostrar como puede utilizarse la composición para combinar las matrices fundamentales R, S y T así como para producir resultados generales deseados. El propósito básico de la composición de transformaciones es que es más eficiente aplicar una sola transformación compuesta a un punto que aplicar una serie de transformaciones, una despues de otra.

Consideremos, por ejemplo, la rotación de un objeto sobre un punto arbitrario P_1. Y_a que solo conocemos como rotar sobre el origen, convertiremos nuestro problema original (dificil) en tres problemas diferentes (sencillos). De este modo, es necesaria una secuencia

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

de tres transformaciones para rotar sobre P :

1) Trasladar, de forma que P, esté en el origen.

2) Rotar.

3) Trasladar, de forma que el punto en el origen retorne a P_1.

Esta secuencia se muestra en la figura 4.25, en la cual el perfil de la casa se rot sobre $P_1(X_1,Y_1)$. La primera traslación es con $(-X_1,-Y_1)$, mientras que la última es con la inversa (X_1,Y_1). El resultado sería bastante diferente si se aplicara solamente la rotación.

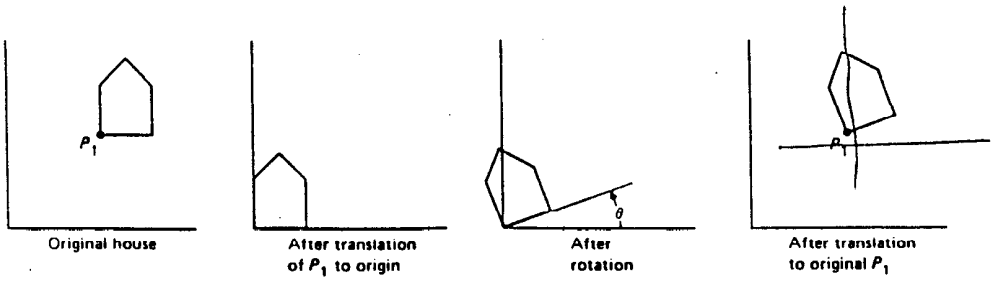


Figura 4.25

La transformación neta es:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_1 & -Y_1 & 1 \end{bmatrix} * \begin{bmatrix} Cos\theta & Sen\theta & 0 \\ -Sen\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X_1 & Y_1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} Cos\theta & Sen\theta & 0 \\ -Sen\theta & Cos\theta & 0 \\ X_1(1-Cos\theta)+Y_1Sen\theta & Y_1(1-Cos\theta)-X_1Sen\theta & 1 \end{bmatrix} \qquad (4.31)$$

Esta composición de transformaciones por multiplicación de matrices es un ejemplo de como las coordenadas homogéneas proporcionan sencillez.

De forma similar se utiliza para escalar un objeto sobre un punto arbitrario P_1: trasladar P_1 al origen, escalar y retornar a P_1. En este caso la transformación neta es:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_1 & -Y_1 & 1 \end{bmatrix} * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X_1 & Y_1 & 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ X_1(1-Sx) & Y_1(1-Sy) & 1 \end{bmatrix} \quad (4.32)$$

Supongamos que queremos escalar, rotar y posicionar la casa mostrada en la figura 4.25, siendo P_1 el centro para la rotación y el escalado. La secuencia sería trasladar P_1 al origen, realizar el escalado y la rotación, y entonces trasladar desde el origen a la nueva posición P_2. Una estructura de datos que apunta esta transformación podría contener la escala de factores, ángulo de rotación y cantidad de traslación, o simplemente la matriz de transformación compuesta:

$$T(-X_1,-Y_1) * S(Sx,Sy) * R(\theta) * T(X_2,Y_2)$$

Dado que M1 y M2 representan una traslación, escalado o rotación fundamental, cuando es M1*M2 = M2*M1, ¿cuando hace conmutativo M1 y M2?. En general, la multiplicación de matrices no es conmutativa. Sin embargo, es fácil mostrar que mantienen la conmutatividad los siguientes casos especiales:

M1	M2
Traslación	Traslación
Escalado	Escalado
Rotación	Rotación
Escalado (con Sx = Sy)	Rotación

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

4.2.7 Consideraciones de Eficiencia

La composición más general de las operaciones de R, S y T producirá una matriz de la forma:

$$M = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ tx & ty & 1 \end{bmatrix} \qquad (4.33)$$

La submatriz superior de $2X_2$ es una rotación compuesta y la matriz de escalado, mientras que tx y ty son traslaciones compuestas. Calculando P*M como un vector de tiempo, una matriz de 3X3 realiza nueve multiplicaciones y seis sumas. La estructura fijada en la última columna de (4.33) simplifica las operaciones actuales a:

$$X' = X*r_{11} + Y*r_{21} + tx'$$
$$Y' = X*r_{12} + Y*r_{22} + ty' \qquad (4.34)$$

reduciendo el proceso a cuatro multiplicaciones y cuatro sumas; un aumento significativo, especialmente porque la operación sería aplicada a cientos o miles de puntos por imagen.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

CAPITULO 5

GRAFICOS EN 3 DIMENSIONES

5.1 SISTEMAS DE COORDENADAS

Una imágen tridimensional suele definirse por los puntos, líneas y planos que la componen. Para especificar estos primitivos elementos de construcción, se necesita un sistema de coordenadas tridimensional, el cual puede concebirse como una extensión del sistema bidimensional. Así pues, la tercera dimensión, que es la profundidad, se representará mediante el eje Z, que se halla en ángulo recto respecto del plano de coordenadas X,Y. (Figura 5.1)

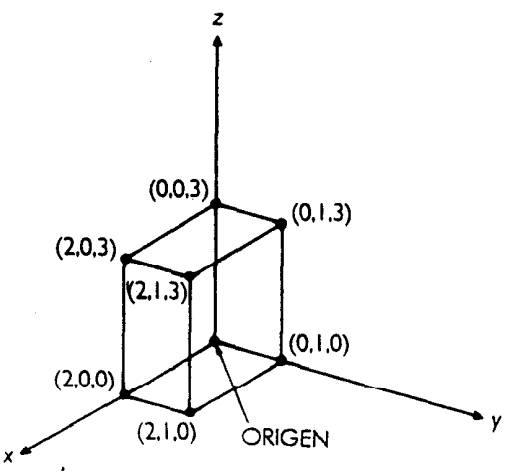


Figura 5.1

Cualquier punto en este sistema de coordenadas rectangular puede describirse por medio de una triada ordenada (x,y,z) de valores. Cada coordenada representa una distancia dirigida, es decir, una dirección positiva o negativa sobre su eje coordenado correspondiente.

El sistema de coordenadas ilustrado es de **Mano Derecha**. Si se cierran los dedos de la mano derecha en el sentido que va del eje positivo X al eje positivo Y, el pulgar apunta en la dirección del eje positivo Z.

Existe otro sistema de coordenadas tridimensional que se usa para presentar imágenes generadas por computador. La pantalla es un plano bidimensional con el origen en la esquina inferior izquierda, el eje Y positivo hacia arriba y el eje X positivo hacia la

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

derecha. El eje Z empieza en el origen y apunta al interior de la pantalla. Este sistema de coordenadas tiene la propiedad de que, cuando la pantalla es visualizada, los objetos que están en el fondo adoptan valores positivos Z más grandes (Figura 5.2). Se conoce a este sistema como de **Mano Izquierda**, ya que si se cierran los dedos de la mano izquierda en la trayectoría de X a Y, el pulgar apunta en la dirección Z. Tal sistema de coordenadas se puede definir de manera independiente de la pantalla de presentación, y se le conoce también como Sistema de Coordenadas del Observador, ya que el ojo del observador está en el origen.

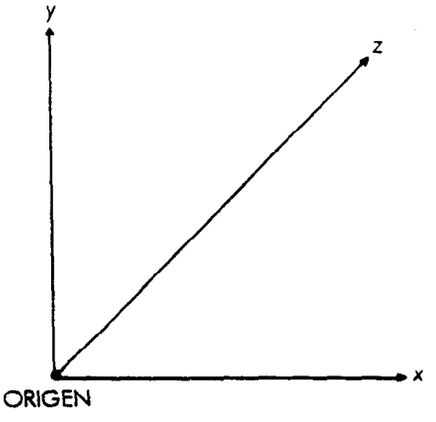


Figura 5.2

5.2 TRANSFORMACIONES

Las transformaciones en tres dimensiones no son más que extensiones de las transformaciones bidimensionales estudiadas en el capítulo anterior. De hecho, todas las transformaciones bidimensionales se pueden describir mediante el uso de transformaciones tridimensionales.

Recuérdese que todas las transformaciones de modelación geométrica bidimensional puede expresarse como matrices de 3X3. Para generalizar lo anterior a tres dimensiones, se asociará un punto tridimensional(x,y,z) con un vector homogéneo [x,y,z,1]. De este modo, pueden representarse todas las transformaciones lineales tridimensionales por medio de multiplicaciones de matrices de 4X4. Tales transformaciones pueden asimismo combinarse multiplicando sus matrices asociadas. Como en el caso bidimensional, el orden en el cual se multiplican las matrices afecta las transformaciones resultantes.

5.2.1 Traslación

La transformación que traslada un punto (x,y,z) T_x unidades en la dirección X, T_y unidades en la dirección Y y T_z unidades en la dirección Z hacia un punto nuevo (x´, y´, z´) se representa por medio de la matriz:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$T(T_x, T_y, T_z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

El punto homogéneo trasladado es:

$$[x', y', z', 1] = [x, y, z, 1] * T(T_x, T_y, T_z)$$

dode:

$$x' = x + T_x$$
$$y' = y + T_y$$
$$z' = z + T_z$$

La matriz de traslación inversa se obtiene invirtiendo el signo de los valores de T_x, T_y, T_z. Así, para trasladar el punto (x´, y´, z´) de regreso a (x, y, z), se multiplica el punto homogéneo [x´, y´,z´, 1] por la matriz $T(-T_x, -T_y, -T_z)$.

5.2.2 Escalado

El escalado tridimensional permite la contracción o la expansión en cualesquiera de las direcciones x, y o z. Como en el caso bidimensional, se describe el escalado cuando el punto fijo es el origen. Para obtener un escalado con un punto fijo arbitrario, éste se debe trasladar al origen, escalar el objeto, y después realizar el inverso de la traslación original. La matriz de escalado con factores de escala E_x, E_y, E_z en las direcciones x, y, z, respectivamente, está dada por la matriz:

$$E(E_x, E_y, E_z) = \begin{bmatrix} E_X & 0 & 0 & 0 \\ 0 & E_Y & 0 & 0 \\ 0 & 0 & E_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El punto homogéneo [x, y, z, 1] se transforma en el punto homogéneo [x´,y´,z´,1], donde:

$$x' = x * E_x$$
$$y' = y * E_y$$
$$z' = z * E_z$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

La inversa de un escalado se obtiene usando los recíprocos de los factores de escala: $1/E_x$, $1/E_y$, $1/E_z$.

5.2.3 Rotación

Una rotación tridimensional se efectúa alrededor de un eje. Por ahora, se considerarán sólo los casos en los que el eje de rotación es uno de los tres ejes de coordenadas: x, y o z. El ángulo de rotación será positivo si se realiza en un eje de rotación positivo desde el origen, y se lleva a cabo una rotación en sentido contrario de las manecillas del reloj (Figura 5.3).

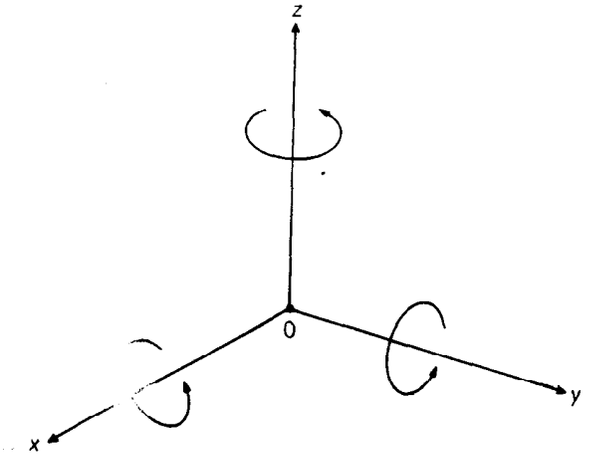


Figura 5.3

Una rotación bidimensional en torno al origen equivale a una rotación tridimensional en el plano X,Y y en torno al eje Z. En general, una rotación del punto alrededor del eje se realiza en un plano perpendicular a éste; esto significa que el punto permanece en este plano.

Una rotación de un ángulo θ sobre el eje Z se representa por medio de la matriz:

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & -sen\theta & 0 \\ 0 & 1 & 0 & 0 \\ sen\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Aplicando $R_y(\theta)$ a un punto homogéneo [x, y, z, 1], se obtiene el punto trnasformado:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$R_z(\theta) = \begin{bmatrix} \cos\theta & sen\theta & 0 & 0 \\ -sen\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x', y', z', 1] = [x, y, z, 1] * R_z(\theta)$$

donde:

$$x' = x\cos(\theta) - y\,sen(\theta)$$
$$y' = x\,sen(\theta) + y\cos(\theta)$$
$$z' = z$$

Nótese que el valor de la coordenada Z no cambia. Esto se debe a que la rotación alrededor del eje Z se realiza en un plano paralelo al plano x,y.

La rotación de un punto alrededor del eje Y se realiza en un plano paralelo al plano X,Z. El valor de la coordenada Y no cambia. La matriz de rotación es:

El punto homogéneo [x, y, z, 1] se convierte en el punto homogéneo [x',y',z',1], donde:

$$x' = x\cos(\theta) + z\,sen(\theta)$$
$$y' = y$$
$$z' = -x\,sen(\theta) + z\cos(\theta)$$

La rotación de un punto alrededor del eje X sucede en un plano paralelo al plano Y,Z. El valor de la coordenada X no cambia. Su matriz de rotación es:

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & sen\theta & 0 \\ 0 & -sen\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El punto homogéneo [x, y, z, 1] se convierte en el punto homogéneo [x',y',z',1] donde:

$$x' = x$$
$$y' = y\cos(\theta) - z\,sen(\theta)$$
$$z' = y\,sen(\theta) + z\cos(\theta)$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

La derivación de estas tres matrices es similar a la matriz de rotación bidimensional estudiada en el capítulo anterior. Basta con usar los planos de coordenadas correspondientes para realizar los cálculos. A fin de poder visualizar la geometría tridimensional, el eje de rotación deberá ser perpendicular al papel (Figura 5.4).

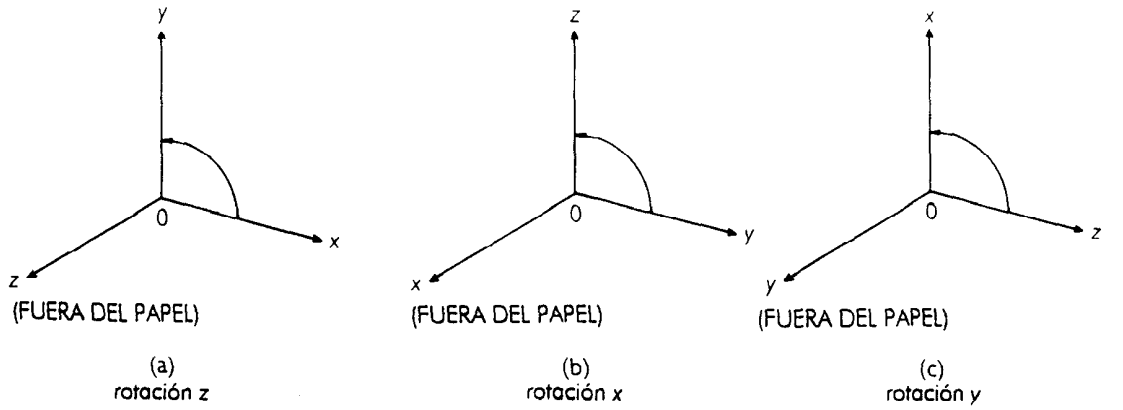


Figura 5.4

La inversa de una rotación de θ grados es simplemente una rotación de $-\theta$ grados (θ grados en la dirección de las manecillas del reloj). Las matrices inversas se obtienen reemplazando θ por $-\theta$ en la matriz correspondiente. Como $\cos(-\theta) = \cos(\theta)$ y $\text{sen}(-\theta) = -\text{sen}(\theta)$, es necesario cambiar los signos sólo en los términos seno.

5.2.4 Afilamiento

Supóngase que se quiere transformar la línea $y = -bz$ en el plano y,z al eje z, de modo que todos los valores de z permanezcan fijos (Figura 5.5a)

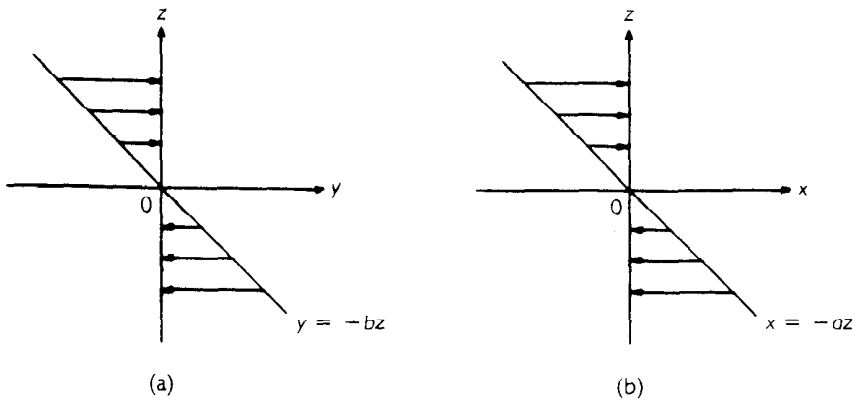


Figura 5.5

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Como el eje z tiene un valor y igual a cero, se debe sustraer el valor y de cada punto sobre la línea, -bz, de cada coordenada y. Por lo tanto, esta transformación envía (y,z) a (y´,z´), donde:

$$y' = y - (-bz) = y + bz$$
$$z' = z$$

Del mismo modo, se transforma la línea x=-az en el plano xz al eje z (figura 5.5b) por medio de las ecuaciones:

$$x' = x - (-az) = x + az$$
$$z' = z$$

La transformación tridimensional combinada es un afilamiento en z que toma una línea arbitraria, no ubicada en el plano x,y, que pasa por el origen, y la mueve al eje z de modo que los valores de z permanecen fijos. Su matriz de representación está dada por:

$$Af(a,b) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Los valores de a y b de la tercera fila se obtienen de la ecuación de la recta. Para calcular estos valores, sea $P:(x_1,y_1,z_1) <> 0$ para cualquier punto sobre la recta que pasa por el origen:

$$x = -a * z$$
$$y = -b * z$$

Al sustituir el punto p en estas ecuaciones, se obtiene:

$$a = -x_1/z_1 \quad y \quad b = -y_1/z_1$$

Al partir de este resultado se entiende por qué la recta original no puede estar en el plano X,Y: pues ello equivaldría a hacer una división entre $z_1 = 0$.

5.2.5 Composición de Transformaciones de 3D

Las transformaciones básicas de 3D pueden componerse para obtener muchos resultados diferentes. Aquí veremos un ejemplo de ello. La tarea consiste en transformar las líneas P_1P_2 y P_1P_3 de la figura 5.6 desde la posición inicial a la final. El punto P_1 se ha

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

trasladado al origen, P_1P_2 se situa en el eje Z negativo y P_1P_3 se situa en el plano YZ. las longitudes de las líneas no son afectadas por la transformación.

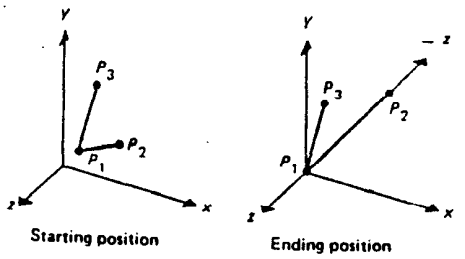


Figura 5.6

De nuevo, dividimos el problema original en problemas más simples. En este caso, la transformación puede realizarse en cuatro pasos:

1. Trasladar P_1 al origen.
2. Rotar sobre el eje Y, así P_1P_2 queda situado en el plano YZ.
3. Rotar sobre el eje X, así P_1P_2 queda situado en el eje Z negativo.
4. Rotar sobre el eje Z, así P_1P_3 queda situado en el plano YZ.

Paso 1: Trasladar P_1 al origen.

$$T(-X_1, -Y_1, -Z_1] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_1 & -Y_1 & -Z_1 & 1 \end{bmatrix}$$

Aplicando T a P_1, P_2 y P_3 queda:

$$P'_1 = P_1 * T(-X_1,-Y_1,-Z_1) = [0\ 0\ 0\ 1]$$

$$P'_2 = P_2 * T(-X_1,-Y_1,-Z_1) = [X_2-X_1\ Y_2-Y_1\ Z_2-Z_1\ 1]$$

$$P'_3 = P_3 * T(-X_1-Y_1,-Z_1) = [X_3-X_1\ Y_3-Y_1\ Z_3-Z_1\ 1]$$

Paso 2: Rotar sobre el eje Y. La figura 5.7 muestra P_1P_2 después del paso 1, siempre con la proyeción de P_1P_2 en el plano XZ. La rotación es mediante el ángulo positivo θ, por el cual

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$Cos\theta = -Z_2'/D_1 = -(Z_2-Z_1)/D_1$$

$$Sen\theta = X_2'/D_1 = X_2-X_1/D_1$$

donde

$$D_1 = \sqrt{(Z_2-Z_1)^2+(X_2-X_1)^2}$$

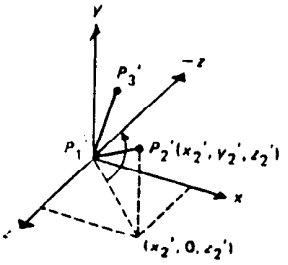


Figura 5.7

Sustituyendo estos valores tenemos:

$$P_2'' = P_2' * R_y(\theta) = [0 \quad Y_2-Y_1 \quad (-(X_2-X_1)^2/D_1)-(Z_2-Z_1)^2/D_1 \quad 1]$$

como lo esperado, la componente X de P_2'' es cero.

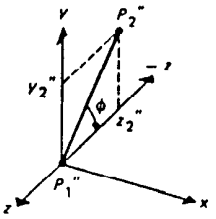


Figura 5.8

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Paso 3: Rotar sobre el eje X. La figura 5.8 muestra P_1P_2 después del paso 2. La rotación es mediante el ángulo negativo θ, por el cual

$$\text{Cos}(-\theta) = \text{Cos}\theta = -Z_2'' / | |P_1P_2| |'$$

$$\text{Sen}(-\theta) = -\text{Sen}\theta = -Y_2'' / | |P_1P_2| |'$$

donde

$$| |P_1P_2| | = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2+(Z_2-Z_1)^2}$$

La notación $| |P_1P_2| |$ significa "La longitud de P_1P_2". El resultado de la rotación en el paso 3 es:

$$P_2''' = P_2'' * R_x(\theta) = P_2' * R_y(\theta) * R_x(\theta) =$$

$$= P_2 * T * R_y(\theta) * R_x(\theta) = [0 \;\; 0 \;\; -| |P_1P_2| | \;\; 1]$$

P_1P_2 coincide ahora con el eje Z negativo.

Paso 4: Rotación sobre el eje Z. La figura 5.9 muestra P_1P_2 y P_1P_3 despues del paso 3, con P_2''' en el eje Z negativo y P_3''' en la posición

$$P_3''' = [X_3''' \;\; Y_3''' \;\; Z_3''' \;\; 1] = P_3 * T(-X_1,-Y_1,-Z_1) * Ry(\theta) * Rx(\theta)$$

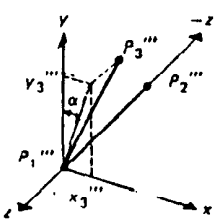


Figura 5.9

La rotación se realiza a través del ángulo α positivo, con:

$$\text{Cos}\,\alpha = Y_3''' / D_2$$

$$\text{Sen}\,\alpha = X_3''' / D_2$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$D_2 = \sqrt{(X_3''')^2 + (Y_3''')^2}$$

El paso 4 es el último paso para alcanzar el resultado final representado en la figura 5.6.

La matriz compuesta

$$T(-X_1,-Y_1,-Z_1) * R_y(\theta) * R_x(\theta) * R_z(\theta) = T * R$$

es la transformación necesitada, con

$$R = R_y(\theta) * R_x(\theta) * R_z(\theta)$$

Aplicando esta transformación a P_1, P_2 y P_3 verifica que P_1 se coloca en el origen, P_2 en el eje Z negativo y P_3 en el eje Y positivo del plano YZ.

5.2.6 Rotación en torno a un eje Arbitrario

Se han descrito las transformaciones alrededor de cualquiera de los tres ejes de coordenadas. Sin embargo, existen situaciones en las que el eje de rotación no es ninguno de éstos. La determinación de esta clase de rotación arbitraria es similar, aunque más compleja que la determinación de la rotación bidimensional alrededor de un punto cualquiera. Antes de estudiar esta transformación, se hará una representación del eje de rotación, que es una línea recta.

Dos puntos, $P_1:(x_1,y_1,z_1)$ y $P_2:(x_2,y_2,z_2)$, definen una recta (Figura 5.10). Las ecuaciones paramétricas de la recta que pasa por estos puntos son:

$$x = (x_2-x_1)t + x_1$$
$$y = (y_2-y_1)t + y_1$$
$$z = (z_2-z_1)t + z_1$$

donde t supone valores reales. Si $t=0$, se alcanza el punto P_1, mientras que $t=1$ proporciona el punto P_2. El segmento de recta que va de P_1 a P_2 tiene valores de t restringidos al intervalo entre 0 y 1.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

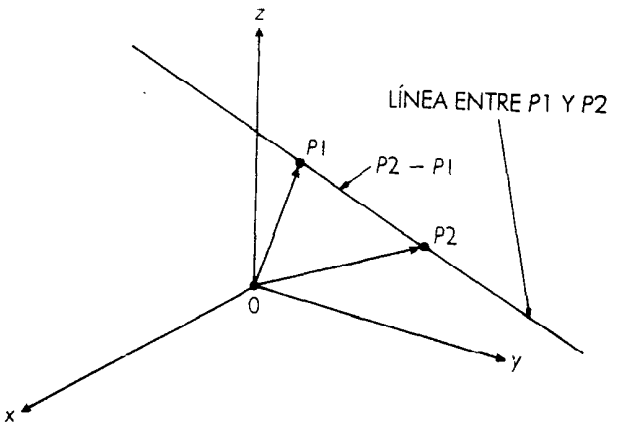


Figura 5.10

Las tres ecuaciones anteriores pueden concebirse en una forma algo diferente, lo que conviene para nuestros objetivos. Sea

$$a = (x_2 - x_1)$$
$$b = (y_2 - y_1)$$
$$c = (z_2 - z_1)$$

Ahora las ecuaciones de la recta tienen la forma

$$x = at + x_1$$
$$y = bt + y_1$$
$$z = ct + z_1$$

Obsérvese que P_1 y P_2 pueden considerarse como vectores, y que la diferencia

$$P_2 - P_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1) = (a,b,c)$$

es el vector (dirección) de P_1 a P_2 sobre la recta que describe la misma trayectoria (Figura 5.10). Por lo tanto, son tres vectores (a,b,.c) los que describen la **dirección** de la recta que transcurre del punto (x_1,y_1,z_1) al punto (x_2,y_2,z_2). De aquí que una recta pueda definirse por medio de un punto superpuesto, (x_1,y_1,z_1), y por una dirección (a,b,c).

Volviendo con el eje de rotación, sea (x_1,y_1,z_1) un punto sobre el cual pasa el eje de rotación y cuyo vector de dirección es (a,b,c). La rotación de un ángulo θ alrededor de este eje puede describirse por medio de los siete pasos siguientes:

1. Trasladar el punto (x1,y1,z1) al origen.

2. Rotar en torno al eje X hasta que el eje de rotación alcance el plano X-Z.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

3. Rotar sobre el eje Y hasta que el eje de rotación corresponda al eje Z.

4. Rotar sobre el eje Z un ángulo θ.

5. Realizar la rotación inversa del paso 3.

6. Realizar la rotación inversa del paso 2.

7. Realizar la traslación inversa del paso 1.

Estos pasos transforman el eje de rotación en el eje Z y realizan la rotación del ángulo Θ, para después transformar el eje de rotación en su posición original. Pongamos ahora en práctica los siete pasos.

La traslación inicial se representa por medio de la matriz:

$$T_r(-x_1, -y_1, -z_1] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix}$$

Después de esta traslación, el vector de dirección (a,b,c) que define el eje de rotación está en la posición ilustrada en la figura 5.11a.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

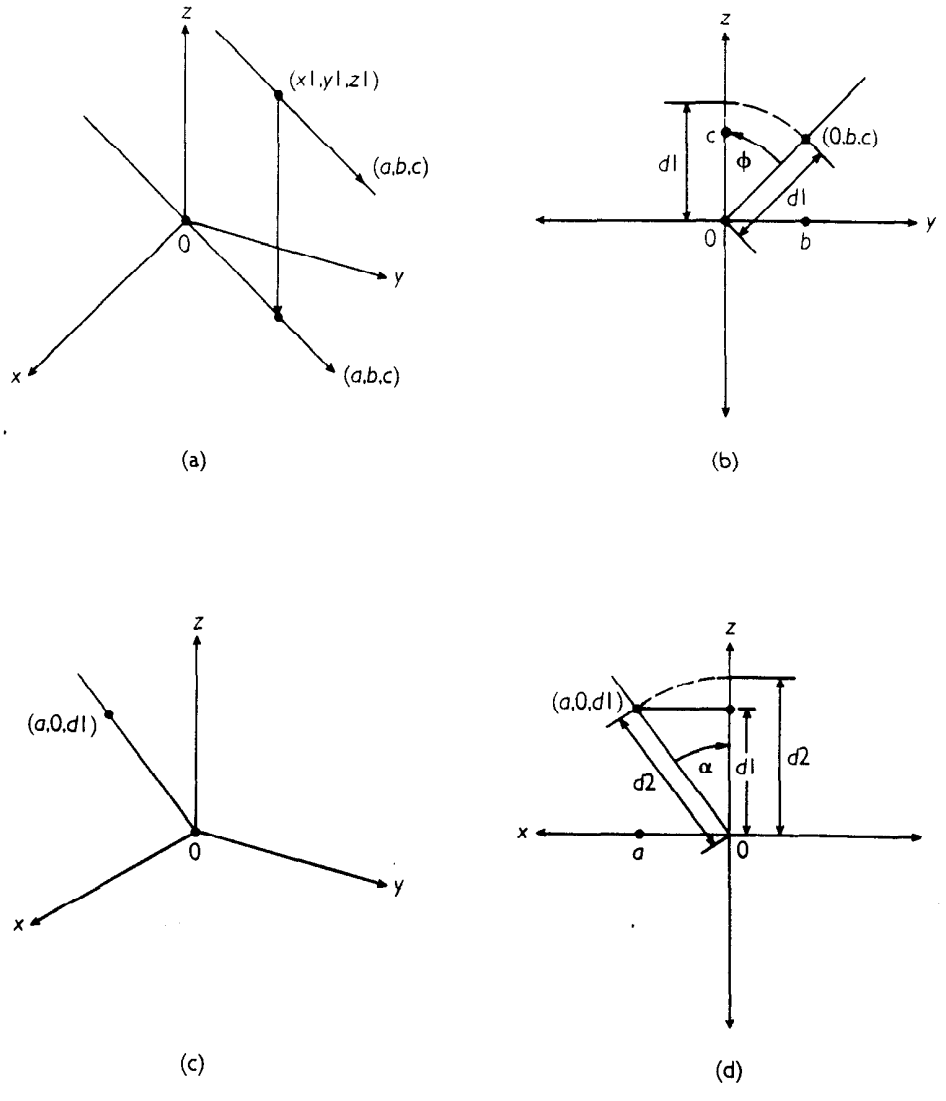


Figura 5.11

Como el paso 2 rota esta recta alrededor del eje X, se puede usar su proyección en el plano Y-Z y considerar esto como una rotación en torno al origen, con el eje X saliéndose del papel (Figura 5.11b). Cuando el eje de rotación se proyecta sobre el plano Y-Z, cualquier punto sobre éste tiene la coordenada x igual a cero. En particular, a = 0.

Esta recta proyectada o, lo que es lo mismo, el punto (0,b,c), se rota ϕ grados hasta que la recta corresponda al eje Z. En realidad no es necesario encontrar el ángulo ϕ sino el seno y el coseno del ángulo, ya que estas funciones trigonométricas definen la matriz de rotación. La figura 5.11 muestra que la distancia entre el origen y (0,b,c) es

$$\sqrt{b^2+c^2}=d_1$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

De aquí,

$$\text{sen } \phi = b/d1$$
$$\cos \phi = c/d1$$

Al sustituir esto valores en la matriz de rotación sobre el eje X, se tiene

$$R_X(\phi] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d_1 & b/d_1 & 0 \\ 0 & -b/d_1 & c/d_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Después de este paso, el eje de rotación se halla en el plano X-Z, y el punto (0,b,c) ha sido transformado en el punto $(0,0,d_1)$. Dado que la rotación en torno al eje X no cambia los valores de las coordenadas X, el punto (a,b,c) está ahora en la posición $(a,0,d_1)$ (Figura 5.11c)

El siguiente paso es una rotación alrededor del eje Y hasta que el punto $(a,0,d_1)$ esté en el eje Z. Como $(a,0,d_1)$ está en el plano X-Z, se puede visualizar esto como una rotación alrededor del origen, con el eje Y rebasando el papel (Figura 5.11d). La rotación de un ángulo α en el sentido de las manecillas del reloj se encarga de esta tarea. Dado que la dirección antihorario es positiva, se debe hacer una rotación de $-\alpha$. Al igual que antes, sólo se necesita calcular el seno y el coseno de α. A partir de la figura 5.11d, se sabe que la hipotenusa es

$$d_2 = \sqrt{a^2+d_1^2} = \sqrt{a^2+b^2+c^2}$$

Así

$$\text{sen } \alpha = a/d_2$$
$$\cos \alpha = d_1/d_2$$

y

$$\text{sen}(-\alpha) = -a/d_2$$
$$\cos(-\alpha) = d_1/d_2$$

Al sustituir estos valores en la matriz de rotación y, se obtiene:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$R_Y(-\alpha] = \begin{bmatrix} d_1/d_2 & 0 & a/d_2 & 0 \\ 0 & 1 & 0 & 0 \\ -a/d_2 & 0 & d_1/d_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El paso 4 requiere sólo la matriz de rotación $R_z(\Theta)$. Los últimos tres pasos emplean las matrices inversas de las primeras tres transformaciones. Las inversas de las matrices de rotación $R_x(\phi)$ y $R_y(-\alpha)$ son $R_x(-\phi)$ y $R_y(\alpha)$, respectivamente. La inversa de la matriz de traslación $T_r(-x_1,-y_1,-z_1)$ es $T_r(x_1,y_1,z_1)$

La trasnformación compuesta que representa los siete pasos está expresada en el producto

$$T_r(-x_1,-y_1,-z_1) * R_x(\phi) * R_y(-\alpha) * R_z(\Theta) * R_y(\alpha) * R_x(-\phi) * T_r(x_1,y_1,z_1)$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

5.3 TRANSFORMACION DE SISTEMAS DE COORDENADAS

Puede ocurrir que los objetos definidos en un sistema de coordenadas se deban expresar en otro sistema de coordenadas.

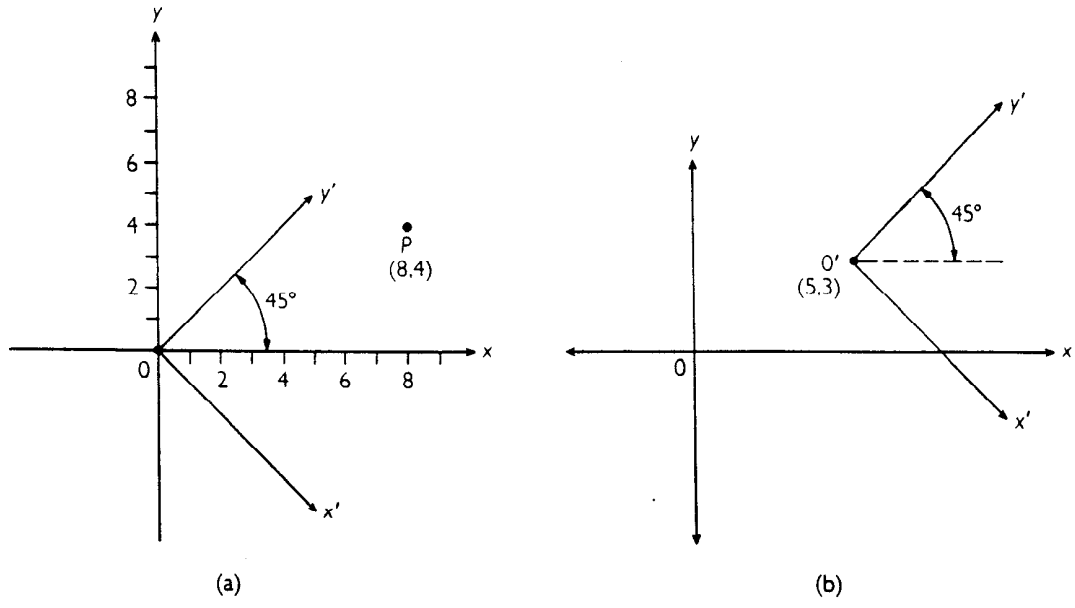
En general, si un objeto está definido según el sistema de coordenadas 1, puede definirse de acuerdo con otro sistema de coordenadas 2 transformando los ejes del sistema de coordenadas 2 en los ejes del sistema de coordenadas 1. Cuando se aplica tal transformación a la representación de un objeto que se encuentra en el sistema de coordenadas 1, se obtiene la representación de ese objeto en el sistema de coordenadas 2.

Examinemos ahora la situación de la figura 5.12a, donde el segundo sistema de coordenadas, denotado por x´,y´, se halla en un ángulo de 45 grados horario del sistema de coordenadas x,y. De acuerdo con esta regla, la transformación bidimensional requerida que remite el segundo sistema de coordenadas, x´,y´, al primer sistema de coordenadas, x,y, es una rotación de 45 grados en sentido antihorario.

$$Rotar(\pi/4) = \begin{bmatrix} \cos\pi/4 & sen\pi/4 & 0 \\ -sen\pi/4 & \cos\pi/4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

o

$$Rotar(\pi/4) = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Figura 5.12

Por tanto, el punto P=(8,4) del sistema de coordenadas x,y se representan como:

$$P' = (2\sqrt{2}, 6\sqrt{2})$$

en el sistema de coordenadas x´,y´, donde

$$[2\sqrt{2}, 6\sqrt{2}, 1] = [8, 4, 1] * Rotar(\pi/4)$$

Si el sistema de coordenadas rotado x´,y´ tiene su origen en (5,3) con respecto al sistema de coordenadas x,y (Figura 5.12b), la transformación de coordenadas se obtiene

premultiplicando la matriz de rotación anterior por la matriz de traslación bidimensional

$$Trasladar(-5,-3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -5 & -3 & 1 \end{bmatrix}$$

Esta técnica funciona correctamente siempre que el ángulo de rotación esté dado o se calcule con facilidad. Sin embargo, en la mayoría de los casos los ángulos son desconocidos y su cálculo requiere un esfuerzo considerable. En particular, se examinarán situaciones en las cuales el nuevo sistema de coordenadas está definido por su origen y las direcciones de sus tres ejes perpendiculares con respecto al sistema de coordenadas estándar. Como se dijo anteriormente, estas direcciones son vectores que recorren los tres ejes.

Un vector de dirección bidimensional, (a,b), que define un eje tiene cierta longitud

$$l = \sqrt{a^2+b^2}$$

Si cada componente del vector se divide entre la longitud, l, el vector resultante, u, tiene una longitud unitaria. Este proceso se conoce como normalización del vector, y el vector resultante tiene una norma (longitud) igual a 1:

u = (a/l,b/l);

$$longitud\ de\ u = \sqrt{(a/l)^2+(b/l)^2} = \frac{\sqrt{(a^2+b^2)}}{(a^2+b^2)} = 1$$

Regresando a la figura 5.12b, el sistema de coordenadas x´,y´ tiene el eje x´definido por el vector unitario de dirección

$$(1/\sqrt{2}, -1/\sqrt{2})$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2003

en tanto que la definición del eje y está definido por el vector unitario de dirección

$$(1/\sqrt{2}, 1/\sqrt{2})$$

Se forma, pues, la matriz de 3X3, cuya primera columna es el vector u, la segunda el vector v y la tercera el vector [0,0,1]

$$R = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Todo lo visto hasta ahora puede extenderse a tres dimensiones. Sea un sistema de coordenadas x´,y´,z´ definido en términos de las coordenadas estándares x,y,z por medio de los vectores unitarios perpendiculares

u = (ux, uy, uz)
v = (vx, vy, vz)
w = (wx, wy, wz)

Más aún, supóngase que el origen del sistema de coordenadas x´,y´,z´ está en (x_1,y_1,z_1) con respecto al sistema de coordenadas x,y,z (Figura 5.13). La transformación que envía el sistema x´,y´,z´ al sistema x,y,z, con u pasando a x, v a y y w a z, está dada por el producto de una traslación seguida de una rotación:

$$T_r * R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix} \begin{bmatrix} ux & vx & wx & 0 \\ uy & vy & wy & 0 \\ uz & vz & wz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

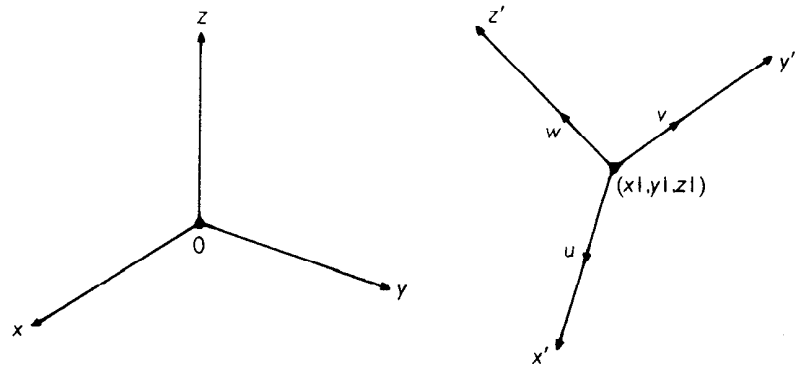


Figura 5.13

Es interesante observar que la matriz de rotación, R, es en realidad el producto de una matriz de rotación en x, y, z. Un punto P definido en el sistema de coordenadas x,y,z tiene la representación P´ en el sistema x´,y´,z´ dado por $P' = P*T_r*R$.

5,4 PROYECCIONES

En los gráficos por computador, un objeto tridimensional es presentado en una pantalla bidimensional. La Proyección es una transformación que convierte una representación tridimensional en bidimensional. Sin embargo, este proceso no es privativo del campo de los gráficos por computador. Los artistas usan una proyección en perpectiva para crear imágenes realistas; los arquitectos se valen de varias proyecciones paralelas para describir diferentes prespectivas de una construcción; etc.

5.4.1 Proyección Paralela

El método más sencillo de proyección de imágenes tridimensionales en dos dimensiones consiste en descartar una de las coordenadas. Se conoce a este método como **proyección paralela u ortagonal.** Aquí se considera sólo el caso de la eliminación de la coordenada z, lo cual proyecta el sistema de coordenadas x,y,z sobre el plano x,y (Figura 5.14).

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

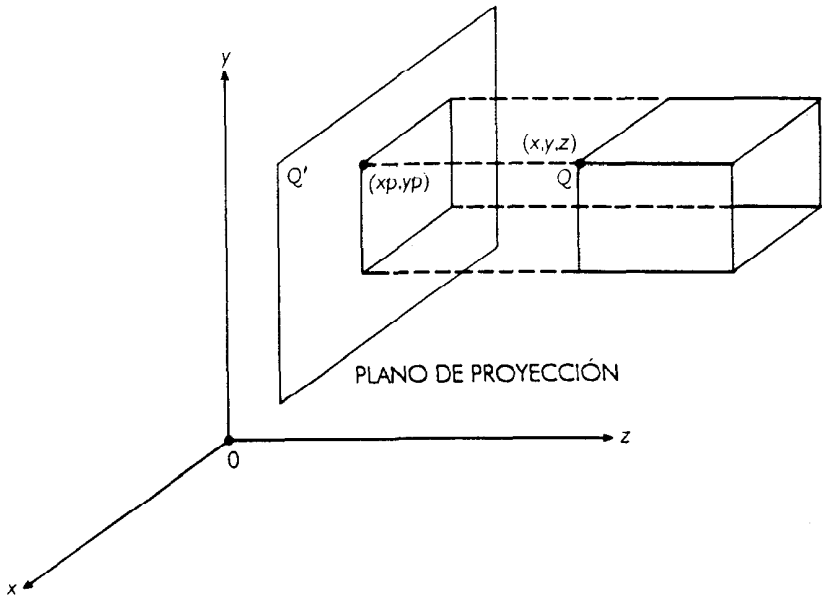


Figura 5.14

La proyección de un punto Q:(x,y,z) dentro del cubo es el punto Q´:(p_x,p_y) en el plano x,y, donde una línea que pasa por Q y que es paralela al eje z intersecta el plano x,y. Estas líneas paralelas se conocen como Proyectores. La matemática que describe lo anterior es muy simple:

$$p_x = x$$
$$p_y = y$$

Las proyecciones paralelas tienen la propiedad de que las líneas rectas permanecen como tales. Por ello, sólo se necesita proyectar los extremos de una recta en tres dimensiones, para luego dibujar una línea bidimensional entre estos puntos. Esto mejora considerablemente la velocidad del proceso de transformación.

La principal desventaja de la proyección paralela es la carencia de información en torno a la profundidad. El observador de la proyección de un cubo (especialmente de un cubo cuadrado) no tiene idea de que el objeto original en tres dimensiones es el cubo. Las imágenes bidimensionales realistas deben proporcionar al observador una representación más exacta de la imagen tridimensional.

5.4.2 Proyección Perspectiva

La Poryección en Perspectiva aumenta el realismo de una imagen a través de la impresión de profundidad. Las partes de la imagen que están lejos de la mirada del observador se dibujan más pequeñas que las del frente. De aquí se obtiene un reflejo del mundo real, donde los objetos más cercanos al observador lucen más grandes que los que

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

se hallan atrás.

En la proyección perspectiva, las líneas de proyección, conocidas como proyectores, atraen al ojo hacia un punto sobre el objeto. El punto de intersección del proyector con el plano, llamado Plano de Poryección, es el punto proyectado. (figura 5.15). A diferencia de las proyecciones paralelas, todos los proyectores, por definición, convergen en el ojo.

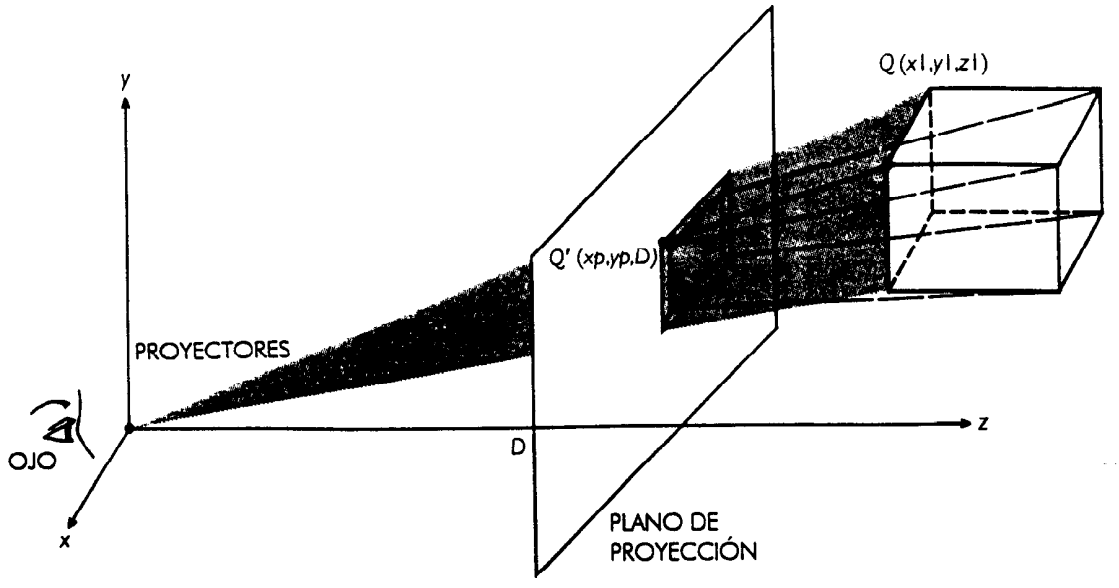


Figura 5.15

A fin de examinar las matemáticas de la proyección en perspectiva, supóngase que el plano de proyección es paralelo al plano x,y a una distancia D del ojo. El ojo se conoce como el centro de proyección. Nos valdremos de un sistema de coordenadas de mano izquierda en nuestras aplicaciones.

Sea $Q=(x_1,y_1,z_1)$ un punto que se proyecta al punto $Q'=(p_x,p_y,D)$, como se muestra en la Figura 5.15. Existen dos métodos, analizados a continuación, que se emplean para obtener las coordenadas de proyección p_x,p_y.

En la primera derivación, la línea que une el centro de proyección (0,0,0) y el punto Q tiene las ecuaciones paramétricas:

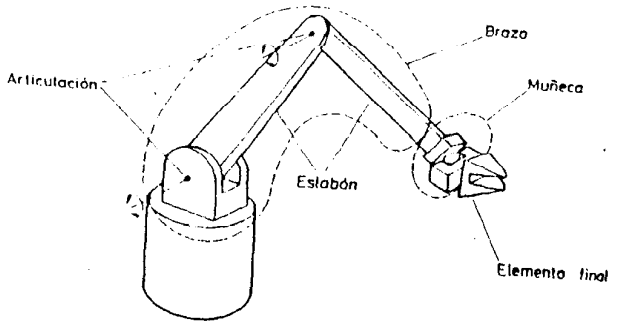
$$x = x_1 * t$$
$$y = y_1 * t \qquad \text{donde t es un número real}$$
$$z = z_1 * t$$

El punto proyectado (p_x,p_y,D) se halla sobre esta recta y tiene $z=D$. De aquí,

$$D = z_1 * t \quad o \quad t = z_1/D$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

- **Sistema Mecánico**: Esta constituido por su estructura mecánica, formada por una serie de elementos mecánicos denominados eslabones, unidos entre sí por articulaciones. Estos elementos conforma tres dispositivos, denominados: Brazo, Muñeca y Elemento Final, normalmente una Pinza.



- **Sistema de Control**: Está formado por uno o varios computadores, que permiten controlar en tiempo real los movimientos del robot, a través de la información que suministran los sensores internos.

- **Generación de la Tarea**: En los robots de la segunda generación es necesario contar con un sistema de generación de tareas compleja a muy alto nivel, como puede ser atornillar, seguimiento de objetos móviles, inserción etc. Este sistema, que puede residir en el computador del sistema de control o encontrarse fuera de éste, trabaja fuera de línea generando programas directamente ejecutables por el sistema de control.

- **Entorno**: Se entiende por tal, el conjunto de objetos que están, o por su movimiento pueden estar, en las proximidades del robot.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

SISTEMA MECANICO

Existen dos tipos de articulaciones: las prismáticas y las de rotación. Las primeras permiten movimientos de traslación de un eslabón respecto al contiguo, mientras las otras permiten giros relativos de los eslavones.

Si cualquier signo de arriba es cero, los segmentos intersectan en alguno de los bordes del span. Para este caso, lo que se hace es calcular la profundidad en el lado opuesto donde se tocan los segmentos en vez de hacerlo en la mitad del span.

La estructura del algoritmo spanning scan line es entonces:

Para preparar los datos:

A. Determinar para cada polígono la mayor línea de rastreo intersectada por el polígono.

B. Colocar el polígono en la celda '**y**' correspondiente a esa línea de rastreo.

C. Almacenar al menos '**y**' (el número de líneas de rastreo que atraviesan el polígono), una lista de los bordes del polígono, los coeficientes de la ecuación del plano (**a**,**b**,**c**,**d**) y los atributos para cada polígono.

Para resolver el problema de superficies ocultas:

Para cada línea de rastreo:

A. Examinar la celda '**y**' correspondiente a esa línea de rastreo para algún nuevo polígono. Añadir el nuevo polígono a la lista de polígonos activos.

B. Examinar la lista de polígonos activos para algún polígono nuevo. Añadir los bordes del polígono nuevo a la lista de bordes activos. La lista de bordes activos contiene la siguiente información, para cada intersección de los bordes del polígono.

- **x** La intersección del borde del polígono con la actual línea de rastreo.
- **Ix** El incremento en **x** de línea de rastreo a línea de rastreo.
- **y** Número de líneas de rastreo atravesadas por el borde del polígono.
- **P** Identificador de polígono.
- **Flag** Indica si el polígono está activo en una línea de rastreo dada.

C. Ordenar la lista de bordes activos en orden creciente de las '**x**'.

D. Procesar la lista de bordes activos. Los detalles se muestran en el

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

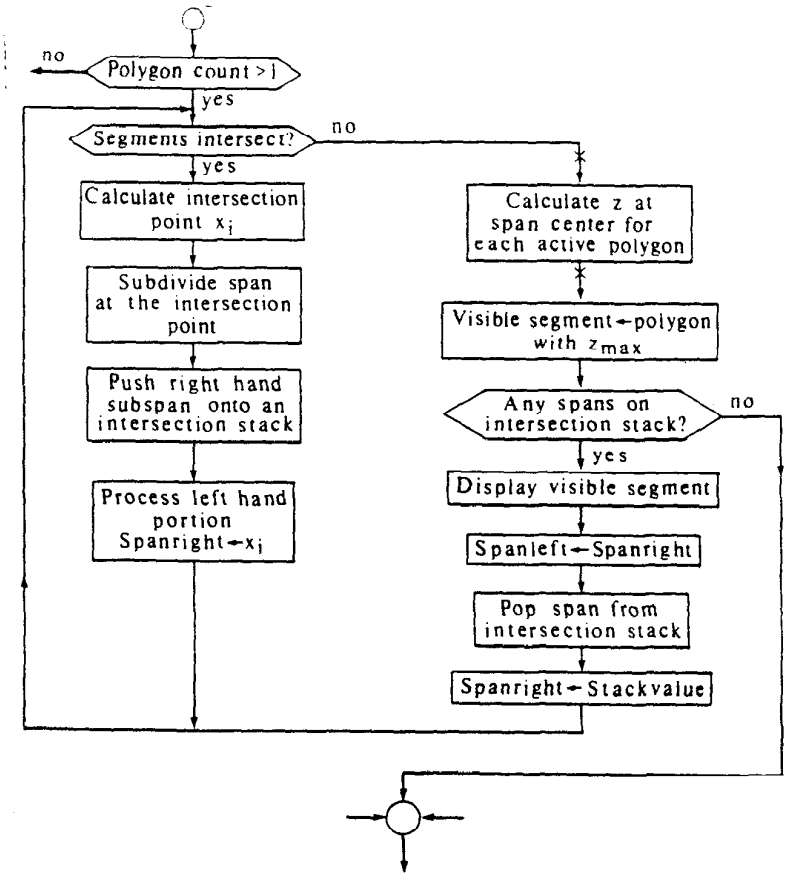


Figura 6.9: Modificación del Diagrama de Flujo de la Figura 6.7 para polígonos que intersectan.

6.5 ALGORITMO RAY TRACING (RAYO TRAZADOR).

La Figura 6.10 ilustra el más simple algoritmo **ray tracing**. El algoritmo supone que la escena ha sido transformada a imagen-espacio. Se supone que el observador está en el infinito en el eje z positivo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

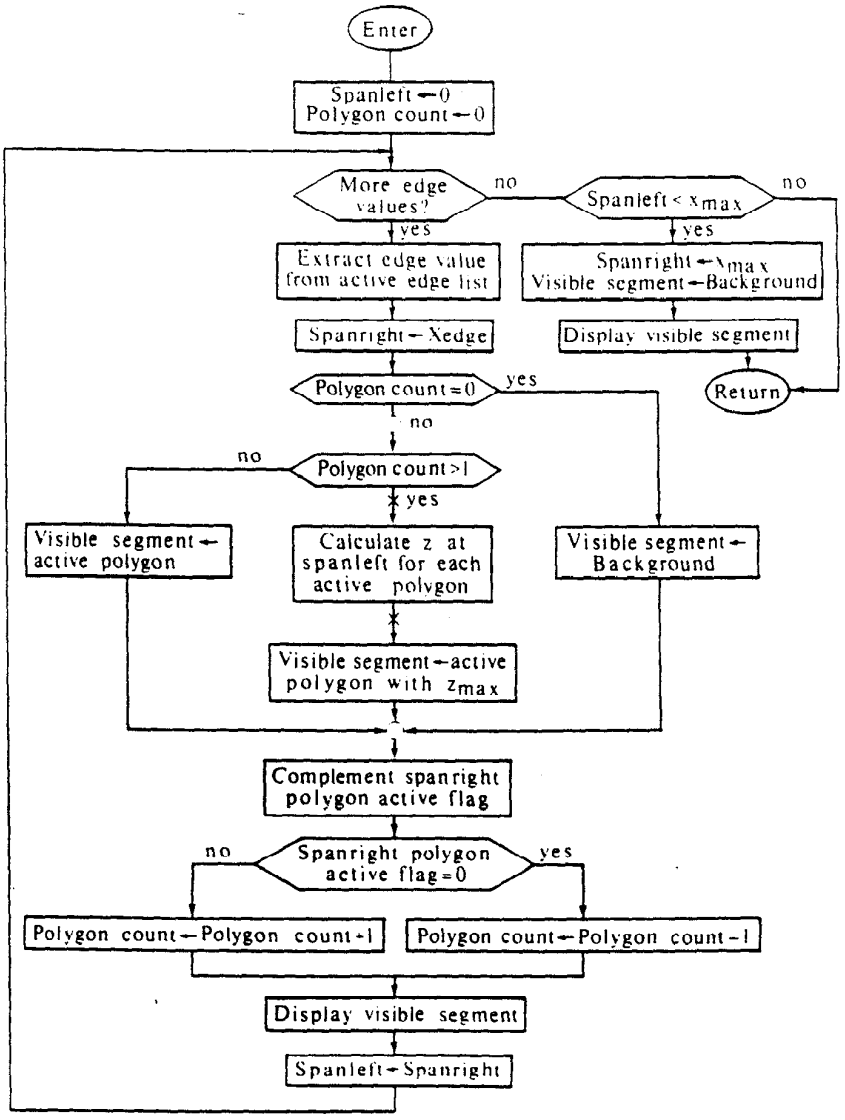


Figura 6.7: Diagrama de Flujo para un span de Polígonos que no se intersectan.

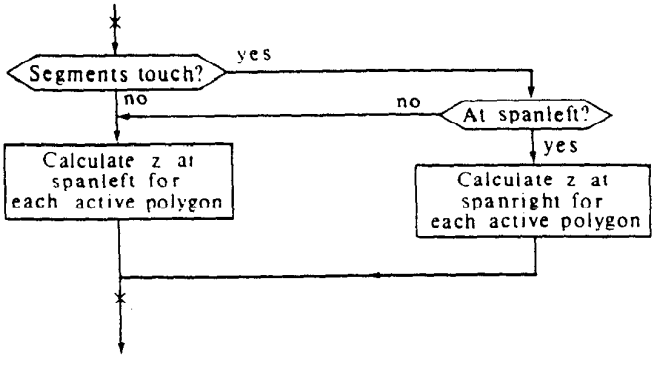


Figura 6.8: Diagrama de Flujo modificado para el cálculo de la profundidad de la Figura 6.7.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2013

diagrama de flujo dado en la Figura 6.7 y las modificaciones dadas en las Figuras 6.8 y 6.9.

E. Actualizar la lista de bordes activos:

Para cada intersección del borde, decrementar '**y**'. Si **y < 0**, eliminar el borde de la lista de bordes activos.

Calcular la nueva '**x**':

$$\mathbf{Xnueva} = \mathbf{Xanterior} + \mathbf{x}$$

F. Decrementar la lista de polígonos activos.

Para cada polígono, decrementar **Yp**. Si **Yp<0** para algún polígono, borrar ese polígono de la lista.

El algoritmo dado en la Figura 6.7 supone que los segmentos de un polígono en un span no intersectan. Si los segmentos intersectan a los lados de un span, entonces, como se discutió anteriormente, el cálculo de profundidad se realiza en el lado opuesto del span donde se produjo la unión de los segmentos. Una simple modificación del bloque de cálculo para el diagrama de flujo mostrado en la Figura 6.7 se da en la Figura 6.8.

La Figura 6.9 muestra las modificaciones del algoritmo dado en la Figura 6.7 cuando se permite la intersección de polígonos. El algoritmo modificado supone que la lista de bordes activos no contiene los puntos de intersección. La intersección de los segmentos debe ser descubierta y procesada sobre la marcha. Aquí, cada span es examinado para ver si existen segmentos intersectados. Si existe alguno, se calcula el punto de intersección y se subdivide el span por el punto de intersección. El subspan del lado derecho se guarda en una pila. El algoritmo se aplica recursivamente al subspan del lado izquierdo, hasta que se encuentre un subspan sin intersecciones de segmentos. Este subspan se muestra, y se saca un nuevo subspan de la pila. El proceso se repite hasta que la pila esté vacía.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Sustituyendo el valor de t en las ecuaciones anteriores, se obtienen los valores del punto proyectado:

$$p_x = D * (x_1/z_1)$$
$$p_y = D * (y_1/z_1)$$

La segunda fórmula se deduce mediante triángulos rectángulos semejantes. La figura 5.16 ilustra una perspectiva superior y una lateral de la figura 5.15. De la propiedad de proporcionalidad de los triángulos semejantes se obtienen las razones:

$$p_x/D = x_1/z_1 \qquad p_y/D = y_1/z_1$$

o

$$p_x = D * (x_1/z_1)$$
$$p_y = D * (y_1/z_1)$$

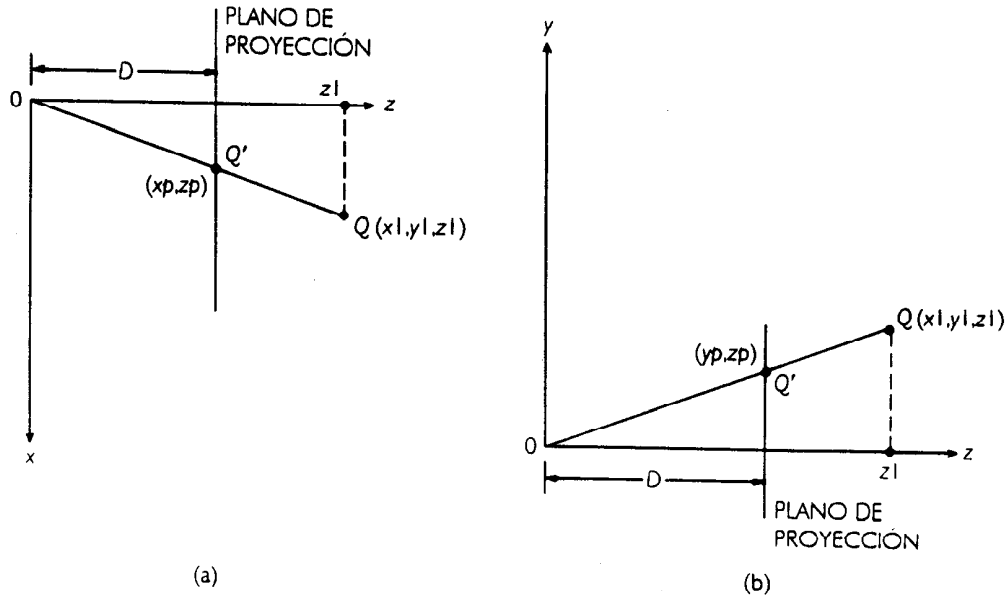


Figura 5.16

Como el valor z representa la distancia de un punto al ojo, la información de profundidad se incluye en estas ecuaciones de proyección. Los objetos más profundos tiene sus coordenadas x,y divididas entre los valores de z mayores que los de los objetos más cercanos, haciendo que los objetos más lejanos se dibujen más pequeños.

Un objeto puede estar a cualquier distancia del ojo. Esto comprende los casos en que los objetos intersectan el plano de proyección o se encuentran atrás del ojo (eje z negativo). Sin embargo, en virtud de que las ecuaciones de proyección en perspectiva implican la división entre los valores de la coordenada z del punto, los objetos no pueden tener el valor z igual a cero.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

La proyección en perspectiva puede describirse usando una matriz. Si D es la distancia del ojo al plano de proyección, la matriz de transformación de perspectiva P_{er}, es:

$$P_{er} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/D \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Al multiplicar el punto homogéneo Q=[x,y,z,1] por la matriz P_{er} se obtiene

$$[x,y,z,1] * P_{er} = [x,y,z,z/D] = Q'$$

Para calcular el punto proyectado (p_x,p_y), es necesario dividir el primero y segundo elementos de Q´ entre su cuarto elemento:

$$p_x = D * (x/z)$$
$$p_y = D * (y/z)$$

La figura 5.17a ilustra este efecto de perspectiva. El rectángulo tridimensional ABCD con coordenadas A=(4,-2,6), B=(4,2,6), C=(-4,2,12), y D=(-4,-2,12) es proyectado sobre el plano que se halla a tres unidades del plano x,y. La figura resultante (Figura 5.17b) es el trapezoide A´B´C´D´ sobre el plano px,py. Las coordenadas de este trapezoide proyectado son:

A´: $p_x = 3 * (4/6) = 2$
$p_y = 3 * (-2/6) = -1$

B´: $p_x = 3 * (4/6) = 2$
$p_y = 3 * (2/6) = 1$

C´: $p_x = 3 * (-4/12) = -1$
$p_y = 3 * (2/12) = 1/2$

D´: $p_x = 3 * (-4/12) = -1$
$p_y = 3 * (-2/12) = -1/2$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

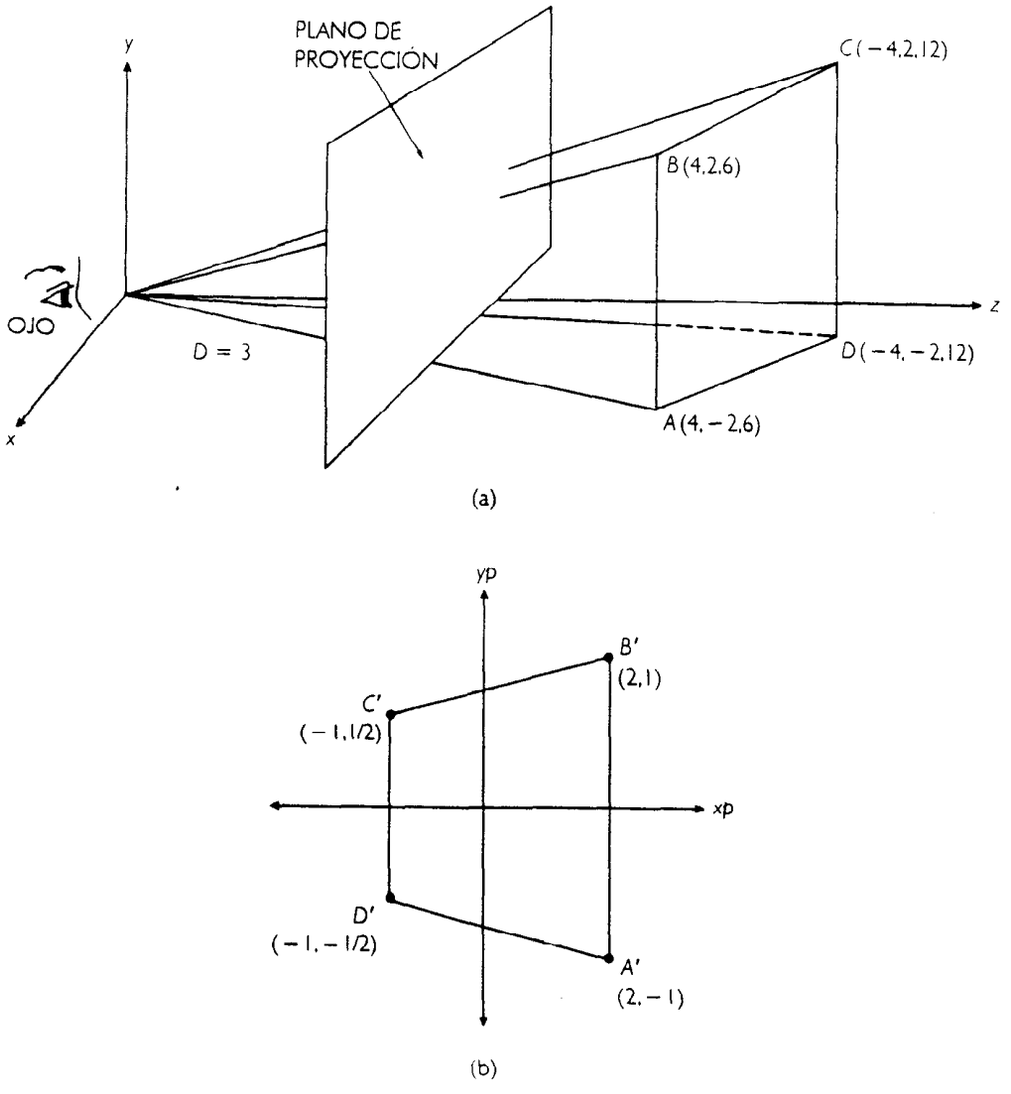


Figura 5.17

5.5 RECORTE EN 3-D.

Antes de generalizar los métodos vistos para el caso de tres dimensiones, es necesario discutir la forma del volumen de recorte. Las formas más comunes son las de una caja y las de la piramide truncada. Estos volumenes son de 6 caras izquierda, derecha, arriba, abajo, frente, detrás.

Como en el recorte de 2-D, las lineas totalmente visible o trivialmente invisibles pueden ser identificadas usando una generalización de los códigos de los puntos extremos de Cohen-Sutherland. Para el recorte de 3-D, se usa un código de 6 bits. De nuevo, el

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

primer bit es el de más a la derecha. Los bits son puesto a 1 según sea :

Primer bit - si el punto está a la izquierda del volumen
Segundo bit - si el punto está a la derecha del volumen
Tercer bit - si el punto está por debajo del volumen
Cuarto bit - si el punto está por encima del volumen
Quinto bit - si el punto está en frente del volumen
Sexto bit - si el punto está detrás del volumen

Para el resto de los casos el bit es puesto a cero. Igualmente que en el caso de 2-D, según sea la operación lógica AND de los códigos de los ptos extremos de la linea :

- Distinta de cero - la línea es totalmente invisible.
- Igual a cero - la línea puede ser parcialmente visible o totalmente invisible.
- Si ambos códigos son cero - la línea es visible.

Para la determinación de los codigos en 3-D se han de realizar unos pasos adicionales debido a la perspectiva del volumen. Se tiene que centrar el volumen en el sistema de coordenadas como muestra la Fig 5.18.

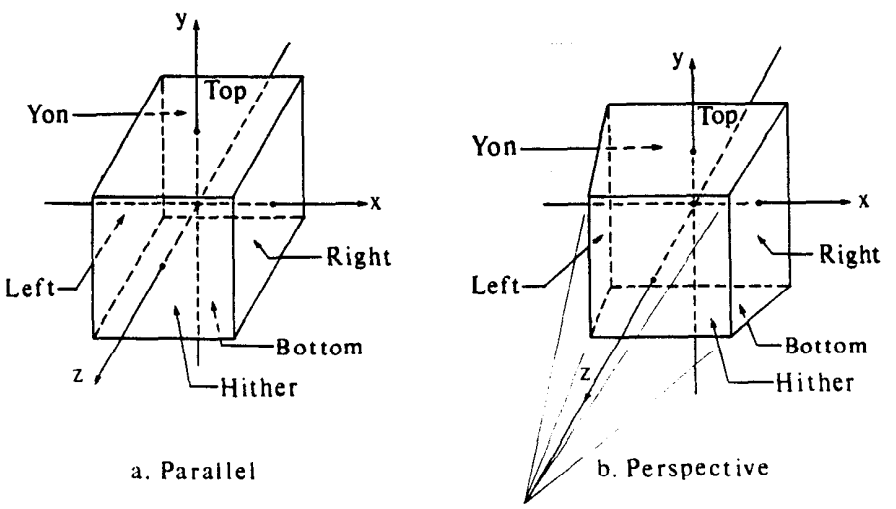


Figura 5.18

En la Fig 5.19 se muestra un visión desde arriba de la perspectiva del volumen de recorte . La ecuación de la línea que representa al lado derecho en esta visión es :

$$x = ((z-z_{cp})/(z_v-z_{cp}))x_r = z\alpha_1 + z\alpha_2;\ \text{donde}\ \alpha_1 = x_r/(z_v-z_{cp});\ \alpha_2 = -\alpha_1 * z_{cp}$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

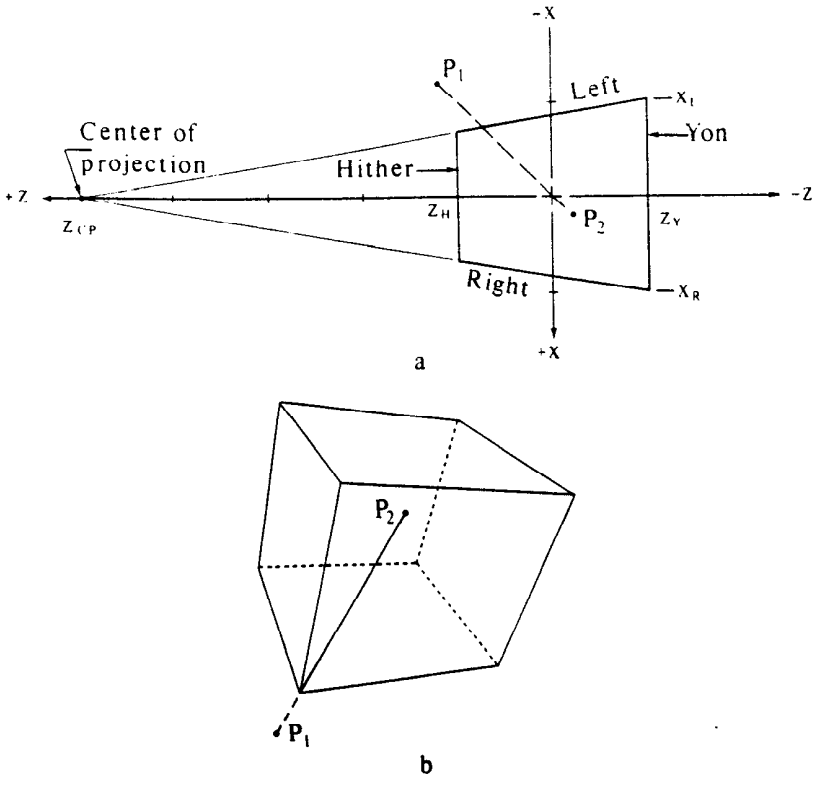


Figura 5.19

La ecuación de este plano puede ser usada para determinar si un punto está a la derecha, sobre, o a la izquierda del plano, esto es, fuera del volumen, sobre el plano derecho, o dentro del volumen, respectivamente. Sustituyendo las coordenadas x,z de un punto P en $x\text{-}z\alpha_1\text{-}\alpha_2$ produce los resultados siguientes :

$f_R = x\text{-}z\alpha_1\text{-}\alpha_2$ > 0 si P está a la derecha del plano
$= 0$ si P está sobre el plano
< 0 si P está a la izquierda del plano

El Test para el caso del los planos izq, sup, inf, frente, detrás son :

$f_L = x\text{-}z\beta_1\text{-}\beta_2$ < 0 si P está a la izquierda del plano
$= 0$ si P está sobre el plano
> 0 si P está a la derecha del plano

donde $\beta 1 = x_l/(z_v\text{-}z_{cp})$ y $\beta_2 = -\beta_1 * z_{cp}$

$f_T = y\text{-}z\mu_1\text{-}\mu_2$ > 0 si P está arriba del plano
$= 0$ si P está sobre el plano
< 0 si P está bajo el plano

donde $\mu_1 = y_t/(z_v\text{-}z_{cp})$ y $\mu_2 = -\mu_1 * z_{cp}$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$f_B = y - z\delta_1 - \delta_2$ < 0 si P está bajo el plano
$= 0$ si P está sobre el plano
> 0 si P está arriba del plano

donde $\delta_1 = z_{cp}$ y $\delta_2 = -\delta_1 * z_{cp}$

$f_H = z - z_h$ > 0 si P está en frente del plano
$= 0$ si P está sobre el plano
< 0 si P está detras del plano

$f_Y = z - z_y$ < 0 si P está a la izquierda del plano
$= 0$ si P está sobre el plano
> 0 si P está a la derecha del plano

Haciendo que z_{cp} se aproxime a infinito, el volumen de recorte se aproxima a un paralelepipedo rectangular (una caja). Como dijeron Liang y Barsky esta aproximación no produce la codifición correcta de los puntos extremos que están detrás del centro de proyección. Esto es debido a que los planos izq, der, sup e inf del volumen de recorte en perspectiva intersectan en el centro de proyección. Entonces un punto puede estar a la derecha de la derecha y a la izquierda de la izquierda simultaneamente. Liang y Barsky sugirieron una técnica para corregir esto. En principio era solo necesario invertir los bits de codigo del izq-der y sup-inf si $z < z_{cp}$.

5.5.1 Algoritmo de Subdivisión del Punto Medio para 3-D.

Es una generalización del visto para el caso de 2-D. Veamos un ejemplo en que aplicamos este algoritmo para el caso de 3-D.

Consideremos la linea P_1(-600,-600,600) P_2(100,100,-100) recortada por el volumen con las coordenadas $x_r = y_t = 500$, $x_l = y_b = -500$ $z_h = 357.14$, $z_y = -500$. El centro de proyección es $z_{cp} = 2500$. Una visión del volumen desde arriba y en perspectiva se muestra en las Fig 5.19a y 5.19b respectivamente. Las funciones para el Test que nos dirá donde se encuentra cualquier punto en relación al volumen son :

Derecha	: $f_r = 6x + z - 2500$
Izquierda	: $f_l = 6x - z + 2500$
Arriba	: $f_t = 6y + z - 2500$
Abajo	: $f_b = 6y - z + 2500$
Frente	: $f_h = z - 357.14$
Atras	: $f_y = z + 2500$

El código del pto extremo para P_1 es (010101), y para P_2 es (000000). Puesto que ambos ptos no son cero, la línea no es totalmente visible. La intersección lógica de ambos codigos es 0 por lo que la línea no es trivialmente invisible. Ya que el codigo de P_2 es (000000), P_2 está dentro del volumen. Por lo que, solo puede ocurrir una intersección con el volumen. El punto medio es :

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$x_m = (x_2 + x_1)/2 = (100 + (-600))/2 = -250$$
$$y_m = (y_2 + y_1)/2 = (100 + (-600))/2 = -250$$
$$z_m = (z_2 + z_1)/2 = (-100 + 600)/2 = 250$$

usando aritmetica entera. El código para el punto medio (000000). El segmento P_mP_2 es totalmente visible. El segmento P_1P_m es parcialmente visible. Continuar con P_1P_m. La subdivisión continua en la Tabla 5.1.

P_1	P_2	P_m	Comment
−600, −600, 600	100, 100, −100	−250, −250, 250	Continue P_1P_m
−600, −600, 600	−250, −250, 250	−425, −425, 425	Continue P_mP_2
−425, −425, 425	−250, −250, 250	−338, −338, 337	Continue P_1P_m
−425, −425, 425	−338, −338, 337	−382, −382, 381	Continue P_mP_2
−382, −382, 381	−338, −338, 337	−360, −360, 359	Continue P_mP_2
−360, −360, 359	−338, −338, 337	−349, −349, 348	Continue P_1P_m
−360, −360, 359	−349, −349, 348	−355, −355, 353	Continue P_1P_m
−360, −360, 359	−355, −355, 353	−358, −358, 356	Continue P_mP_2
−358, −358, 356	−355, −355, 353	−357, −357, 354	Continue P_1P_m
−358, −358, 356	−357, −357, 354	−358, −358, 355	Continue P_mP_2
−358, −358, 355	−357, −357, 354	−358, −358, 354	Success

Tabla 5.1

El actual punto de intersección es (-357.14,-357.14,357.14). La diferencia es debida al uso de la aritmetica entera en el algoritmo.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

5.5.2 Algoritmo de Cyrus-Beck para 3-D.

En el desarrollo del algoritmo de Cyrus-Beck para el recorte en 2-D, la única restricción a la región de recorte es que esta sea convexa, por lo que puede ser un volumen convexo tridimensional. Entonces podemos aplicar directamente el algoritmo. En lugar de que k sea el nº de bordes, es ahora el nº de planos. Todos los vectores tienen tres componentes; x,y,z. La conversión de la subrutina del cálculo del producto escalar para el caso de vectores de 3-D es sencilla. Para una mejor ilustración del algoritmo vemos los siguientes ejemplos. El primero considera el recorte de una caja.

Ejemplo: ALGORITMO DE CYRUS-BECK PARA 3-D

Una línea desde P_1(-2,-1,1/2) a P_2(3/2,3/2,-1/2) sera recortada por el volumen(x_l,x_r,y_b,y_t,z_h,z_y) = (-1,1,-1,1,1,-1) como se muestra en la Fig 5.20. Por inspección los seis normales interiores son:

Superior:	$n_t = -j = [\ 0\ -1\ \ 0]$	Frente : $n_h = [\ 0\ \ 0\ -1]$
Inferior:	$n_b = \ j = [\ 0\ \ 1\ \ 0]$	Atras : $n_y = [\ 0\ \ 0\ \ 1]$
Derecho:	$n_r = -i = [-1\ \ 0\ \ 0]$	
Izquierdo:	$n_l = \ i = [\ 1\ \ 0\ \ 0]$	

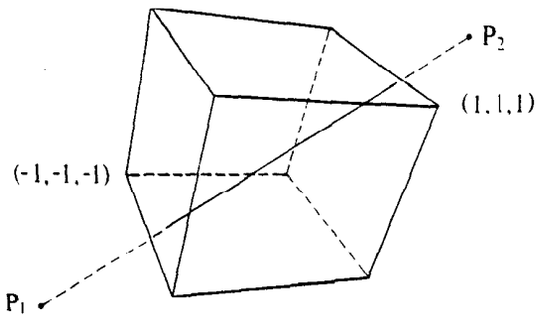


Figura 5.20

Los puntos en cada plano de recorte tambien deben ser seleccionados por inspección. Son suficientes la elección de dos puntos en el final de la diagonal entre esquinas opuestas del volumen de recorte. Entonces $f_T = f_R = f_H(1,1,1)$ y $f_R = f_L = f_Y(-1,-1,-1)$

Alternativamente un punto del centro o de una esquina de cada plano de recorte podría ser usado. La dirección para la linea P_1P_2 es

$$D = P_2 - P_1 = [3/2\ 3/2\ -1/2] - [-2\ -1\ 1/2] = [7/2\ 5/2\ -1]$$

Para el punto del borde $f_L(-1,-1,-1)$

$$w = P_1 - f = [-2\ -1\ 1/2] - [-1\ -1\ -1] = [-1\ 0\ 3/2]$$

Para la parte izquierda del plano de recorte el vector normal hacia dentro es:

$$n_L = [1\ 0\ 0]$$

Por lo tanto:

$$D * n_L = [7/2\ 5/2\ -1] * [1\ 0\ 0] = 7/2 > 0$$

y hemos de buscar el limite menor : $w * n_L = [-1\ 0\ 3/2] * [1\ 0\ 0] = -1$ y

$$t = -(-1/(7/2)) = 2/7$$

La Tabla 5.2 da los resultados completos

Plane	n	f	w	w · n	D · n†	t_L	t_U
Top	[0 −1 0]	(1, 1, 1)	[−3 −2 −1/2]	2	−5/2		4/5
Bottom	[0 1 0]	(−1, −1, −1)	[−1 0 3/2]	0	5/2	0	
Right	[−1 0 0]	(1, 1, 1)	[−3 −2 −1/2]	3	−7/2		6/7
Left	[1 0 0]	(−1, −1, −1)	[−1 0 3/2]	−1	7/2	2/7	
Hither	[0 0 −1]	(1, 1, 1)	[−3 −2 −1/2]	1/2	1	−1/2	
Yon	[0 0 1]	(−1, −1, −1)	[−1 0 3/2]	3/2	−1		3/2

Tabla 5.2

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$D*n<0$ limite superior (t_u), $D*n>0$ limite inferior (t_l).

Examinando la Tabla 5.2 se muestra que el máximo limite inferior es $t_l = 2/7$ y el minimo limite superior es $t_u = 4/5$. La ecuación parametrica de la linea P_1P_2 es

$$P(t) = [-2\ -1\ 1/2] + [7/2\ 5/2\ -1]t$$

Substituyendo t_l y t_u :

$$P(2/7) = [-1\ -2/7\ 3/14] \text{ y } P(4/5) = [4/5\ 1\ -3/10]$$

como los puntos de intersección con los planos izquierdo y superior del volumen de recorte.

Ejemplo: RECORTE DE UN VOLUMEN EN PERSPECTIVA

Consideramos la misma linea que en el ejemplo anterior y el mismo volumen con el centro de proyección en $z_{cp} = 5$. Ver Fig 5.18b.

El vector normal de entrada para los planos de recorte del frente y de detras pueden ser obtenidos por inspección. El resto de ellos pueden ser calculados desde el producto en cruz de los vectores desde el centro de proyección a las esquinas en $z=0$, el plano de proyección. Esos vectores son :

$$V_1 = [\ 1\ \ 1\ -5]$$
$$V_2 = [-1\ \ 1\ -5]$$
$$V_3 = [-1\ -1\ -5]$$
$$V_4 = [\ 1\ -1\ -5]$$

Los vectores normales de entrada son

$$n_t = V_1 * V_2 = [\ 0\ -10\ -2]$$
$$n_l = V_2 * V_3 = [\ 10\ \ 0\ -2]$$
$$n_b = V_3 * V_4 = [\ 0\ \ 10\ -2]$$
$$n_r = V_4 * V_1 = [-10\ \ 0\ -2]$$
$$n_h = [0\ 0\ -1]$$
$$n_v = [0\ 0\ \ 1]$$

Ya que el centro de proyección esta en cuatro de los seis planos, es conveniente hacer :

$$f_T = f_L = f_B = f_R = (0,0,5)$$

y el centro de los planos del frente y de detras

$$f_H(0,0,1) \text{ y } f_Y(0,0,-1)$$

como los puntos de los bordes.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

La dirección de P_1P_2 es:

$$D = P_2 - P_1 = [7/2\ 5/2\ -1]$$

Para el punto del borde en el plano izquierdo de recorte

$$w = P_1 - f_L = [-2\ -1\ 1/2] - [0\ 0\ 5] = [-2\ -1\ -9/2]$$

Notese que:

$$D * n_l = [7/2\ 5/2\ -1] * [10\ 0\ -2] = 37 > 0$$

y hallando el limite inferior, tenemos :

$$w * n_l = [-2\ -1\ -9/2] * [10\ 0\ -2] = -11; \qquad t_l = -(-11/37) = 0.297$$

La Tabla 5.3 da los resultados completos.

Plane	n	f	w	w · n	D · n	t_L	t_U
Top	[0	−10 −2]	(0, 0, 5)	[−2 −1	−9/2]	19	−23		0.826
Bottom	[0	10 −2]	(0, 0, 5)	[−2 −1	−9/2]	−1	27	0.037	
Right	[−10	0 −2]	(0, 0, 5)	[−2 −1	−9/2]	29	−33		0.879
Left	[10	0 −2]	(0, 0, 5)	[−2 −1	−9/2]	−11	37	0.297	
Hither	[0	0 −1]	(0, 0, 1)	[−2 −1	−1/2]	1/2	1	−0.5	
Yon	[0	0 1]	(0, 0, −1)	[−2 −1	3/2]	3/2	−1		1.5

Tabla 5.3

$D*n<0$ limite superior (t_u), $D*n>0$ limite inferior (t_l).

La Tabla 5.3 muestra que el limite inferior maximo es $t_l = 0.297$ y el limite superior minimo es $t_u = 0.826$. De la ecuación parametrica los valores de intersección son

$$P(0.297) = [-0.961\ -0.258\ 0.203] \quad y \quad P(0.826) = [0.8911.065\text{-}0323]$$

para los planos de recorte izquierdo y superior.

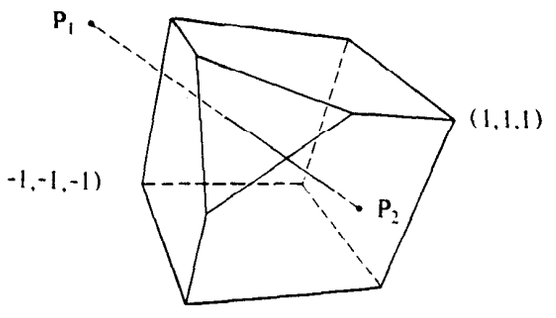


Figura 5.21

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Ejemplo: RECORTE POR UN VOLUMEN ARBITRARIO

El volumen de recorte se muestra en la Fig 5.21. Es un cubo con una esquina cortada. Las coordenadas de los vertices de cada cara son :

Derecha: (1,-1,1),(1,-1,-1),(1,1,-1),(1,1,1)
Izquierda: (-1,-1,1),(-1,-1,-1),(-1,1,-1),(-1,1,0),(-1,0,1)
Inferior: (1,-1,1),(1,-1,-1),(-1,-1,-1),(1,-1,-1)
Superior: (1,1,1),(1,1,-1),(-1,1,-1),(-1,1,0),(0,1,1)
Frente: (1,-1,1),(1,1,1),(0,1,1),(-1,0,1),(-1,-1,1)
Atras: (-1,-1,-1),(1,-1,-1),(1,1,-1),(-1,1,-1)
Oblicua: (-1,0,1),(0,1,1),(-1,1,0)

La Tabla 5.4 da los resultados completos para la linea $P_1(-2,3/2,1)$ $P_2(3/2,-1,-1/2)$ recortada en este volumen.

Plane	n	f	w	w * n	D * n†	t_L	t_U
Top	[0 −1 0]	(1, 1, 1)	[−3 1/2 0]	−1/2	5/2	1/5	
Bottom	[0 1 0]	(−1, −1, −1)	[−1 5/2 2]	5/2	−5/2		1
Right	[−1 0 0]	(1, 1, 1)	[−3 1/2 0]	3	−7/2		6/7
Left	[1 0 0]	(−1, −1, −1)	[−1 5/2 2]	−1	7/2	2/7	
Hither	[0 0 −1]	(1, 1, 1)	[−3 1/2 0]	0	3/2	0	
Yon	[0 0 1]	(−1, −1, −1)	[−1 5/2 2]	2	−3/2		4/3
Skew	[1 −1 −1]	(−1, 0, 1)	[−1 3/2 0]	−5/2	15/2	1/3	

Tabla 5.4

$D*n<0$ limite superior (t_u), $D*n>0$ limite inferior (t_l).

De la Tabla se ve que el limite inferior maximo es $t_l = 1/3$, y el limite superior minimo es $t_u = 6/7$. Los puntos de intersección son :

$P(1/3) = [-5/6\ 2/3\ 1/2]$ y $P(6/7) = [1\ -9/14\ -2/7]$

en el plano oblicuo y derecho respectivamente.

Notese que el coste computacional del algoritmo Cyrus-Beck crece linealmente con el numero de bordes o planos del volumen.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

CAPITULO 6

SUPRESION DE SUPERFICIES Y LINEAS OCULTAS.

6.1 INTRODUCCION.

Una consideración importante en la generación de escenas realistas es la identificación y supresión de las partes de la imagen definida que no son visibles desde una posición de observación seleccionada. Existen muchos métodos que podemos utilizar para resolver este problema y se han creado numerosos algoritmos para eliminar las partes ocultas de escenas en forma eficaz en diferentes tipos de aplicaciones. Algunos métodos requieren más memoria, en otros interviene más el tiempo de procesamiento y otros sólo se aplican a tipos especiales de objetos. El método elegido para una aplicación determinada depende de factores, tales como, la complejidad de la escena, los tipos de objetos que se desplegarán, el equipo del cual se dispone y si se van a generar despliegues animados o bien estáticos. En este capítulo, exploramos algunos de los métodos que se usan más comunmente para eliminar superficies y líneas ocultas.

6.2 CLASIFICACION DE ALGORITMOS.

Los algoritmos de líneas y superficies ocultas, a menudo se clasifican según se refieren a definiciones de objetos en forma directa o bien con sus imágenes proyectadas. Estos dos métodos se denominan métodos de **objeto-espacio** y métodos de **imagen-espacio**, respectivamente. El primer método compara objetos y partes de objetos unos con otros, para determinar qué superficies y líneas, como un todo, deben rotularse como invisibles. En un algoritmo de imagen-espacio, la visibilidad se decide punto por punto en cada posición de pixel sobre el plano de proyección. Muchos algoritmos de superficie oculta aplican métodos de imagen-espacio, pero los métodos de objeto-espacio pueden usarse en forma eficaz en algunos casos. Los algoritmos de línea oculta, por lo general, emplean métodos de objeto-espacio, aunque muchos algoritmos de superficie oculta de imagen-espacio pueden adaptarse fácilmente a la supresión de líneas ocultas.

Aunque existen diferencias importantes en el método básico abordado por los diversos algoritmos de superficie y línea oculta, muchos de ellos emplean métodos de ordenamiento y coherencia para mejorar el rendimiento. El ordenamiento se utiliza para facilitar las comparaciones de profundidad mediante el ordenamiento de las líneas, superficies y objetos individuales de una escena, de acuerdo con su distancia desde el plano de visión. Los métodos de coherencia se usan para aprovechar las regularidades de una escena.

6.3 ALGORITMO DEL HORIZONTE FLOTANTE.

El horizonte flotante es el algoritmo más frecuentemente usado para borrar líneas ocultas en la representación de funciones de superficies de la forma:

$$F(x,y,z) = 0$$

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Funciones de esta forma aparecen en diversas disciplinas, tales como, la ingeniería, las matemáticas, etc.

Gran número de algoritmos han sido desarrollados usando esta técnica. Dado que la representación de la función es el principal interés, el algoritmo, normalmente, se implementa en **imagen-espacio**. La idea fundamental de este algoritmo es la de convertir un problema tridimensional en uno de dos dimensiones por intersección de superficies con una serie de planos paralelos cortantes a unos valores constantes de **x**, **y** o **z**. Esto se muestra en la Figura 6.1, donde valores constantes de **z** definen planos paralelos.

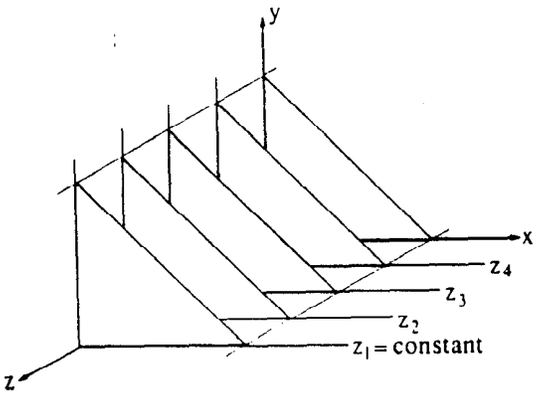


Figura 6.1: Planos Cortantes de Coordenadas Constantes.

La función **F (x,y,z) = 0** es reducida a una curva en cada uno de estos planos paralelos, tal que:

$$y = f(x,z) \quad o \quad x = g(y,z)$$

donde **z** es una constante para cada uno de los planos paralelos.

De este modo, la superficie es construída como una serie de curvas en estos planos, tal y como se muestra en la Figura 6.2. Si el resultado es proyectado en el plano **z = 0** como se muestra en la Figura 6.3, un algoritmo para borrar porciones ocultas de una superficie, se reconoce inmediatamente. El algoritmo primero ordena los planos **z = constante** por incremento de la distancia con el punto de vista. Comenzando con el plano más cercano al punto de vista, se genera la curva en cada plano; tal que para cada valor de la coordenada **x**, se determina la coordenada **y**. Entonces, el algoritmo de línea oculta es:

Si dado un valor de **x**, el valor de la coordenada **y** de la curva en el plano actual es mayor que cualquier valor de **y** de curvas de planos anteriores para ese valor de **x**, entonces la curva es visible en ese valor de **y**. De otro modo, está oculta.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

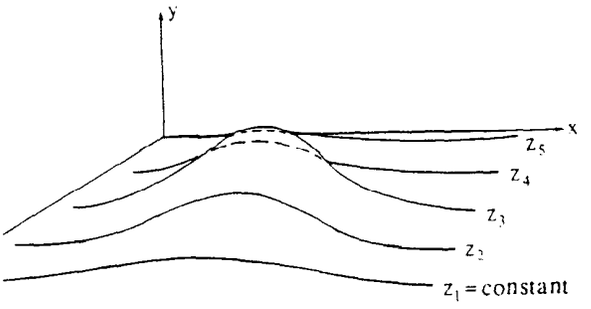


Figura 6.2: Curvas en Planos Cortantes de Coordenadas Constantes.

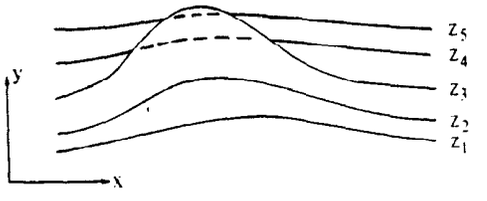


Figura 6.3

Esto se muestra en la Figura 6.3 por medio de las líneas discontinuas. La implementación de este algoritmo es bastante simple. Un array de tamaño igual a la resolución en la dirección **x**, contendrá la **y** máxima para cada valor de **x**. Los valores de este array representan el horizonte actual.

El algoritmo trabaja bien para superficies que van en ascenso; sin embargo, si alguna de las siguientes curvas tiene algún punto, **y**, inferior al de la primera curva que se dibujó en el primer plano, entonces, el algoritmo falla, ya que ese trozo de curva se debería de ver como se muestra en la Figura 6.4a. La solución para esto, es la de mantener otro array con los valores de **y** más pequeños que vayan apareciendo para cada valor de **x**. Ahora, el algoritmo sería:

Si dado un valor de **x**, el valor de **y** de la curva en el plano actual es mayor que el máximo o menor que el mínimo valor de **y** para cualquier curva previa a ese valor de **x**, entonces, la curva es visible en ese punto. De otro modo, está oculta.

El resultado se muestra en la Figura 6.4b. La Figura 6.5 muestra un resultado típico del horizonte flotante.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

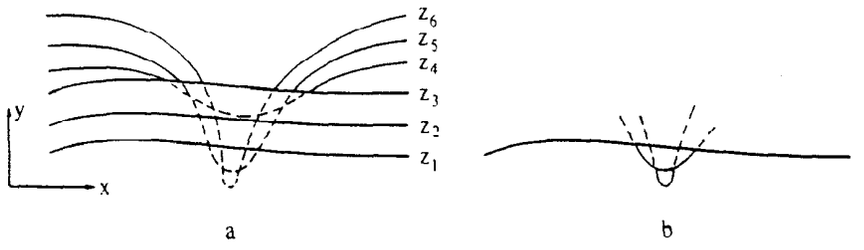


Figura 6.4: Manejo del lado más bajo de la superficie.

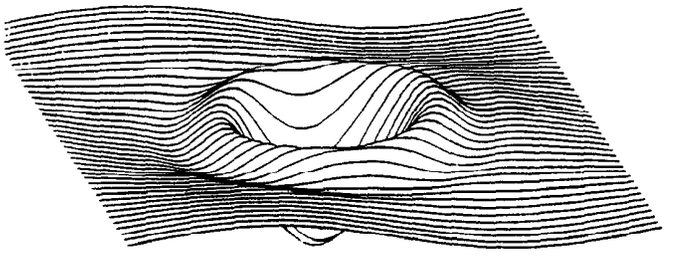


Figura 6.5: La Función:
$y = (1/5) \text{ sen } x \cos z - (3/2) \cos (7a/4) \exp(-a)$,
donde $a = (x - \pi)^2 + (z - \pi)^2$
mostrada para valores de 0 a 2π usando un algoritmo del horizonte flotante.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

6.4 ALGORITMO SPANNING SCAN LINE.

El algoritmo de la línea de rastreo calcula la profundidad de los polígonos en cada pixel de la línea de rastreo. El número de cálculos de profundidad puede reducirse introduciendo el concepto de **span**. La Figura 6.6a muestra la intersección de dos polígonos con el plano de rastreo. Dividiendo el plano de rastreo en cada intersección que el polígono tiene con el plano de rastreo, aparecen los spans; la solución del problema de superficies ocultas se reduce a seleccionar el segmento visible en cada span. La Figura 6.6a muestra que solamente son posibles tres tipos de spans:

A. El span está vacío, por ejemplo el span 1 de la Figura 6.6a.

B. El span contiene solamente un segmento, por ejemplo los spans 2 y 4 de la Figura 6.6a.

C. El span contiene múltiples segmentos, por ejemplo el span 3 de la Figura 6.6a. Se calcula la profundidad de cada segmento en el span. El segmento con el valor mayor de **z**, es el segmento visible.

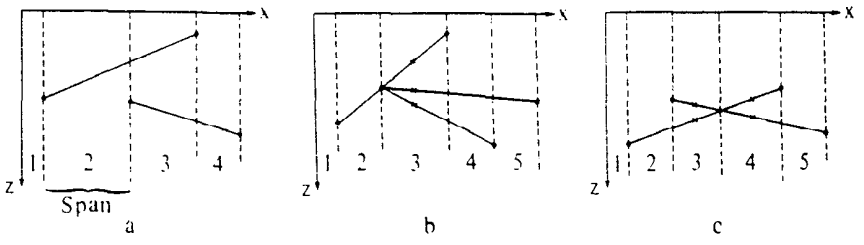


Figura 6.6: Spans de Línea de rastreo.

Si no se permiten polígonos intersectantes, es suficiente calcular la profundidad de cada segmento en un span. Si dos segmentos se tocan pero no se intersectan en el final del span, el cálculo de profundidad se realiza en la mitad del span como se muestra en la Figura 6.6b. Para el span 3, el cálculo de profundidad realizado en el borde izquierdo del span, produce un resultado incorrecto. Realizando el cálculo de profundidad en la mitad del span, como se muestra marcado en la Figura 6.6b, produce un resultado correcto.

Si se permiten polígonos intersectantes, entonces el plano de rastreo no se divide sólo con las intersecciones de los bordes del polígono con dicho plano sino también por las intersecciones de los segmentos como se muestra en la Figura 6.6c.

La estructura básica desarrollada para el algoritmo de línea de rastreo también es aplicable a éste. Aquí no es necesario mantener las intersecciones de los polígonos con la línea de rastreo en pares. Se colocan bordes individuales en la lista de bordes activos. Esta lista se ordena por x crecientes. A cada polígono se le asocia un flag y un identificador, los cuales se usan para identificar los elementos izquierdo y derecho del par de bordes. El flag inicialmente está a falso y se complementa cada vez que se procesa un borde del polígono. Cuando se encuentra el borde izquierdo del polígono, el flag se pone a 'cierto', mientras que encontrando el borde derecho se pone a 'falso'.

Los spans para cada polígono pueden determinarse cada vez que se procese una línea de rastreo. Si no se permiten intersecciones de polígonos, cada intersección de los bordes con el plano de rastreo, representa uno de los límites de un span. Como se discutió anteriormente, el número de polígonos activos dentro de un span determina como va a procesarse el span. Los cálculos de profundidad sólo se realizan si hay más de un polígono dentro de un span. Si se permiten las intersecciones de polígonos, y más de un polígono está activo dentro de un span determinado, entonces es necesario chequear si existen intersecciones (Figura 6.6c). Un método para realizar ésto, es comparar los signos de las diferencias de profundidad entre pares de segmentos en los límites del span. Cada par de segmentos dentro del span debe ser examinado. Por ejemplo, si dos segmentos tienen profundidades Z_{1l}, Z_{1r}, Z_{2l}, Z_{2r} a la izquierda y a la derecha, entonces:

$$\text{SI SIGN } (Z_{1l} - Z_{2l}) <> \text{ SIGN } (Z_{1r} - Z_{2r})$$

los segmentos intersectan. Si los segmentos intersectan, entonces se subdivide el span en la intersección. El proceso se repite con el span que está a mano izquierda hasta que esté libre de intersecciones. Para este span, el cálculo de profundidad se realiza en el centro del span.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

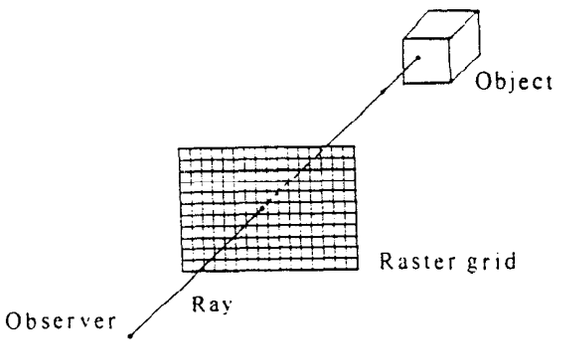


Figura 6.10: Rayo trazador simple.

De este modo, todos los rayos de luz son paralelos al eje **z**. Cada rayo sale del observador, atraviesa la rejilla (raster) y llega a la escena. El camino de cada rayo se traza para determinar qué objeto, si lo hay, intersecta el rayo. Cada objeto de la escena debe ser examinado para cada rayo. Si el rayo intersecta el objeto, se determinan todas las posibles intersecciones entre el rayo y el objeto. Esto puede producir múltiples intersecciones para múltiples objetos. Las intersecciones se ordenan según la coordenada **z** (profundidad). La intersección con el máximo valor de **z** representa la superficie visible para ese pixel. Los atributos para esa superficie se colocan en ese pixel. El elemento más importante para este algoritmo es la rutina que realiza las intersecciones entre el rayo y los polígonos. Debido a que este algoritmo gasta entre un 75-90% de su esfuerzo en realizar las intersecciones, la eficiencia de la rutina de intersección afecta significativamente a la eficiencia del algoritmo. Determinar las intersecciones de una línea arbitraria (un rayo) con un objeto, puede ser computacionalmente caro. Para eliminar cálculos innecesarios de intersecciones, lo que se hace es determinar la intersección del rayo con el volumen limitante (bounding volume) del objeto. Si el rayo no intersecta al volumen limitante del objeto, entonces el objeto no es examinado. Cualquier caja limitante o esfera limitante puede utilizarse como volumen limitante. Aunque, como se muestra en la Figura 6.11, una esfera limitante pueda parecer ineficiente, determinar si un rayo tridimensional atraviesa una esfera, es bastante simple. En particular, si la distancia del centro de la esfera al rayo es mayor que el radio de la esfera, entonces el rayo no intersecta a la esfera limitante. De este modo, tampoco puede intersectar al objeto.

La mínima distancia **D** de una línea a un punto P_o (X_o, Yo, Z_o) es:

$$D^2 = (X-X_o)^2 + (Y-Y_o)^2 + (Z-Z_o)^2$$

Si $D^2 > R^2$, donde **R** es el radio de la esfera limitante del objeto, entonces el rayo no puede intersectar al objeto.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

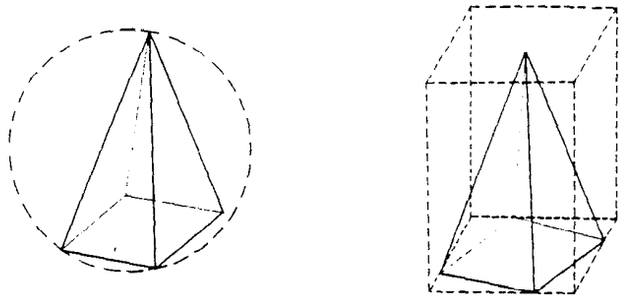


Figura 6.11: Volúmenes limitantes.

Realizar la intersección entre un rayo y una caja limitante (bounding box) resulta computacionalmente caro. Para evitar esto, lo que se hace es trasladar el rayo de manera que coincida con el eje z; de igual manera, la traslación que se ha hecho con el rayo, también se realiza con la caja limitante del objeto. El rayo intersecta la caja limitante si, en el sistema coordenado trasladado, los signos de X_{min} y X_{max} son opuestos a los de Y_{min} e Y_{max} como se muestra en la Figura 6.12.

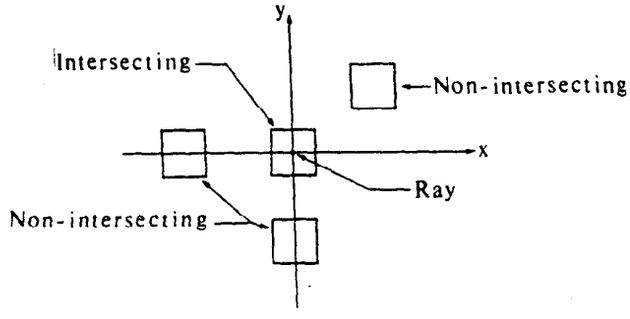


Figura 6.12: Intersecciones de la caja limitante en el sistema de coordenadas transformado.

Para múltiples intersecciones de un rayo con varios objetos, es necesario determinar cuál es la intersección visible. Para este algoritmo, la intersección con la máxima coordenada **z** es la superficie visible.

El algoritmo **ray tracing** para superficies opacas sería entonces:

Preparar los datos de la escena:

Crear una lista de objetos que contenga, al menos, las siguientes cosas:

A. Descripción completa de los objetos: tipo, superficie, características, etc.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

B. Centro y radio de la esfera limitante.

C. Flag de la caja limitante. Si el flag está activado, se realizará el test; sino será saltado.

D. Descripción de la caja limitante: X_{min}, X_{max}, Y_{min}, Y_{max}.

Para cada rayo trazador:

Para cada objeto realizar el test de la esfera limitante. Si el rayo intersecta la esfera limitante, colocar el objeto en la lista de objetos activos.

Si la lista de objetos activos está vacía, mostrar el pixel con el color de fondo y continuar con el siguiente rayo.

Sino, trasladar el rayo de tal manera que coincida con el eje **z**. Guardar la traslación realizada.

Para cada objeto de la lista de objetos activos hacer:

Si el flag de la caja limitante está activado, realizar la misma traslación que se le hizo al rayo y realizar el test. Si el rayo no intersecta la caja, entonces continuar con el siguiente objeto de la lista de objetos activos.

Si el rayo intersecta la caja limitante, transformar también el objeto y determinar la intersección del rayo con el objeto, si la hay. Colocar la intersección en la lista de intersecciones.

Si la lista de intersecciones está vacía, displayar el pixel con la intensidad de fondo.

Sino, determinar la intersección con mayor profundidad de la lista de intersecciones (Z_{max}).

Calcular la traslación inversa.

Aplicar la traslación inversa al punto de intersección para que esté en su lugar original.

Displayar el pixel usando los atributos del objeto intersectado.

6.6 ALGORITMO DEL BUFFER-Z.

Un método de imagen-espacio que se utiliza comúnmente para eliminar superficies ocultas es el método del **buffer de profundidad, también conocido como método del buffer-z**. Básicamente, este algoritmo verifica la visibilidad de superficies punto por punto. En

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

cada posición del pixel (x,y) sobre el plano de visión, es visible la superficie con la menor coordenada z en dicha posición. La Figura 6.13 muestra tres superficies en varias profundidades con respecto a la posición (x,y) en un sistema de visión del lado izquierdo.

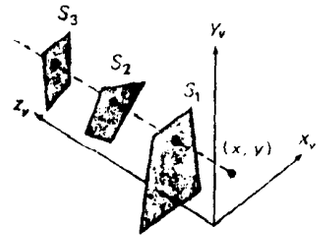


Figura 6.13: En la posición (x,y), la superficie S_1 tiene el menor valor de profundidad y de esta manera es visible en esa posición.

La superficie S_1 tiene el menor valor de z en esta posición, de manera que se salva su valor de intensidad en (x,y).

Se requieren dos áreas diferentes para implantar este método. Se utiliza un buffer de profundidad para almacenar valores de z para cada posición (x,y) a medida que se comparan las superficies y el buffer de renovación almacena los valores de intensidad de cada posición.

La profundidad en los puntos situados sobre la superficie de un polígono se calcula a partir de la ecuación del plano. Inicialmente, todas las posiciones del buffer de renovación se hacen igual a la profundidad máxima y el buffer de renovación se inicializa con la intensidad del fondo. Cada superficie incluída en las tablas de polígonos se procesa después, una línea de rastreo a la vez, calculando la profundidad o valor de z, en cada posición (x,y). El valor de z calculado se compara con el valor que se almacenó previamente en el buffer de profundidad en esa posición; si el valor de z calculado es menor que el valor almacenado en el buffer de profundidad, el nuevo valor de z se almacena y la intensidad de la superficie en esa posición se coloca en la misma posición del buffer de renovación.

Podemos resumir las etapas de un algoritmo de buffer de profundidad como sigue:

1. Inicializar el buffer de profundidad y el de renovación, de manera que, para todas las posiciones coordenadas (x,y), **buffer_profundidad (x,y) = max_profundidad** y **buffer_renovación (x,y) = background**.

2. Para cada posición de cada superficie, comparar los valores de la profundidad con los valores previamente almacenados en el buffer de profundidad para determinar la visibilidad.

   A. Calcular valores de z para cada posición (x,y) de la superficie.

   B. Si z < **buffer_profundidad** (x,y), entonces hacer

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

buffer_profundidad (x,y) = z y **buffer_renovación (x,y)** = i, donde i es el valor de la intensidad de la superficie en la posición (x,y).

En el último paso, si z no es menor que el valor del buffer de profundidad de esa posición, el punto no es visible. Cuando se haya realizado este proceso en todas las superficies, el buffer de profundidad contiene valores de z para las superficies visibles y el buffer de renovación contiene sólo los valores de intensidad visible.

Los valores de profundidad de una posición (x,y) se calculan a partir de la ecuación del plano de cada superficie:

$$z = - (Ax + By + D) / C$$

Para cualquier línea de rastreo (Figura 6.14), tanto las coordenadas x que atraviesan la línea, como los valores de y entre las líneas, difieren en 1.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

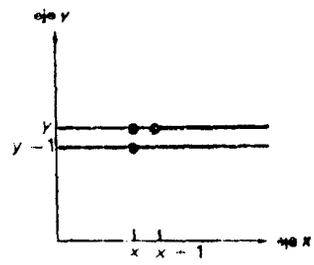


Figura 6.14: De la posición (x,y) en una línea de rastreo, la siguiente posición que cruza la línea tiene las coordenadas (x + 1, y), y la posición que está inmediatamente debajo de la siguiente línea, tiene las coordenadas (x, y - 1).

Si la profundidad de la posición (x,y) se ha determinado como z, la profundidad z' de la siguiente posición (x + 1,y) a lo largo de la línea de rastreo se obtiene a partir de la ecuación anterior como:

$$z' = - [A(x+1) + By + D] / C$$

o bien

$$z' = z - A / C$$

La razón A / C es constante en cada superficie, de manera que los valores de profundidad sucesivos a través de una línea de rastreo, se obtienen a partir de los valores anteriores con sólo una resta.

Podemos obtener valores de profundidad entre líneas de rastreo en forma similar. Una vez más, supongamos que la posición **(x,y)** tiene la profundidad **z**. Entonces en la posición **(x,y-1)** en la línea de rastreo que está inmediatamente debajo, el valor de profundidad se calcula a partir de la ecuación del plano como:

$$z'' = -[Ax + B(y-1) + D] / C$$

o bien

$$z'' = z + B / C$$

Lo que requiere una suma de la constante **B / C** al valor de profundidad anterior **z**.

El método del buffer de profundidad es fácil de implantar y no requiere el ordenamiento de las superficies de una escena. Pero sí requiere disponer de un segundo buffer, el buffer de renovación. Por ejemplo, un sistema con una resolución de 1024x1024 requeriría más de un millón de posiciones en el buffer de profundidad, con cada posición que contiene los bits suficientes para representar el número de incrementos de la coordenada **z** que se necesitan. Una manera de reducir los requisitos de almacenamiento consiste en procesar una sección de la escena cada vez, utilizando un buffer de profundidad de menor tamaño. Después de que se procesa cada sección de la vista, el buffer se vuelve a utilizar en la siguiente sección.

6.7 METODO DE LA LINEA DE RASTREO.

Este método de imagen-espacio que se usa para eliminar superficies ocultas es una extensión del algoritmo de la línea de rastreo para llenar interiores de polígonos. En vez de llenar sólo una superficie, ahora se trabaja con múltiples superficies. Conforme se procesa cada línea de rastreo, todas las superficies poligonales que cortan esa línea se examinan a fin de determinar cuáles son visibles. En cada posición situada a lo largo de una línea de rastreo, se hacen cálculos de profundidad en cada superficie para determinar cuál es la más próxima al plano de visión. Cuando se ha determinado la superficie visible, el valor de intensidad de esa posición se mete en el buffer de renovación.

La representación de las superficies poligonales de una escena tridimensional puede formarse para incluir una tabla de aristas y una de polígonos. La tabla de aristas contiene extremos coordenados de cada línea de la escena, la pendiente inversa de cada línea y apuntadores en la tabla de polígonos, para identificar las superficies limitadas por cada línea. La tabla de polígonos contiene coeficientes de la ecuación del plano para cada superficie, información de intensidad de las superficies y posibles apuntadores en la tabla de aristas.

Para facilitar la búsqueda de superficies que atraviesan una línea de rastreo dada, podemos preparar una lista activa de aristas a partir de la información contenida en la tabla de aristas. Esta lista activa contendrá solamente aristas que atraviesen la línea de rastreo actual, ordenada en sentido de **x** creciente. Además, se define una señal para cada superficie que se hace on u off para indicar si una posición a lo largo de una línea de rastreo está dentro o fuera de la superficie. Las líneas de rastreo se procesan de izquierda a derecha. En la frontera de más a la izquierda de la superficie, la señal de la superficie se

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

activa y en la frontera de más a la derecha, la señal se desactiva.

La Figura 6.15 ilustra el método de la línea de rastreo para localizar porciones visibles de superficies situadas en una línea de rastreo.

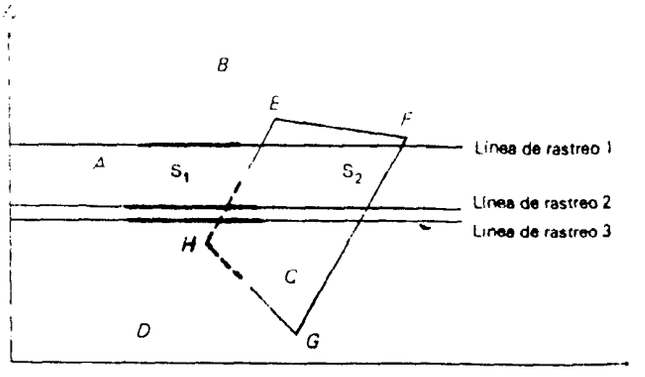


Figura 6.15: Líneas de rastreo que atraviesan la proyección de las superficies S_1 y S_2, en el plano de visión. Las líneas punteadas indican las fronteras de superficies ocultas.

La lista activa de la línea de rastreo 1 contiene información de la tabla de aristas de las aristas **AB**, **BC**, **HE** y **EF**. Para las posiciones en esta línea de rastreo que se encuentran entre las aristas **AB** y **BC**, sólo está activada la señal de la superficie S_1. Por lo tanto, no se necesitan cálculos de profundidad y la información de la intensidad de la superfice S_1 se introduce desde la tabla de polígonos en el buffer de renovación. Análogamente, entre las aristas **HE** y **EF**, sólo está activada la señal de la superficie S_1. Ninguna otra posición contenida en la línea de rastreo 1 corta superficies, de manera que los valores de intensidad de las otras áreas se hacen igual a la intesidad del fondo. La intensidad del fondo puede cargarse en todo el buffer en una rutina de inicialización.

La lista activa de las líneas de rastreo 2 y 3 de la Figura 6.15 contiene las aristas **DA**, **HE**, **BC** y **FG**. En la línea de rastreo 2 de la arista **DA** a la arista **HE**, sólo está activada la señal de la superficie S_1. Pero entre las aristas **HE** y **BC**, están activadas las señales de ambas superficies. En este intervalo, los cálculos de la profundidad deben hacerse con los coeficientes del plano de las dos superficies. En este ejemplo, la profundidad de la superficie S_1 se supone menor que la de S_2, de modo que las intensidades de la superficie S_1 se cargan en el buffer de renovación hasta que se encuentra la frontera **BC**. Después se desactiva la señal de la superficie S_1 y las intensidades de la superficie S_2 se almacenan hasta que se pasa la arista **FG**.

Con este método de la línea de rastreo puede procesarse cualquier número de superficies poligonales en superposición.. Se colocan las señales de las superficies para indicar si una posición está dentro o fuera, y se realizan cálculos de profundidad cuando las superficies se superponen.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

6.8 METODO DE ORDENAMIENTO CON PROFUNDIDAD.

El método de ordenamiento con profundidad cumple con las siguientes funciones básicas:

1. Las superficies se ordenan en sentido de profundidad decreciente.

2. Las superficies se convierten con rastreo en orden, comenzando con las superficies de máxima profundidad.

Las operaciones de ordenamiento se efectúan en objeto-espacio, y la conversión con rastreo de las superficies poligonales se realizan en imagen-espacio.

Este método para resolver el problema de la superficie oculta, algunas veces se conoce como el **algoritmo del pintor**. Un artista, al crear una pintura al óleo, primero pinta los colores de fondo. Después, se suman los objetos más distantes. Por último, se pintan los objetos de primer plano en el lienzo sobre los objetos más distantes. Cada capa de pintura cubre la capa anterior. Mediante una técnica análoga, primero ordenamos las superficies según su distancia desde el plano de visión. Los valores de intensidad de la superficie más alejada se introducen después en el buffer de renovación. Tomando cada superficie sucesiva por turno (en orden de profundidad decreciente), "pintamos" las intensidades de la superficie en el buffer de estructura sobre las intensidades de las superficies que se procesaron anteriormente.

El trazo de superficies poligonales sobre el buffer de estructura según la profundidad, se lleva a cabo en varias etapas. En el primer paso, se ordenan las superficies de acuerdo con el mayor valor de **z** en cada superficie. La superficie con la máxima profundidad (llámese **S**) se compara después con las otras superficies de la lista para determinar si hay superposición de profundidad. Si no ocurre ninguna superposición de profundidad, **S** se convierte con rastreo. La figura 6.16 muestra dos superficies sin superposición de profundidad, que se han proyectado en el plano **xz**.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

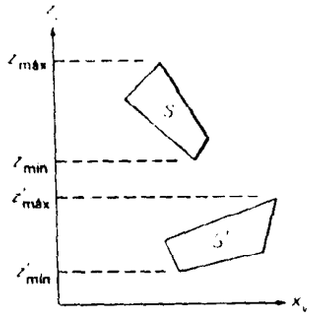


Figura 6.16: Dos superficies sin superposición de profundidad.

Este proceso se repite después con la siguiente superficie de la lista. Mientras no ocurran superposiciones, cada superficie se procesa por orden de profundidad hasta que todas se hayan convertido con rastreo. Si se detecta una superposición de profundidad en cualquier punto de la lista, necesitamos hacer algunas otras comparaciones para

determinar si alguna de las superficies debe reordenarse.

Para cada superficie que se superpone con **S**, se hacen las siguientes pruebas. Si alguna de las pruebas se cumple, no se necesita reordenamiento de esa superficie. Las pruebas se relacionan en orden de dificultad creciente.

1. Los rectángulos limitantes en el plano **xy** de las dos superficies no se superponen.

2. La superficie **S** está en la "parte exterior" de la superficie de superposición, relativa al plano de visión.

3. La superficie de superposición está en la "parte interior" de la superficie **S**, relativa al plano de visión.

4. Las proyecciones de las dos superficies sobre el plano de visión no se superponen.

Tan pronto se determina que se cumple alguna de las pruebas en relación con una superficie de superposición, se sabe que esa superficie no está detrás de **S**. Así que pasamos a la siguiente superficie. Si todas las superficies de superposición pasan por lo menos una de estas pruebas, no se necesita reordenamiento y **S** puede convertirse con rastreo.

La prueba 1 se efectúa en dos partes. Primero se verifica si hay superposición en la dirección **x**, luego en la dirección **y**. Si alguna de estas direcciones no muestra superposición, los dos planos no pueden oscurecerse entre sí. Un ejemplo de dos superficies que se superponen en la dirección **z** pero no en la dirección **x**, se muestra en la Figura 6.17.

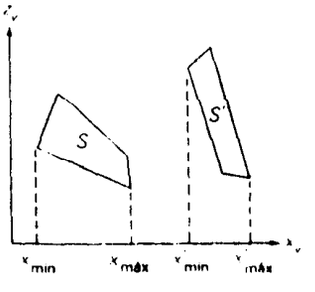


Figura 6.17: Dos superficies con superposición de profundidad pero sin superposición en la dirección **x**.

Podemos realizar la prueba 2 sustituyendo las coordenadas de todos los vértices de **S** en la ecuación del plano para la superficie de superposición y comprobando el signo del resultado. Supóngase que la superficie de superposición tiene los coeficientes del plano **A'**, **B'**, **C'** y **D'**. Si

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

$$A'x + B'y + C'z + D' > 0$$

para cada vértice de S; entonces, la superficie **S** está "fuera" de la superficie de superposición (Figura 6.18).

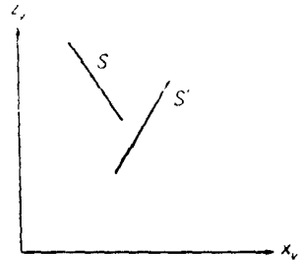


Figura 6.18: La superficie **S** está completamente "fuera de" la superficie de superposición **S'** relativa al plano de visión.

La prueba 3 se lleva a cabo con los coeficientes del plano **A**, **B**, **C** y **D** de la superficie **S**. Si las coordenadas de todos los vértices de la superficie de superposición satisfacen la condición $Ax + By + Cz + D < 0$, la superficie de superposición está "dentro de" la superficie **S** (siempre que la normal de la superficie **S** apunte lejos del plano de visión). La Figura 6.19 muestra una superficie de superposición **S'** que cumple esta prueba. En este ejemplo, la superficie **S** no está "fuera de" **S'** (la prueba 2 no se cumple).

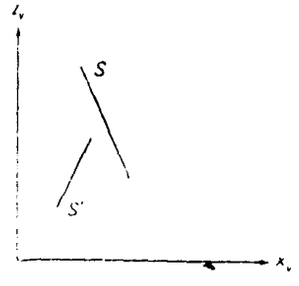


Figura 6.19: La superficie de superposición **S'** está completamente "dentro de" la superficie **S**, relativa al plano de visión.

Si las pruebas 1 a 3 no se han cumplido, se intenta la prueba 4 verificando las intersecciones entre las aristas limítrofes de las dos superficies, utilizando ecuaciones de rectas en el plano **xy**. Dos superficies pueden o no cortarse, aunque sus volúmenes limítrofes se superpongan en las direcciones **X**, **Y** y **Z**.

Si las cuatro pruebas llegan a fallar con una superficie de superposición **S'** determinada, se intercambian las superficies **S** y **S'**. Un ejemplo de dos superficies que se reordenarían con este procedimiento se da en la Figura 6.20. Sin embargo, todavía no sabemos con certeza que ya se encontró la superficie más alejada del plano de visión.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

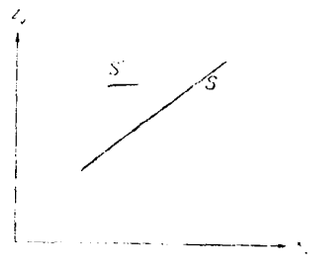


Figura 6.20: La superficie **S** tiene mayor profundidad pero oscurece la superficie **S'**.

La Figura 6.21 ilustra una situación en la cual primero se intercambiarían **S** y **S''**. Pero ya que **S''** oscurece una parte de **S'**, es necesario intercambiar **S''** y **S'** para colocar las tres superficies en el orden de profundidad correcto. Por lo tanto, es necesario repetir el proceso de comprobación con cada superficie que se reordene en la lista.

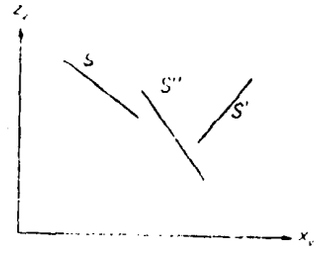


Figura 6.21: Tres superficies metidas en una lista de superficies ordenada en el orden **S, S', S''** debe reordenarse **S', S'', S.**

Es posible que el algoritmo recién descrito entre en un ciclo infinito si dos o más superficies se oscurecen alternativamente entre sí, como en la figura 6.22. En tales situaciones, el algoritmo reorganizaría continuamente las posiciones de las superficies de superposición. Para evitar ciclos, podemos señalar alguna superficie que se haya reordenado en una posición de profundidad más lejana, de manera que no pueda volver a moverse. Si se hace un intento de cambiar la superficie por segunda vez, ésta se divide en dos partes en la línea de intersección de los dos planos que se someten a comparación. La superficie original se sustituye entonces por las dos nuevas superficies y se continúa el proceso como antes.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

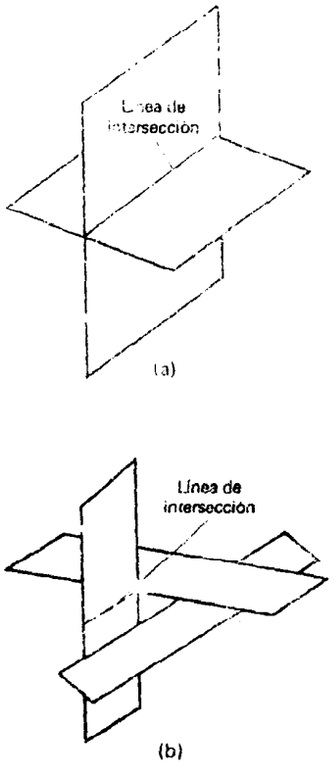


Figura 6.22: Ejemplos de orientaciones de superficies que se oscurecen alternativamente entre sí.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

6.9 METODO DE SUBDIVISION DE AREAS (WARNOCK).

Esta técnica de supresión de superficies ocultas es, esencialmente, un método de imagen-espacio, pero pueden utilizarse operaciones de objeto-espacio para realizar el ordenamiento por profundidad de las superficies. El método de subdivisión de áreas se aprovecha del área de una escena, localizando aquellas áreas de visión que representan parte de una sola superficie. Este método se aplica dividiendo sucesivamente el área de visión total en rectángulos cada vez de menor tamaño, hasta que cada área pequeña sea la proyección de parte de una superficie visible o bien no sea parte de ninguna superficie.

Para implantar este método, se necesitan establecer pruebas que puedan identificar rápidamente el área como parte de una superficie o bien que puedan indicarnos que el área es demasiado compleja para analizarse con facilidad. Comenzando con la vista total, se aplican las pruebas para determinar si se debe subdividir el área total en rectángulos de menor tamaño. Si las pruebas indican que la vista es, suficientemente compleja, ésta se subdivide. Después, se aplican a cada una de las áreas de menor tamaño, subdividiéndose éstas si las pruebas indican que la visibilidad de una superficie sigue siendo incierta. Se continúa este proceso hasta que las subdivisiones se analizan con facilidad y se determina que pertenecen a una superficie o bien hasta que se reduzcan al tamaño de un pixel. Una manera de subdividir un área, consiste en dividir sucesivamente las dimensiones del área

entre 2, como se muestra en la Figura 6.23.

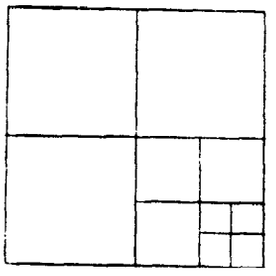


Figura 6.23: Subdivisiones de áreas realizadas con divisiones sucesivas entre 2.

Se hacen pruebas para determinar la visibilidad de una superficie dentro de un área especificada, comparando superficies con la frontera del área. Existen cuatro relaciones posibles que puede tener una superficie con una frontera de área especificada. Podemos describir estas características relativas a las superficies, en la forma siguiente (Figura 6.24):

Una **superficie circundante** es aquella que encierra completamente el área.

Una **superficie de superposición** es aquella que está parcialmente en el interior o en el exterior del área.

Una **superficie interior** es aquella que está completamente dentro del área.

Una **superficie exterior** es aquella que está completamente fuera del área.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

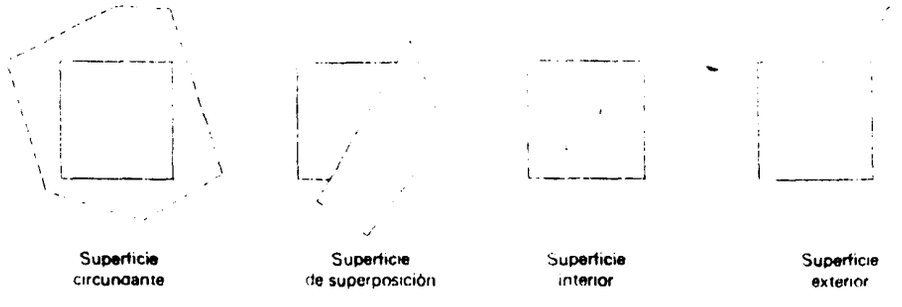


Figura 6.24: Posibles relaciones entre superficies poligonales y un área rectangular.

Las pruebas para determinar la visibilidad de una superficie contenida en un área pueden expresarse en términos de estas cuatro clasificaciones. Si se cumple alguna de las siguientes condiciones, no se necesitan subdivisiones adicionales de un área especificada.

1. Todas las superficies están fuera del área.

2. Sólo una superficie interior, de superposición o circundante está en el área.

3. Una superficie circundante oscurece todas las otras superficies dentro de las fronteras del área.

La prueba 1 puede efectuarse verificando los rectángulos limítrofes de todas las superficies contra las fronteras del área. La prueba 2 también puede utilizar los rectángulos limítrofes en el plano **xy** para identificar una superficie interior. Para otro tipo de superficies, pueden utilizarse los rectángulos limítrofes como comprobación inicial. Si un solo rectángulo limítrofe corta el área en alguna forma, se recurre a verificaciones adicionales para determinar si la superficie es circundante, de superposición o exterior. Una vez que se ha identificado una superficie interior, de superposición o circundante, sus intensidades de pixeles se transfieren al área indicada dentro del buffer de estructura.

Un método para realizar la prueba 3 consiste en ordenar las superficies según su profundidad mínima. Para cada superficie circundante, calculamos entonces el valor máximo de **z** dentro del área en cosideración. Si el valor máximo de **z** de una de estas superficies circundantes es menor que los valores calculados de **z** de todas las otras superficies, se cumple la prueba 3. Entonces el área puede llenarse con valores de intensidad de la superficie circundante.

En algunos casos, los dos métodos para verificar la prueba 3, no podrán identificar correctamente una superficie circundante que oscurezca todas las otras superficies. Podrían realizarse más pruebas para identificar la superficie que cubre el área, pero es más rápido subdividir el área que continuar con pruebas más complejas.

Identificadas ya, las superficies externas y circundantes de un área, éstas seguirán siendo superficies exteriores y circundantes para todas las subdivisiones del área. Además, algunas superficies interiores y de superposición pueden llegar a eliminarse conforme continúe el proceso de subdivisión, de manera que las áreas se vuelvan más fáciles de analizar. En el caso límite, cuando se produce una subdivisión del tamaño de un pixel, simplemente se calcula la profundidad de cada superficie relevante en ese punto y se transfiere la intensidad de la superficie más cercana al buffer de estructura.

Una variación en el proceso básico de subdivisión consiste en subdividir áreas a lo largo de las fronteras de superficies en vez de dividirlas a la mitad. Si las superficies se han ordenado según la profundidad mínima, podemos utilizar la superficie con el mínimo valor de **z** para subdividir un área determinada. La figura 6.25 ilustra este método de subdivisión de áreas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

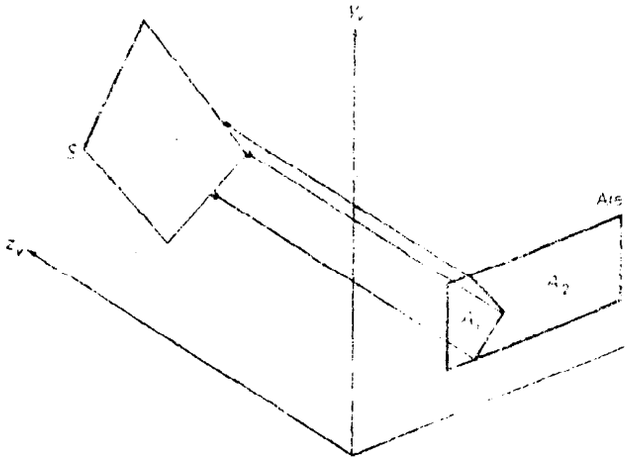


Figura 6.25: El área **A** se subdivide en A_1 y A_2 usando la frontera de la superficie **S** sobre el plano de visión.

La proyección de la frontera de la superficie **S** se utiliza para particionar el área original en las subdivisiones A_1 y A_2. La superficie **S** es, por tanto, una superficie circundante de A_1 y las pruebas de visibilidad 2 y 3 pueden aplicarse para determinar si se necesita una mayor subdivisión. En general, se requieren menos subdivisiones utilizando este método, pero se necesita más procesamiento para subdividir áreas y para analizar la relación de las superficies con las fronteras de subdivisión.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

CAPITULO 7

CAD/CAM

7.1 INTRODUCCION AL CAD/CAM

CAD/CAM es un término que significa:

CAD: Diseño Asistido por Computador.

CAM: Fabricación Asistida por Computador.

Es una tecnología que está ligada con el uso de los computadores para realizar ciertas funciones en el Diseño y la Producción.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

7.1.1. C.A.D.

El CAD puede definirse como el uso del computador como ayuda en la **Creación, Modificación, Análisis** u **Optimización** de un diseño.

El hardware del CAD incluye generalmente los siguientes elementos:

1.- Una o más Estaciones de Trabajo (Workstation):

* Un Terminal Gráfico

. Pantallas de Refresco

Pantallas Vectoriales

Pantallas de Barrido

. Pantallas de Almacenamiento

* Dispositivos de Entrada

. Tableta Gráfica

. Joystick

. Ratón

. Lapiz Optico

. Trackball

2.- Uno o más Plotters y otros dispositivos de salida

* Plotter Plano

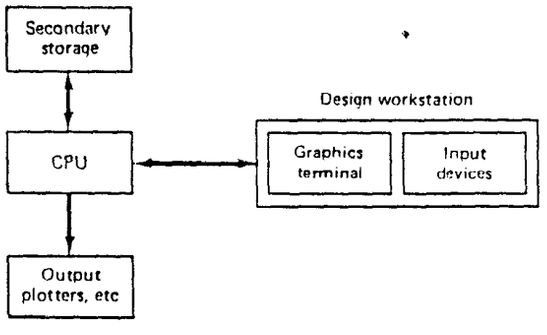
* Plotter de Tambor

* Plotter de Fricción

* Impresoras Gráficas

3.- Unidad Central de Proceso

4.- Almacenamiento Secundario



El Software consiste de:

1.- Programas para implementar gráficos en el sistema.

2.- Programas de aplicaciones específicas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

7.1.2. C.A.M.

El CAM puede definirse como el uso del computador para **Planificar**, **Manejar** y **Controlar** las operaciones de una planta de fabricación mediante la interfase del computador (directa o indirecta) con los recursos de producción de dicha planta.

Las distintas aplicaciones del CAM caen dentro de dos categorías:

* Computer Monitoring and Control

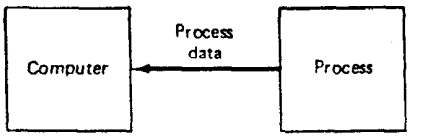
* Manufacturing Support Applications

1.- Computer Monitoring and Control

Son las aplicaciones Directas en las cuales el computador está conectado directamente al proceso de fabricación para monitorizar y controlar el proceso.

1a.- Monitorización del Proceso por Computador.

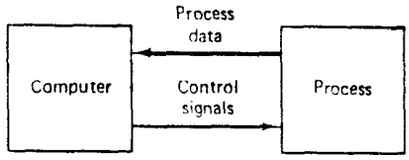
Implica una interfase directa entre el computador y el proceso de fabricación para observar y obtener datos del proceso.



1b.- Control del Proceso por Computador

Implica una interfase directa entre el computador y el proceso de fabricación para observar el proceso y controlarlo basándose en dichas observaciones.

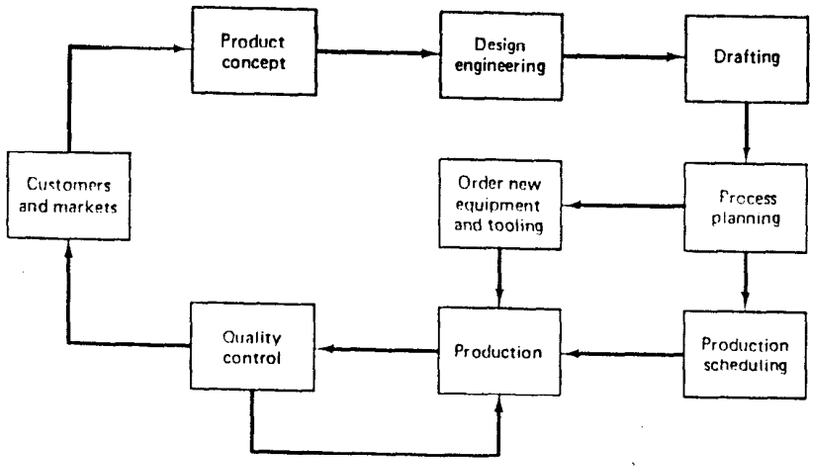
© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



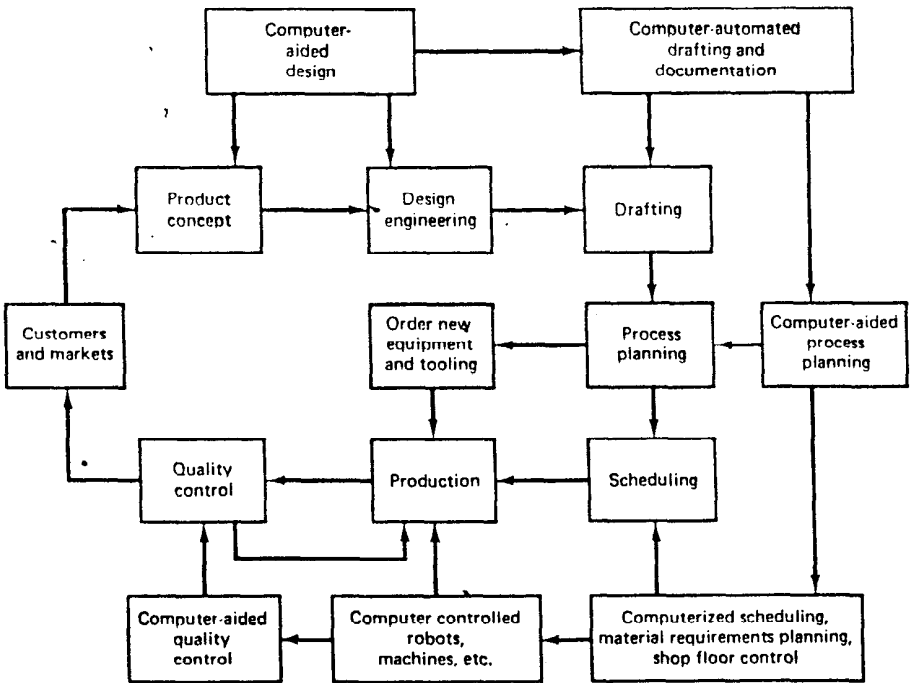
2.- Manufacturing Support Aplications

Son las aplicaciones Indirectas en las cuales el computador se utiliza para soportar las operaciones de producción de la planta.

Ciclo del Producto y CAD/CAM



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

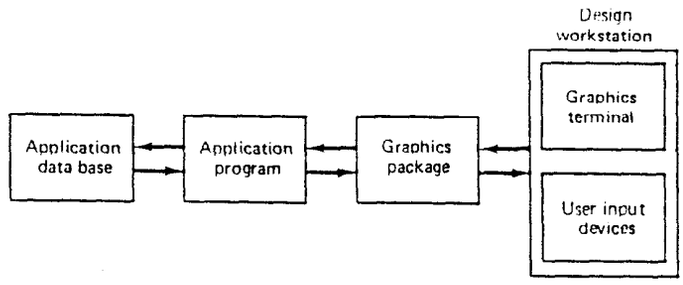


7.2 SISTEMA C A D.

7.2.1. Introducción

Los sistemas CAD están basados en los Gráficos por Computador Interactivos. Estos aportan un sistema orientado al usuario en el cual el computador se emplea para **Crear**, **Transformar** y **Mostrar** datos en forma de imágenes o símbolos. El usuario en el sistema de diseño de gráficos por computador es el Diseñador, el cual comunica datos y comandos al computador mediante distintos dispositivos de Entrada. Este se comunica con el usuario mediante el CRT. La configuración del software se muestra en la siguiente figura:

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



Las razones fundamentales para implementar un sistema CAD son:

1. Aumentar la Productividad del Diseñador:

Esto se realiza ayudando al diseñador a visualizar el producto y sus componentes de una forma total o parcial, reduciendo el tiempo requerido en la Síntesis, Análisis y Documentación del diseño.

2. Majorar la Calidad del Diseño:

Un sistema CAD permite un mayor análisis y puede investigarse un mayor número de alternativas del diseño. También se reduce los errores de diseño por una mayor exactitud suministrada por el sistema.

3. Mejorar la Comunicación:

El uso de un sistema CAD suministra mejores dibujos, mayor estandarización en los dibujos, mejor documentación del diseño, menor errores de dibujo y mayor legibilidad.

4. Creación de una Base de Datos para la Fabricación:

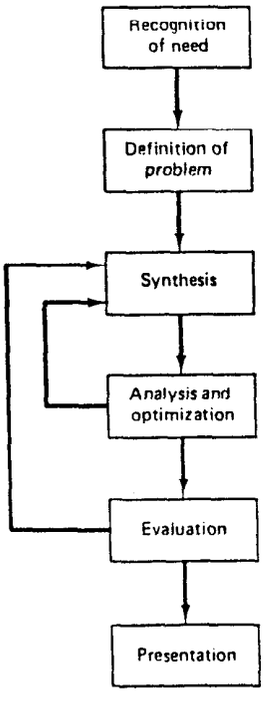
En el proceso de creación de la documentación para el diseño del producto (geometría y dimensiones del producto y sus componentes, especificaciones del material para las componentes, etc) también se crea una base de datos necesitada para la fabricación del producto.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

7.2.2. Proceso de Diseño

El proceso de diseñar algo se puede caracterizar como un procedimiento interactivo, el cual consta de 6 fases:

1. Reconocimiento de la necesidad
2. Definición del problema
3. Síntesis
4. Análisis y Optimización
5. Evaluación
6. Presentación



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

7.2.3. Aplicaciones de los Computadores para el Diseño

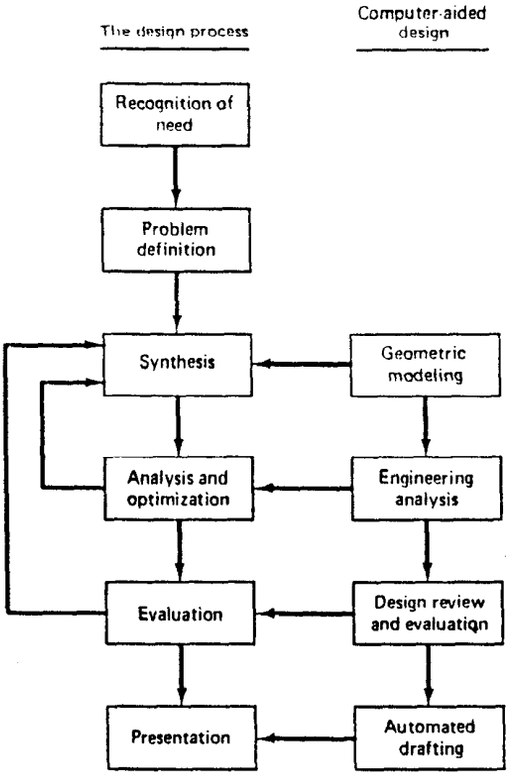
Las distintas fases vistas anteriormente son realizadas por un sistema CAD agrupadas en 4 áreas funcionales:

A. Modelado Geométrico

B. Análisis de Ingeniería

C. Revisión del Diseño y Evaluación

D. Dibujo Automatizado



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

A. Modelado Geométrico

En CAD, el Modelado Geométrico tiene que ver con la descripción matemática compatible con el computador de la geometría de un objeto. La descripción matemática permite la representación y manipulación de la imagen de un objeto sobre un terminal gráfico mediante señales desde el sistema CAD a la CPU.

Con el uso del modelado geométrico, el diseñador construye la imagen gráfica del objeto sobre la pantalla del sistema gráfico entrando 3 tipos de comandos al computador:

a) El primer tipo genera elementos geométricos básicos tales como puntos, líneas y círculos.

b) El segundo tipo se utiliza para realizar transformaciones geométricas tales como rotación, escalado y traslación.

c) El tercer tipo produce que se unan varios elementos y se muestren sobre el sistema de gráficos por computador.

Durante el proceso de modelado geométrico, el computador convierte los comandos en un modelo geométrico, lo almacena en un fichero de datos en el computador y muestra la imagen en la pantalla.

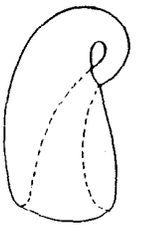
Existen varios métodos de representar el objeto en modelado geométrico.

En primer lugar, el sistema debe ser capaz de distinguir los sólidos geométricamente correctos de aquellos que no lo son, y sólo debe permitir almacenar los primeros. En concreto, se define como sólido representable el que cumple las siguientes restricciones:

1.- Debe ser rígido: su forma ha de ser independiente de la posición espacial y orientación.

2.- Debe ser finito, ocupando una porción finita del espacio.

3.- Su superficie externa debe ser cerrada y orientable: ha de determinar, sin ambiguedad, que parte es interior al sólido y que parte es exterior.



4.- Debe cumplir la ecuación de Euler:

$$C + V - A = 2 + R - 2H$$

C: Número de caras

V: Número de vértices

A: Número de aristas

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

R: Número de anillos interiores a las caras

H: Número de agujeros ue atraviesan el cuerpo

5.- Toda operación geométrica o booleana entre sólidos representables debe producir como resultado otros sólidos representables.

Para poder comparar entre sí los distintos métodos que se utilizan, se definen los siguientes parámetros:

1.- **Dominio**: Cuanto mayor es el dominio en un sistema de representación de sólidos, mayor es el número de objetos reales que son representables.

2.- **Ambiguedad**: Un sistema es ambiguo si un modelo representado en él corresponde a más de un sólido real.

3.- **Grado de Unicidad**: es deseable que todo sólido real tenga un único modelo de representación interna, para poder detectar y evitar posibles duplicaciones de la base de datos.

4.- **Validez**: Las representaciones internas deben corresponder a sólidos correctos.

5.- **Ocupación de Memoria**: La cantidad de memoria necesaria para almacenar un determinado sólido real debe ser reducida.

6.- **Facilidad de Creación de Objetos**: Las herramientas para la creación de diseños por parte del usuario deben ser simples.

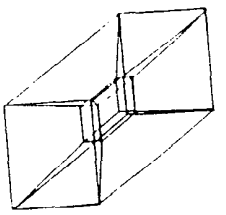
No existe en estos momentos ningún sistema de representación de sólidos óptimo con respecto a todos estos parámetros.

EL MODELO DE ALAMBRES

Es uno de los sistemas más sencillo para almacenar información tridimensional. En él, el computador dispone de las coordenadas en el espacio de todos los vértices del cuerpo, junto con la información de que pares de vértices se encuentran unidos mediante aristas. Mediantes sencillas transformaciones geométricas de proyección, se puede obtener cualquier vista del objeto. Sin embargo, este modelo de alambres tiene el grave inconveniente de ser ambiguo y de no permitir la producción de secciones y vistas con eliminación de partes ocultas. Por todo ello, es poco utilizado en sistemas avanzados de

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

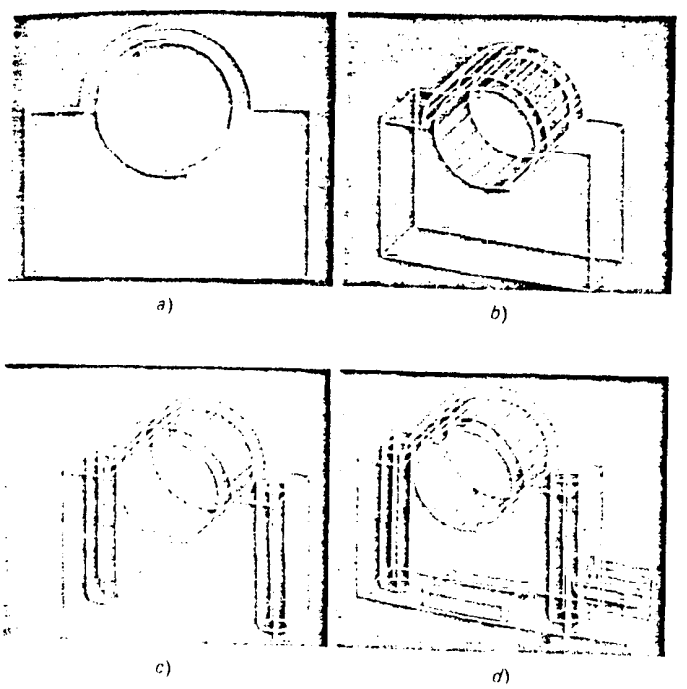
diseño. Como muestra de la ambiguedad de este sistema, la siguiente figura refleja la ambiguedad de este sistema.



EL MODELO DE FRONTERAS

Es uno de los sistemas de representación más utilizado en la actualidad. En este sistema, lo único que hace es ampliar la información que se almacenaba en el modelo de alambres, incluyendo datos de los polígonos y de las caras del objeto. Contiene toda la información tridimensional, es no ambiguo, es completo y posibilita todo tipo de operaciones y representaciones realista del sólido.

Una de las técnicas más utilizadas en los sistemas de modelado geométrico para la creación de nuevos sólidos es el llamado Método de Barrido. Con este sistema, el usuario genera el objeto tridimensional mediante traslaciones o rotaciones de caras planas que dibuja en patalla.



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

La fase final de todo proceso de modelado, una vez diseñada interactivamente la forma del sólido y modificada con las operaciones booleanas y de sección, debe generar salidas numéricas y gráficas que permitan el análisis de sus características y la posible fabricación de un prototipo. Entre este tipo de salidas se encuentran:

1.- Las propiedades volumétricas del mismo: volumen, masa, momentos y productos de inercia.

2.- La representación del objeto mediante proyecciones bidimensionales.

3.- La conexión del objeto con otros elementos de una base de datos.

4.- La generación de las cintas de Control Numérico, para la mecanización automática del objeto diseñado.

B. Análisis de Ingeniería

En la formulación de cualquier proyecto de diseño de ingeniería se requiere algún tipo de análisis, utilizando el computador como ayuda. Generalmente, es necesario que se desarrolle programas específicos para solventar un problema de diseño particular. En otras aplicaciones puede utilizarse programas de propósito general disponibles en el mercado.

Veamos dos ejemplos importantes de este tipo:

- Análisis de Propiedades de Masa.
- Análisis de Elementos Finitos.

El Análisis de Propiedades de Masa es un análisis característico de un sistema CAD y probablemente el de mayor aplicación. Proporciona las propiedades de un objeto sólido que se esté analizando, tales como el área de la superficie, peso, volumen, centro de gravedad y momento de inercia. Para una superficie plana, la computación correspondiente incluye el perímetro, área y propiedades de inercia.

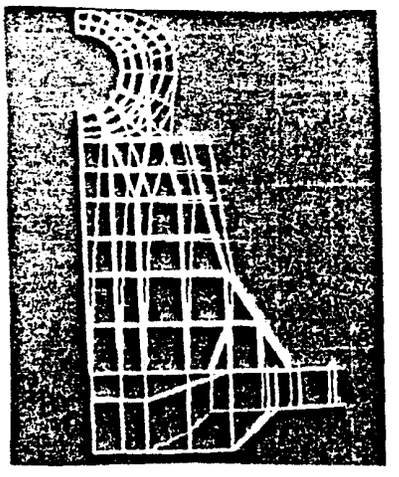
Probablemente el análisis más característico es el Método de Elementos Finitos. Con esta técnica, el objeto se divide en un gran número de elementos finitos (generalmente rectangular o triangular) los cuales forman una red de interconexión de nodos.

Algunos sistemas CAD tienen la capacidad de definir automáticamente los nodos y la estructura de red de un objeto dado. El usuario simplemente define ciertos parámetros para el modelado de elementos finitos y el sistema CAD procede con la computación.

La salida del análisis de elementos finitos es presentado generalmente por el sistema

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

en formato gráfico en el CRT.

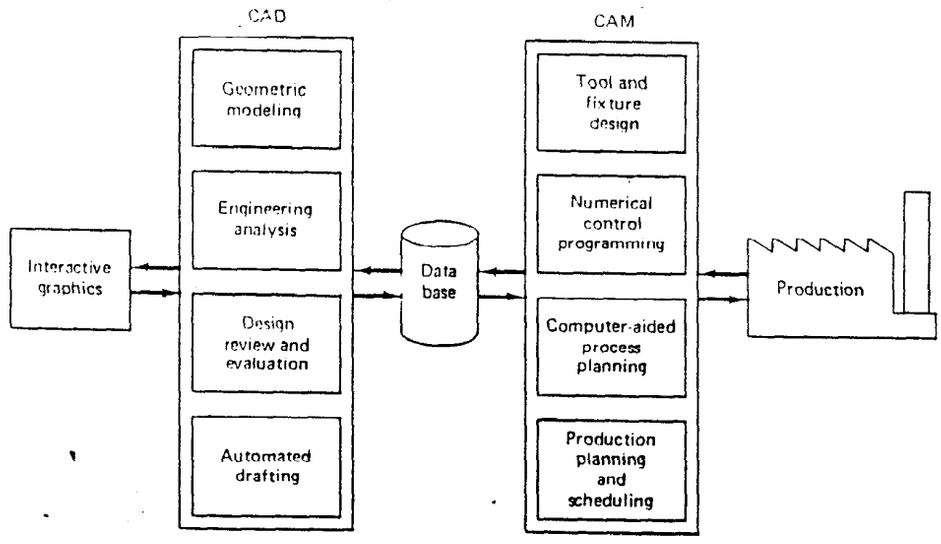


7.2.4. Creación de una Base de Datos para la Fabricación

Una de las razones importantes de la utilización de un sistema CAD es que ofrece la oportunidad de desarrollar una base de datos necesaria para la fabricación del producto. En un sistema CAD/CAM integrado, se establece una conexión directa entre el diseño y la fabricación del producto. La función del CAD/CAM no sólo es automatizar ciertas fases del diseño y ciertas fases de fabricación, sino también la transición desde el diseño a la fabricación.

La Base de Datos de la fabricación es una base de datos de CAD/CAM integrada. Incluye todos los datos del producto generado durante el diseño (datos geométricos, lista de materiales, especificaciones del material, etc.) así como datos adicionales requerido para la fabricación, muchos de los cuales están basados en el diseño del producto. La siguiente figura muestra como la base de datos CAD/CAM está relacionada con el diseño y la fabricación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



7.3 SISTEMA C. A. M.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

7.3.1. Relaciones entre CAD y CAM. Conceptos de CIM

El CAD y el CAM constituyen dos técnicas que, aunque diferentes, han estado estrechamente relacionadas desde su aparición. Sin embargo, su evolución no ha logrado ser lo suficientemente convergente para que la comunicación entre ambos procesos alcance los niveles mínimos deseables.

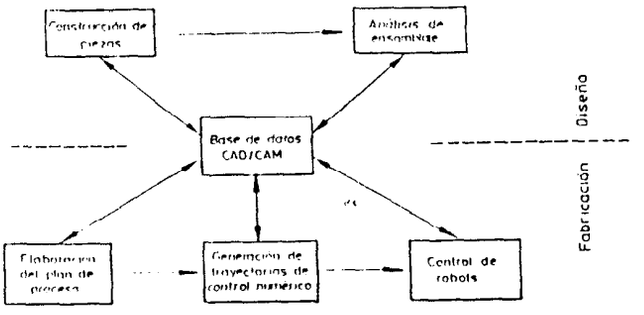
El CAD ha pasado de ser un simple sistema automatizado de dibujo a ser un sistema de análisis y construcción de sólidos. Por su parte, el CAM incluye actualmente aspectos de planificación así como de control de la máquina-herramienta y de los equipos de alimentación del material transformado por aquella (Computer Aided Process Planning, CAPP).

Comunicación CAD-CAM

Las Bases de Datos generadas por sistemas de CAD contienen, esencialmente, información previamente entrada por el usuario: geometría y topología de las piezas más ciertos requisitos de tolerancia y acabado de las superficies.

Al no contener la base de datos el proceso de fabricación, un sistema de CAM no puede utilizarla directamente siéndole necesario, para generar automáticamente el plan de fabricación, extraer de la base de datos aspectos de fabricación o conocimiento semántico acerca de las piezas.

La siguiente figura muestra diversos módulos de un sistema de CAD/CAM



El buen resultado del proceso efectuado por cada uno de ellos y, por tanto del sistema, presupone que el conocimiento semántico necesario existe o puede extraerse de la base de datos, por ejemplo:

1.- El módulo de elaboración del plan de proceso necesita tener acceso a la descripción de la pieza en términos de relación entre la geometría de la pieza y las operaciones de fabricación disponibles.

2.- El módulo de generación de trayectorias de control numérico necesita un conocimiento semántico acerca del entorno del punto mecanizado a fin de prevenir colisiones.

Cualquier sistema integrado de CAD/CAM tiene que abordar esta necesidad de conocimiento semántico y, por tanto, ser capaz de interpretar la base de datos y obtener la información necesaria.

Concepto de CIM

La Fabricación Integrada por Computador (CIM) tiende a unificar la comunicación, no únicamente entre CAD y CAM, sino también con otros medios de producción, como puede ser almacenes, transporte interno o robots.

Un robot integrado en una célula de fabricación puede estar dedicado a procesos tales como soldadura o pintura, que pueden programarse adecuadamente mediante técnicas no textuales o textuales a nivel manipulador, existente en la actualidad.

Para procesos de mecanizado y ensamblaje son deseables técnicas de programación a nivel de tareas basadas en modelos geométricos que, entre otras cosas, permitan en

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

tiempo de programación detectar coliciones entre mano, pieza o herramienta y el entorno dentro del que se está operando.

El proceso de ensamblaje puede verse como la creación de un nuevo objeto compuesto a partir de otras existentes inicialmente.

Esto implica establecer una base de montaje y decidir cuál será la primera pieza a manipular y sobre la que se irán ensamblando las restantes piezas componentes del objeto final.

Deberá determinarse el orden en que tienen que ser montadas las piezas y cuáles son las condiciones de acoplamiento entre ellas, dos a dos o en contacto múltiple.

Habrá que tenerse en cuenta la estabilidad del conjunto, ya que no es lo mismo realizar el montaje sobre una cinta transportadora que hacerlo sobre una mesa manteniendo fija la base del ensamblaje. En el primer caso la disposición por gravedad será más crítica que el segundo, y cualquier movimiento para el acoplamiento de dos piezas sobre una cinta será más complejo de formular, dada la necesidad de tener en cuenta el movimiento de la base en función del tiempo.

Los principales problemas a resolver en una operación de ensamblaje son:

1.- Determinación de Trayectorias exentas de riesgo de colisión.

Dado un objeto con una posición inicial, una posición final y un conjunto de obstáculos de situación conocida, se trata de encontrar una trayectoria contínua que lleve el objeto de la posición inicial a la final sin colicionar con los obstáculos.

2.- Presión

El problema de presión consiste en determinar cómo coger un objeto con la mano del robot. Se incluyen aquí, además de un problema de trayectoria, un conjunto de condicionantes relacionados con el tipo de garra que será más apropiada para cada caso y la orientación en que ésta deberá trabajar, así como la posibilidad para el robot de tomar configuraciones adecuadas.

3.- Acoplamiento de Piezas

En ausencia de incertidumbre este problema no consistiría más que en una simple secuencia de movimientos encaminada a establecer entre las piezas la relación geométrica prefijada. En la práctica, acoplar dos piezas presupone seguir una estrategia que reduzca la incertidumbre y permita alcanzar la relación geométrica. Ello hace necesario un estudio previo de las ligaduras que existen entre las dos piezas, que conduzca a determinar cuáles serán los indicadores del grado de acoplamiento.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

BIBLIOTECA GENERAL

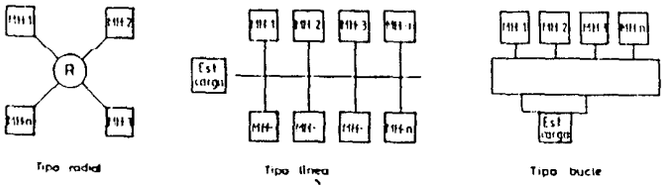
Introducción

Los Sistemas Flexibles de Producción (FMS) consiguen en gran medida trabajar en condiciones próximas a las consideredas como ideales. El concepto de FMS fue introducido por Dolezalek en 1967 y los primeros sistemas flexibles de fabricación se pusieron en funcionamiento al final de los años sesenta. Fueron una consecuencia lógica del enorme desarrollo experimentado por las técnicas de control numérico y los mini/microcomputadores. Aunque no existe acuerdo unánime sobre la definición de sistemas flexibles de producción, una de las más difundidas es la siguiente:

> "Se denomina Sistema Flexible de Producción a uno formado por máquinas e instalaciones técnicas, enlazadas entre sí por un sistema común de transporte y control, de forma que exista la posibilidad, en un determinado margen, de realizar tareas diversas correspondientes a piezas diferentes sin necesidad de interrumpir el proceso de fabricación para el reequipamiento del conjunto".

En la actualidad puede decirse que los diversos FMS existentes han sido concebidos y diseñados para realizar tareas de fabricación bastante concretas localizadas sobre objetivos que, en gran medida, dependen de la configuración geométrica, tamaño y volumen de la producción de las piezas a ser mecanizadas. Tal circunstancia ha dado lugar a la aparición de una gran variedad de sistemas, lo que representa un grave impedimento para establecer una clasificación clara y sistemática de los mismos. Sin embargo, desde un punto de vista inmediato, puede establecerse la clasificación siguiente: Sistemas de Configuración Radial, Sistemas de Configuración en Línea y Sistemas de Configuración en Bucle, tal como puede verse en la siguiente figura.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



Sistema de Configuración Radial

En este sistema, las diversas máquinas están instaladas en torno a un equipo de medida y un robot que efectúa las tareas de carga y descarga de piezas, atiende al cambio de herramientas, manipulación etc. Cada máquina dispone de su propio microcomputador y

del robot, lo que permite controlar la secuencia de trabajo establecida, a través de la red informática que se crea al conectar los microcomputadores entre sí. Un sistema de esta clase es adecuado para la producción de piezas muy similares, que requieren variaciones pequeñas en la secuencia de trabajo. Lógicamente, una configuración de este tipo no está pensada para un número grande de máquinas, pero se puede estructurar un conjunto potente por asociación adecuada de un determinado número de sistemas sencillos.

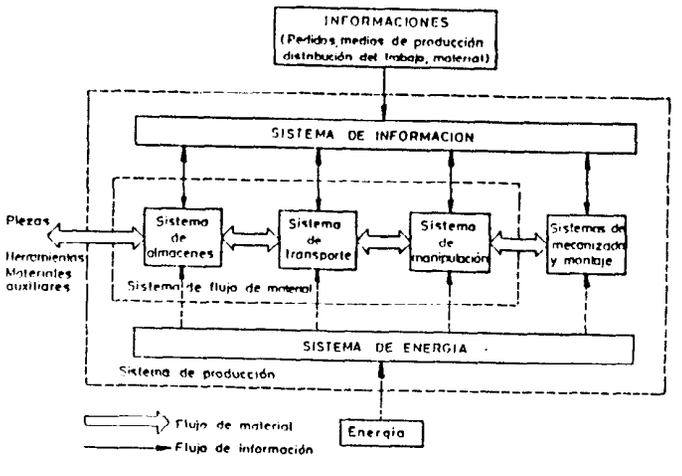
Sistema de Configuración en Línea

En este tipo de sistemas, las máquinas-herramientas están dispuestas a lo largo de una línea de transporte, por la cual son enviadas las piezas a las estaciones de carga y descarga o a los almacenes intermedios, encargados de cubrir las necesidades de cada máquina-herramienta. Esta clase de sistemas ocupan menos espacio y permiten el transporte de piezas pesadas mediante el uso de vehículos guiados (automáticamente o no) de características adecuadas. El transporte utilizado es importante, en el sentido de que podría resultar limitada la flexibilidad y rendimiento del sistema como consecuencia de la elección realizada.

Sistema de Configuración en Bucle

En este tipo de sistemas, con una disposición de trasnporte de uno o varios bucles, se atiende a las estaciones de carga y descarga, los almacenes, las máquinas-herramientas, los almacenes intermedios, etc. Con este tipo de configuración se pueden construir sistemas flexibles de grandes dimensiones, en los que pueden realizarse variadas secuencias de operaciones debido al fácil acceso a cada máquina-herramienta. Se requiere, sin embargo, una planificación de las operaciones muy elaborada para que el sistema actúe eficazmente.

La estructura de los sistemas flexibles de producción puede ser analizada contemplando sus flujos de material y de información en forma específica y analizando, posteriormente, las interconexiones que ligan a ambos. La siguiente figura muestra un sistema flexible de producción con sus correspondientes subsistemas y flujos de material e información:



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Desde un punto de vista técnico, en un FMS se dispone de un grupo de máquinas e instalaciones, controlables por computador, enlazadas entre sí funcionalmente por un flujo de material, y controladas y supervisadas a través de un flujo de información que establece una unión informática entre las mismas.

El flujo de material abarca a todos los procesos de movimiento y almacenado necesarios para realizar la carga y descarga de las diversas estaciones de trabajo, en cuanto se refiere a materiales, piezas, elementos auxiliares para la fabricación y residuos.

El flujo de información sirve para establecer la unión informática, que se materializa a través del tratamiento de la información que proviene de los dispositivos y elementos integrantes del sistema de control de FMS. El enlace técnico-informático se consigue, generalmente mediente sistemas de Control Numérico Directo (DNC) y de control de robots desde computadores externos que, de una manera centralizada, gestionan y distribuyen la información requerida para realizar el control de varias estaciones de trabajo y del flujo de material, así como la captación y elaboración de datos específicos tomados del propio proceso, para mantener la supervisión de éste. La coordinación y distribución de tareas se establece teniendo en cuenta los datos técnicos e información obtenidos de los flujos de material y de información y se ejecuta a través de una red jerarquizada de microcomputadores y unidades de control programables.

7.3.3. Organización Funcional de un FMS

La organización funcional de un FMS puede estructurarse mediante tres conceptos: planificación de las misiones a realizar por todos y cada uno de los elementos integrantes del sistema, realización de dichas misiones, control y supervisión de las misiones realizadas y del sistema.

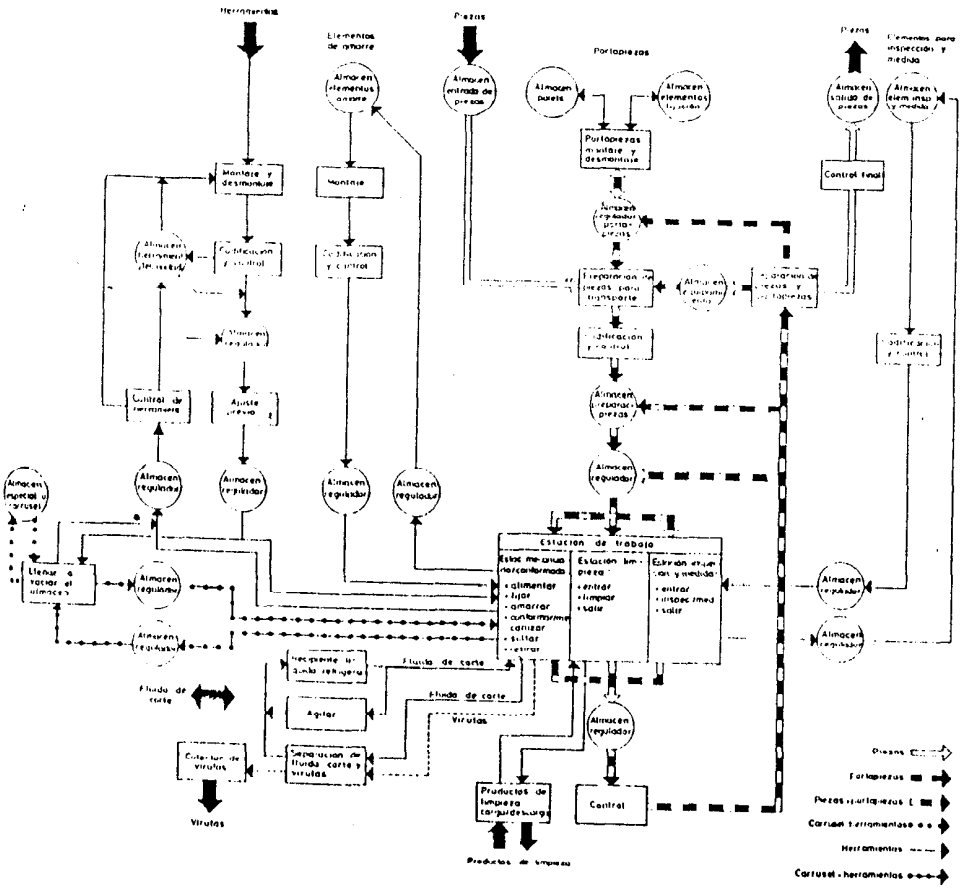
El primero de ellos es atendido por la red informática jerárquicamente superior del sistema. El segundo precisa del establecimeinto de una configuración física del sistema y del estudio pormenorizado del flujo de material. El tercero basa su actuación en la definición de la red informática que ha de controlar el sistema y el estudio específico del flujo de información.

Configuración Física de un FMS. Flujo de Material

La siguiente figura presenta un diagrama que contempla las más importantes actividades que se realizan en todo FMS.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
BIBLIOTECA GENERAL



Se distinguen tres clases fundamentales de instalaciones de almacenamiento:

1.- Almacenes: donde el material está ordenado según posicionamientos conocidos, accecibles y sin limitaciones en el tiempo y en la cantidad.

2.- Almacenes Reguladores Intermedios: destinados a regular los problemas tipo cuello de botella en el transporte y las oscilaciones de capacidad.

3.- Almacenes Especiales: destinados al almacenamiento, preparación y alimentación o retirada de herramientas en las estaciones de trabajo.

Las Estaciones de Trabajo debe considerarse en su concepto plural de puestos de trabajo y sus correspondientes combinaciones. El sistema dispone de un amplio y variado número de estaciones de trabajo. En el esquema presentado cabe distinguir, dentro del flujo de material, una serie de flujos secundarios que contribuyen a sistematizar el conjunto y a definir más claramente las funciones, entre ellos están los que se citan a continuación.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

El Flujo de Piezas. La configuración del conjunto transportador de piezas se realiza situando sobre cada palet elementos que soportan las piezas, con sus correspondientes mecanismos de fijación. Dichos conjuntos son enviados al almacén regulador de piezas donde, además, llegan los conjuntos procedentes del desmontaje de las piezas ya mecanizadas. Con la incorporación de dichos conjuntos de las correspondientes piezas, quedan éstas en condiciones de ser transportadas. Frente a dicha alternativa, existe la más compleja de que éstas vengan libremente sobre una cinta transportadora, sean identificadas por células de visión con cámaras de TV y sistemas de reconocimientos de formas y enviadas por uno o varios robots, directamente, al sistema de amarre o mordazas de la máquina-herramienta.

Seguidamente, la pieza es codificada en cuanto a su tipo y estado de macanizado y, posteriormente, después de ser controlada, es llevada a un almacén general de distribución, al que también retornan aquellas piezas que requieren la realización de más ciclos de trabajo. Según la secuencia fijada por el sistema de transporte, los conjuntos son llevados al almacén regulador de la estación de trabajo y de allí pasan a ésta, para que se realicen los correspondientes ciclos de trabajo. Una vez mecanizadas, las piezas pasan al almacén regulador siguiente, de donde son tomadas para someterlas al control de calidad. Finalmente, son separadas del conjunto transportador y enviadas al almacén general de piezas.

El Flujo de Elementos de Sujeción. Para el posicionamiento de conjuntos portadores de piezas en la estación de trabajo se requiere que ésta disponga de medios adecuados de sujeción o fijación. A tal fin, se dispone de un almacén en el cual están depositados con su correspondiente codificación. Una vez controlada su identificación, los medios de amarre o mordaza son enviados al correspondiente almacén regulador de la estación de trabajo, para pasar finalmente a ésta. El retorno de estos elementos se realiza a través de otro almacén regulador hasta el almacén general.

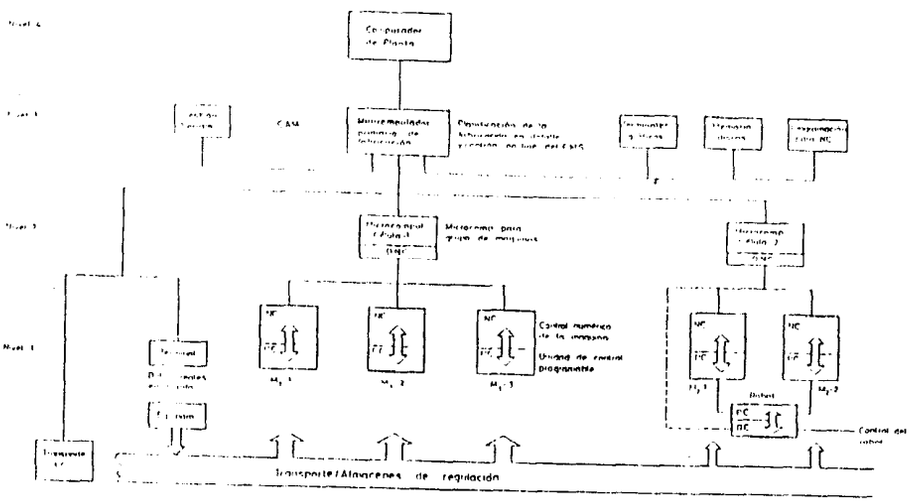
El Flujo de Herramientas. Según el tipo de máquina y proceso de conformación, pueden requerirse herramientas aisladas o carruseles de herramientas, controladas y preperadas para ir directamente a la máquina o a almacenes reguladores conectados con los carruseles.

El Flujo de Elementos de Control de Calidad. Para realizar las funciones de control y seguridad de calidad, han de situarse los distintos dispositivos y sensores correspondientes en la estación de trabajo. Para ello se parte de un almacén central donde están codificados, pasando por un proceso de puesta a punto y un almacen regulador, son llevados a la estación. En forma similar se realiza el retorno.

Configuración Informática y Control de un FMS

El control y supervisión de los FMS se realiza, usualmente, a través de un sistema de información jerarquizado, integrado por unidades de control autárquicas con inteligencia descentralizada, un sistema de transmisión de datos y un computador primario de fabricación que actúa como unidad central de control y supervisión para toda la instalación. La siguiente figura muestra el esquema de una configuración clásica.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



Las funciones a realizar por el sistema están estructuradas en tres niveles jerárquicos, que correspondería propiamente al proceso de CAM.

El primer nivel sirve, básicamente, a dos funciones: unas propias del control numérico (NC) de cada máquina y otras correspondientes al control del entorno de la misma. Mientras que para las primeras se utiliza la unidad destinada a realizar el Control Numérico por Computador (CNC) de cada máquina, con posibilidad de efectuar Control Númerico Directo (DNC), las segundas se valen de una unidad de Control Programable (PC), a la cual se le pueda conectar periféricos inteligentes, como sistemas de toma de datos en la máquina, sistemas para diagnosis, terminales de supervisión del proceso, sistema de control de herramientas, etc.

El segundo nivel asume tareas de administrador y transmisor de datos entre los niveles jerárquicos superior (computador primario de fabricación) e inferior (control y supervisión de máquinas). Puede ser un sistema multimicroprocesador cuyas funciones son: la distribución de datos para el control de las máquinas e instalaciones; supervisión de los valores actualizados de los procesos CNC y de las unidades PC; pretratamiento de los programas de NC y de los datos destinados a las unidades de PC.

En el tercer nivel se realizan, generalmente, las tareas destinadas a planificar, iniciar y coordinar los procesos que se ejecutan posteriormente durante el ciclode producción. Las funciones básicas a efectuar son: planificación, control y supervisión.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

7.3 4. Control Numérico

INTRODUCCION

El Control Numérico se puede definir como "todo dispositivo, normalmente electrónico, capaz de dirigir posicionamientos de uno o varios órganos mecánicos móviles de forma que las órdenes relativas a sus desplazamientos son elaboradas, en forma totalmente automática, a partir de informaciones numéricas y simbólicas definidas por intermedio de un programa".

La aplicación más universalmente conocida del control numérico es como ayuda en la manufacturación. Posteriormente sus técnicas se han aplicado a otras áreas que van desde la automatización total del proceso de fabricación (máquina- herramienta, robótica, etc) al gobierno de mecanismos de cualquier tipo.

Siempre que las series de fabricación se mantengan dentro de unos límites, el control numérico representa la solución ideal dadas las notables ventajas que se obtienen de su utilización. Entre estas ventajas cabe destacar las siguientes:

a) Posibilidad de fabricación de piezas imposibles o muy difíciles.

b) Seguridad. El control numérico es especialmente recomendable para el trabajo con productos peligrosos.

c) Precisión. Esta ventaja es debida a la mayor precisión de la máquina-herramienta de control numérico respecto a las máquinas clásicas.

d) Aumento de la productividad de las máquinas.

e) Reducción de controles y desechos.

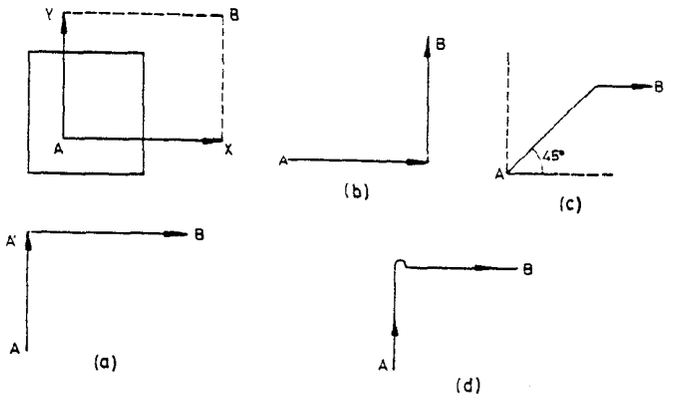
f) Flexibilidad.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

CLASIFICACION DE LOS SISTEMAS DE CONTROL NUMERICO

Independientemente del uso a que se destine, se puede realizar una clasificación de acuerdo con sus posibilidades para gobernar posicionamientos. Podemos distinguir tres tipos básicos: sistemas de posicionamientos, también llamados punto a punto, sistemas punto a punto y paraxial, y sistemas de contorneo.

Supongamos una pieza colocada sobre la mesa de una máquina- herramienta (ver figura) y que en el punto A se quiere realizar una perforación. Sea el eje X el eje longitudinal de la mesa y el eje Y el eje transversal. En la figura, B representa la proyección del eje de la broca sobre la mesa.



El problema de llevar el punto A al punto B se puede resolver de las siguientes formas:

1.- Accionar el motor del eje Y hasta alcanzar el punto A' y a continuación el motor del eje X hasta alcanzar el punto B (a).

2.- Análogo al anterior pero accionando primero el motor del eje longitudinal.

Estos dos modos de direccionamiento reciben el nombre de Posicionamiento Secuencial y se realiza, normalmente, a la máxima velocidad que soporta la máquina.

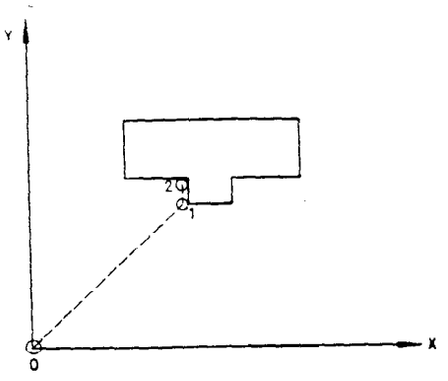
3.- Accionar ambos motores a la vez y a la misma velocidad. En este caso, la trayectoría seguida será una recta de 45°. Una vez llegado a la altura del punto B, el motor del eje Y será parado opara continuar exclusivamente el motor del eje X hasta llegar al punto B (c).

Este tipo de posicionamiento recibe el nombre de Posicionameinto Simultáneo (punto a punto).

4.- Accionamiento secuencial de los motores, pero realizando la aproximación a un punto siempre en el mismo sentido (d). Este tipo de aproximación recibe el nombre de aproximación unidireccional y es utilizado exclusivamente en los posicionamientos punto a punto.

Supongamos ahora que queremos realizar el fresado de la siguiente figura.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



La primera operación será pasar del punto 0 al punto 1 y se realiza de alguna de las formas antes mencionadas.

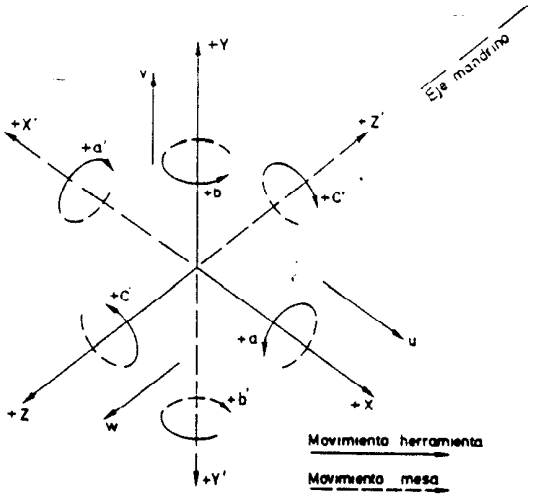
La segunda operación será desplazar la fresa del punto 1 al punto 2. La trayectoria ahora no podrá ser cualquiera, sino que deberá ser una recta perfecta a lo largo del eje Y y sin poder rebasar, en ningún caso, el punto 2, puesto que de otra forma la pieza sería destruida. Este desplazamiento, según en eje Y, no podrá realizarse con cualquier velocidad, sino a la velocidad que permita la naturaleza del material y el diámetro de la fresa utilizada. Este tipo de movimiento recibe el nombre de Movimiento Paraxial y los equipos que los realizan reciben el nombre de equipos punto a punto y paraxial.

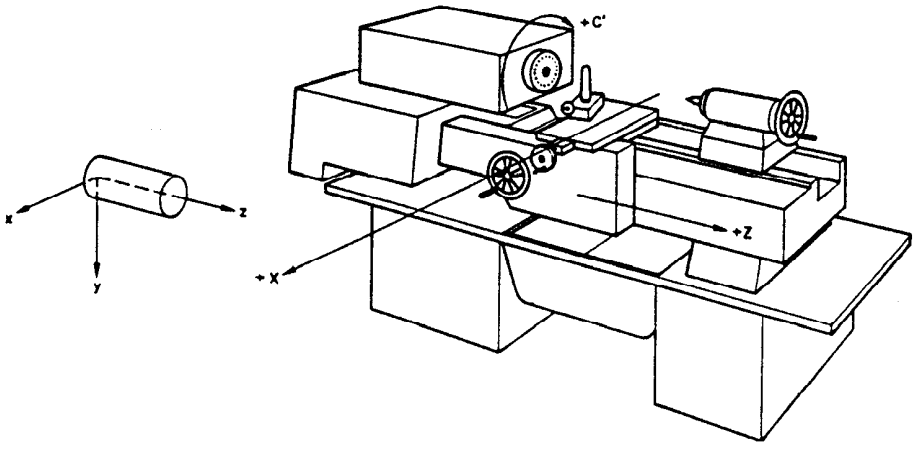
Los equipos que permiten generar curvas reciben el nombre de equipos de contorneo. Los sistemas de contorneo controlan no sólo la posición final, sino el movimiento en cada instante de los ejes en los que se realiza la interpolación. Deberá existir una sincronización perfecta entre los distintos ejes, controlándose, por tanto, la trayectoria real que debe seguir la punta de la herramienta. Con estos sistemas se pueden generar recorridos,tales como: rectas con cualquier pendiente, arcos de circunferencia, cónicas o cualquier otra curva definible metamáticamente.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

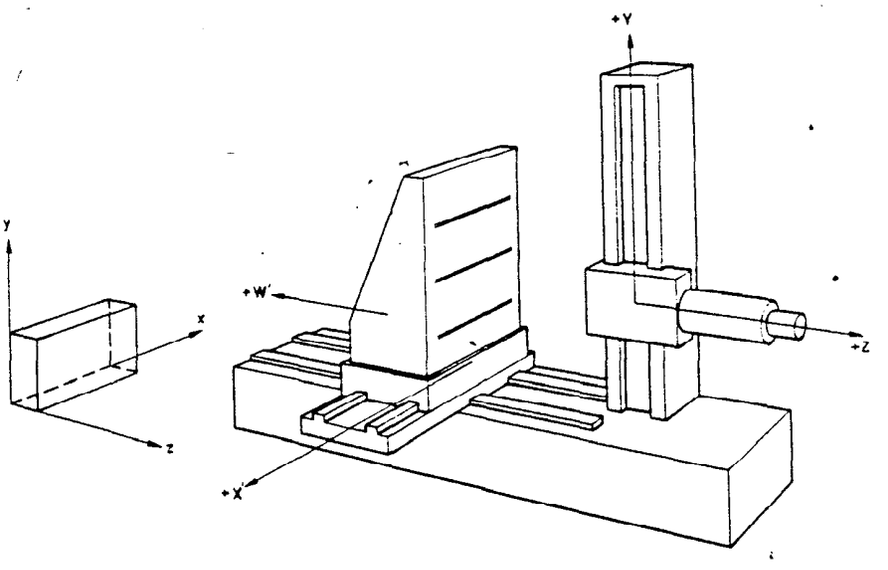
NOMENCLATURA DE EJES Y MOVIMIENTOS

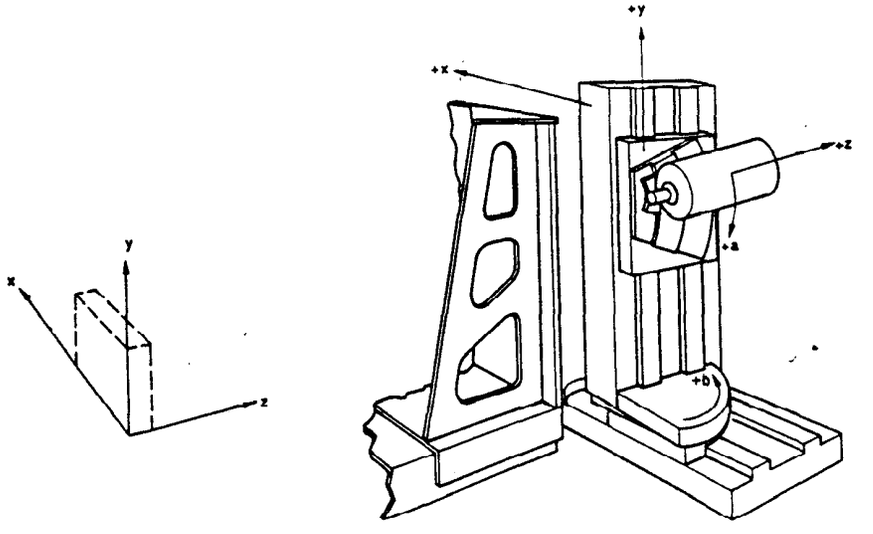
Para la denominación de los ejes de una máquina-herramienta, se aplica la norma EIA estándar RS-267-A conforme a la recomendación ISO-R841 (Control numérico de máquinas-herramienta - Nomenclatura de ejes y movimientos).





© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003





© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

ARQUITECTURA GENERAL DE UN CONTROL NUMERICO

De acuerdo con la definición dada, todo control numérico debe poseer cuatro subconjuntos funcionales:

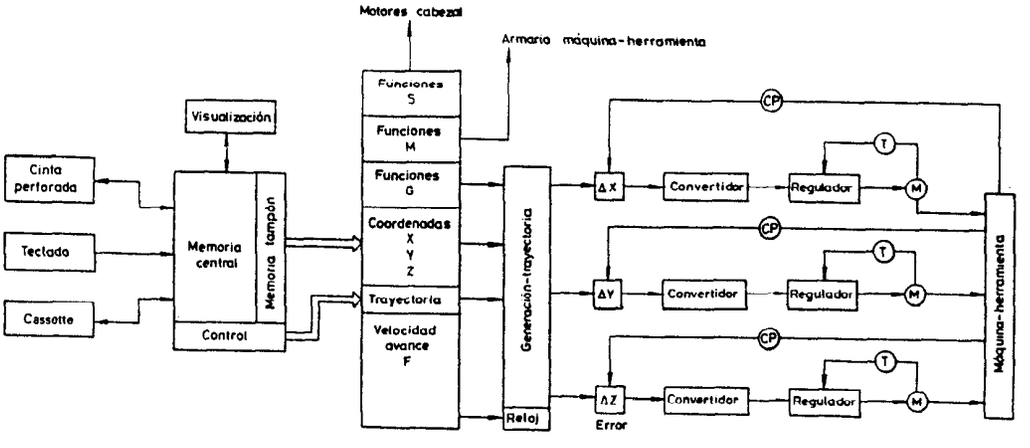
1.- Unidad de entrada-salida de datos y visualización.

2.- Unidad de memoria interna e interpretación de las órdenes.

3.- Unidad de cálculo.

4.- Unidad de enlace con los elementos mecánicos.

La siguiente figura muestra un diagrama funcional simplificado de un control numérico de maquina-herramienta que gobierna tres grados de libertad.



La unidad de entrada de datos sirve para introducir los programas en el equipo de control numérico, utilizando un lenguaje inteligible para áquel. Es el llamado lenguaje máquina. Estos programas pueden ser incorporados en el equipo utilizando medios diferentes: cinta perforadora, tarjeta perforada, cinta magnética, etc.

La unidad de memoria interna almacena no sólo el programa principal, sino también los parámetros máquina y compensaciones (aceleración y deceleración, ceros, compensaciones y correcciones de herramienta en su caso, ganancia de servomecanismos, etc.).

Una vez interpretado un bloque de información (un conjunto de instrucciones), la unidad de cálculo se encarga de crear el conjunto de órdenes que constituirán las referencias de posición y velocidad de los servomecanismos que gobiernan los motores de la máquina o dispositivo mecánico. En este bloque de información está toda la información necesaria para la ejecución de una operación de posicionamiento, es decir, una nueva posición a alcanzar (en forma absoluta o incremental, en coordenadas cartesianas o polares, etc.), velocidad a la que debe realizarse el trayecto, trayectoria que debe describirse.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

En el caso de un posicionamiento punto a punto no será preciso generar ninguna trayectoria, ni realizar ninguna operación compleja. La unidad de cálculo es, en este caso, un simple sumador-restador que calcula el camino total a recorrer para cada uno de los ejes y el sentido del desplazamiento.

Cuando el control numérico posee contorneo, la frase del programa correspondiente deberá suministrar todos los datos adicionales necesarios. Entre estos datos deberá estar el tipo de trayectoria a realizar. Esta trayectoria podrá ser una de las posibles trayectorias que el control puede generar u otra trayectoria distinta. En el primer caso, la unidad de cálculo procederá a la generación de la trayectoria programada, realizando el proceso de convertir una curva definida matemáticamente en un conjunto de pequeños pasos (puntos intermedios) a lo largo de los ejes del dispositivo mecánico. En el segundo caso, se deberán encontrar todos los puntos intermedios que, posteriormente, puedan ser unidos mediantes curvas que el control pueda generar.

PROGRAMACION DE UN CONTROL NUMERICO

Cuando un programador quiere generar un programa de mecanizado puede utilizar métodos distintos según la complejidad de la pieza.

1.- **Programación Manual.** En este caso el programa se realiza en lenguaje máquina, utilizando únicamente razonamientos y cálculos que realiza el operario. El programador debe realizar las siguientes operaciones:

a) Descomponer el mecanizado de la pieza en operaciones elementales, definiendo, además, las diversas corrrecciones del útil y demás órdenes tecnológicas.

b) Determinar el orden preferencialde las distintas operaciones.

c) Definir las curvas y superficies mediantes curvas de las que el control puede generar con la interpolación correspondiente.

d) De acuerdo con cada control numérico y su formato de programación característico, escribir en lenguaje máquina el listado del programa.

e) Introducir el programa

2.- **Programación Automática.** En este caso se utiliza un computador de propósito general como ayuda en la programación, de tal forma que suministre en su salida el programa de la pieza en lenguaje máquina. Utilizando la programación automática, se limita el papel del programador a la elaboración de las órdenes de mando, quedando como tareas del computador todas las operaciones que el computador realiza a mucha más velocidad y con una probabilidad mínima de cometer errores. Entre otras, el computador realiza las siguientes tareas:

a) Calcula automáticamente las cotas características del mecanizado.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital. 2003

b) Realiza la composición automática de los bloques de información.

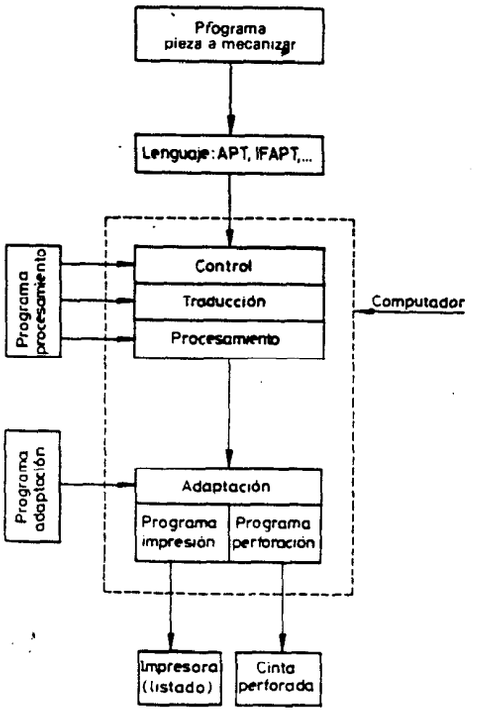
c) Detecta y corrige los errores.

d) Permite la definición de subprogramas para su utilización en las operaciones repetitivas del mecanizado.

e) La mayoría de los programas son del tipo conversacional y permiten una modificación inmediata del programa.

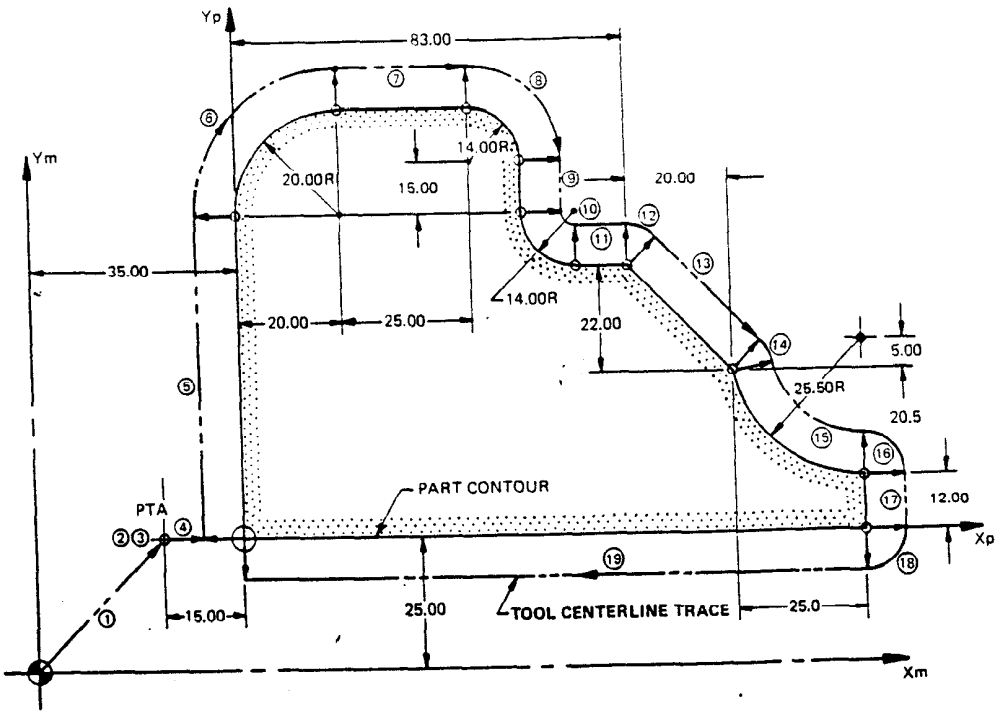
f) Gobierna la perforación automática dela cinta, según el código ISO.

En la actualidad existen numerosos lenguajes de programación automática utilizados en control numérico. El más completo y evolucionado es el lenguaje APT (Automatically Programmed Tools), del cual se han derivado los demás: EXAPT, ADAPT, IFAPT, AUTOSPOT, y otros menos importantes.



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

EJEMPLO



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

```
NOO10 G21 X0 Y0 Z0
N0030 S800 M03
N0040 G45 G00 X0 D01
N0050 G45 Y0 D02
N0060 G18 Z-302.0
N0070 G18 G01 Z-16.0 F10.0
N0080 G91 G17 G01 G41 X15.0 Y0 J68.5 D03
. . . . . .
```

7.3.5. Robots Industriales

Uno de los elementos mas representativos de los actuales sistemas de producción lo constituyen los Robots Industriales que cuentan con un elevado grado de flexibilidad y adaptabilidad a las variaciones del entorno. Estas características permiten que sean utilizados cada vez más en una amplia gama de actividades.

DESCRIPCION GENERAL

Definición de Robot

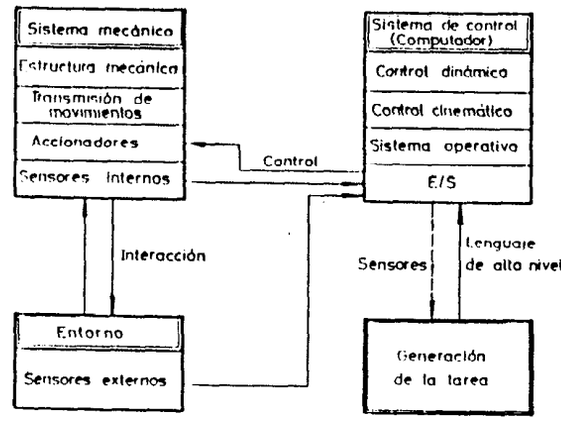
> "Un robot es un manipulador reprogramable y multifuncional, diseñado para mover cargas, piezas, herramientas o dispositivos especiales, según variadas trayectorias, programadas para realizar diferentes trabajos".

La idea más ampliamente reconocida como robot está asociada a la existencia de un dispositivo digital de control que, mediante la ejecución de un programa almacenado en memoria, va dirigiendo los movimientos del sistema mecánico.

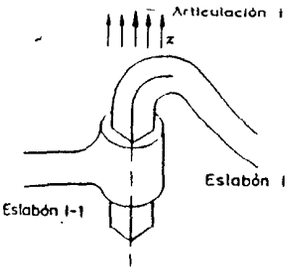
Aunque la definición es muy concisa, no permite diferenciar las distintas generaciones de robots. Actualmente existen dos generaciones. La primera, agrupa a los robots que realizan operaciones elementales, siendo sus movimientos muy repetitivos. La segunda generación, que es la que actualmente se está desarrollando, es capaz de realizar operaciones muy complejas en condiciones de entorno variable, para la cual los robots están equipados con sensores específicos.

Configuración General de un Robot

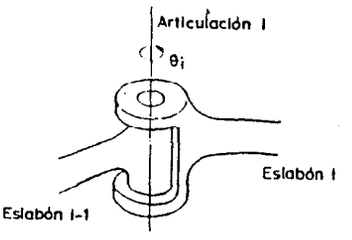
La configuración de un robot se muestra en la siguiente figura:



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



a) Prismática



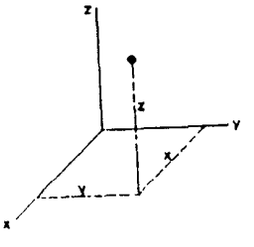
b) De rotación

Se define el número de Grados de Libertad de un robot (GDL) como el número de movimientos independientes que puede realizar su estructura respecto a un sistema de coordenadas inmóvil, situado normalmemte en la base del mismo.

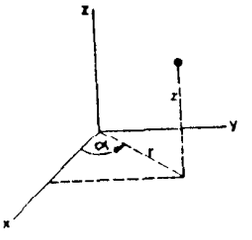
Tipos de Estructuras Mecánicas

Con seis GDL, y contemplando la posibilidad de asignar a las correspondientes articulaciones movimientos de rotación o traslación, se pueden obtener 216 tipos diferentes de estructuras mecánicas.

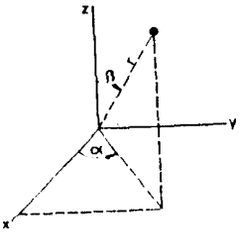
Teniendo en cuenta únicamente el problema de posicionar el extremo del robot, desde el punto de vista geométrico, es usual considerar los cuatro tipos de sistemas de coordenadas representados en la siguiente figura.



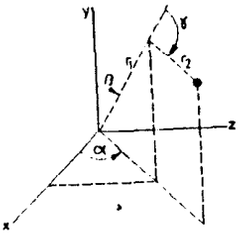
a) Cartesiano



b) Cilíndrico



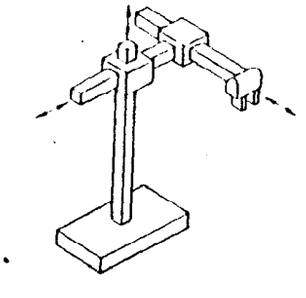
c) Esférico



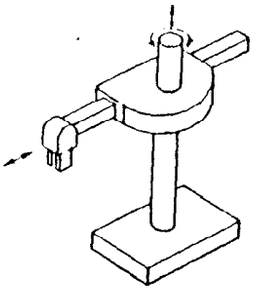
d) Angular

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

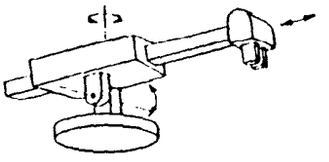
Asociados con estos sistemas de coordenadas, se podrían distinguir cuatro estructuras mecánicas de robots:



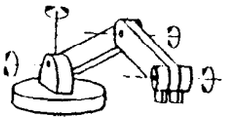
a) Cartesiana



b) Cilíndrica

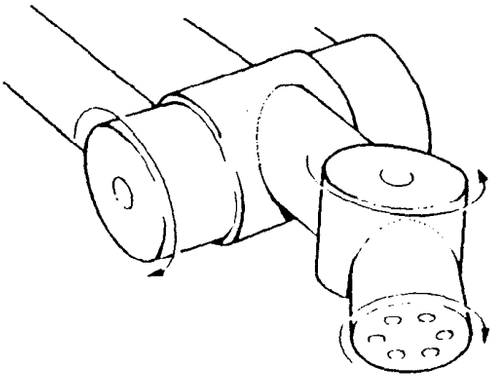


c) Esférica



d) Angular

En cuanto a la orientación de elemento final, se utilizan muñecas con tres rotaciones, según el esquema de la siguiente figura:



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Accionamiento del Sistema Mecánico

El accionamiento del sistema de un robot es el elemento motor que permite mover sus articulaciones. Existen tres tipos de accionamientos, cada uno de ellos asociado a diferentes formas de energía y tecnología: Neumático, Hidraúlico y Eléctrico.

El Accionamiento Neumático usa aire comprimido a presión inferior a 10 bar para accionar generalmente cilindros neumáticos lineales.

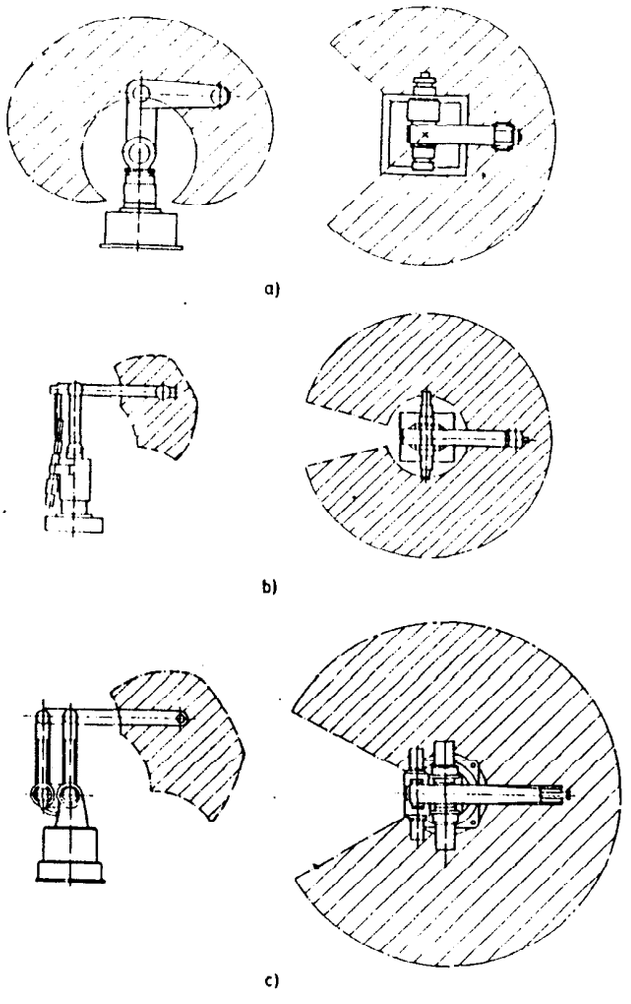
El Accionamiento Hidraúlico utiliza líquidos, que suelen ser aceites, a presión inferior a 100 bar que accionan motores hidraúlicos.

El Accionamiento Eléctrico es el que se usa en la inmensa mayoría de los robots actuales. Su gran ventaja, en comparación con los demás accionamientos, es que permite una precisa y fácil regulación de la posición, a través de servomecanismos. Los motores que se usan son de dos tipos: de paso a paso y de corriente continua.

Campo de Acción de un Robot

Se define el campo de acción de un robot como el conjunto de puntos (espacio) del sistema cartesiano accesible al extremo del mismo. El campo de acción se forma por los ángulos límites de variación de cada una de las articulaciones, en le caso de robots articulados. En la siguiente figura se muestra los campos de acción de diversos robots articulados.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Sensores Internos

Los Sensores Internos permiten al sistema de control conocer la posición de las articulaciones del robot, por lo que cada articulación cuenta con sus propios sensores. Estos suelen situarse lo más cerca posible de aquéllos, para evitar los errores de transmisión.

Sensores de Posición

Se utilizan básicamente dos tipos de sensores de posición: Opticos y Magnéticos. Los primeros funcionan de forma incremental, lo que hace necesario tener la misma referencia del cero cada vez que se vayan a usar. Los segundos sensores funcionan de forma absoluta, no necesitando la sincronización.

Características de Posicionamiento de un Robot

- **Resolución (o Precisión)**: Define en ±mm el mínimo incremento que el robot puede realizar en su extremo.

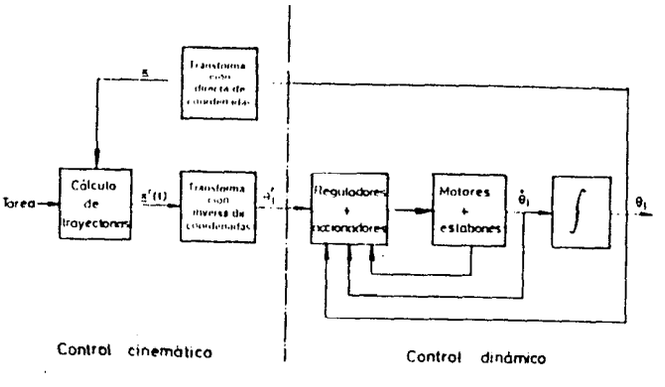
- **Repetitividad**: Se define como una medida estadística del error de posicionamiento del extremo en ±mm al retornar éste al mismo punto del espacio varias veces y desde distintas posiciones iniciales.

- **Exactitud**: Permite conocer con qué error se posiciona el robot respecto de un extremo base inmóvil genérico.

SISTEMA DE CONTROL DE UN ROBOT

El sistema de control tiene como misión gobernar los movimientos del robot necesarios para llevar a cabo una tarea concreta.

El esquema del sistema de control de un robot se representa en la siguiente figura:



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Esta, está separada en dos partes fundamentales:

1.- Control Cinemático. Tiene como misión generar, a partir de la tarea, las referencias angulares de los servos de posición de cada articulación. Para ello se necesita efectuar un cálculo de las trayectorias de los movimientos entre los distintos puntos. Este bloque genera un vector posición de referencia cartesiana, a partir del cual, mediante un algoritmo denominado transformación inversa de coordenadas, se obtienen los ángulos de referencia.

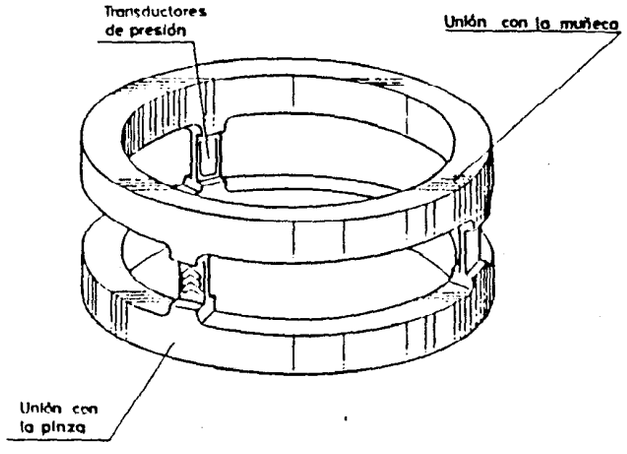
2.- Control Dinámico. Tiene como misión controlar la posición y la velocidad reales de cada una de las articulaciones. Esto se realiza mediante los servos de posición.

SENSORES EXTERNOS AL ROBOT

Los sensores externos al robot suministran al sistema de control la información necesaria sobre el entorno y sobre la operación en curso. Algunos de estos sensores suelen situarse en el extremo del robot, mientras otros se sitúan fuera de él.

Sensores de Esfuerzos

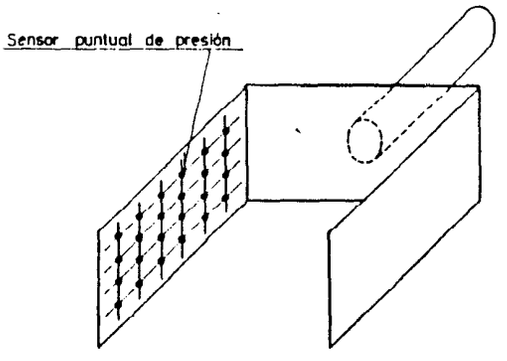
Los sensores de esfuerzos se sitúan, normalmente, entre la muñeca y la pinza del robot midiendo los esfuerzos y los pares en el extremo del robot que se generan durante la operación.



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

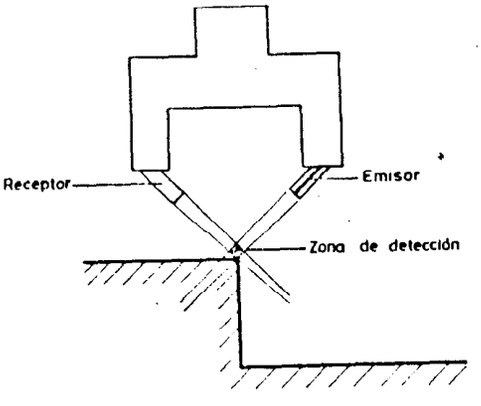
Sensores de Tacto y Deslizamiento

Estos sensores se sitúan en los dedos (extremidades) de la pinza. Permiten conocer aproximadamente las formas del objeto y detectar si ha existido deslizamiento. Constructivamente suelen estar formados por matrices de sensores de presión puntual.



Sensores de Proximidad

Este tipo de sensores permite detectar la presencia de objetos u obstáculos cerca del elemento final sin ningún contacto. Suelen ser de tipo inductivo o infrarrojo, situándose en cada uno de los dedos del robot. La siguiente figura presenta una pinza con un emisor y receptor de infrarrojos.



© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

Sensores de Visión

Este tipo de sensores permiten detectar e identificar cualquier variación del entorno de su área de actuación.

Los tipos de sensores de visión que se utilizan en robótica son: Vidicón, Dispositivos de Acoplo de Carga y RAM Dinámicas.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

CAPITULO 8

EL ESTANDAR GRAFICO

8.1 PERSPECTIVA HISTORICA

El GKS (Grafical Kernel System) se define como un estándar de manipulación de gráficos por ordenador. Fué desarrollado en los años 1975-1983. Por aquellas fechas se experimentó un gran incremento en la variedad y uso de dispositivos gráficos, debido al descubrimiento de nuevos equipos y sobre todo al abaratamiento de los ya existentes. Esto originó una diversificación de los medios empleados para controlarlos, teniendo que reescribir el software necesario cada vez que se cambiaba el equipo o se transportaba un paquete gráfico de un sistema computador a otro.

Para remediar esta situación, varios grupos de trabajo, (especialmente el Instituto Alemán de Normalización, D.I.N.), todos ellos pertenecientes a la Organización Internacional de Standards (I.S.O.) empezaron a trabajar en el diseño del GKS. Este tenía como objetivo compatibilizar el software de gráficos creados para diferentes equipos, independientemente del hardware. Para ello, el GKS se construye como un núcleo (Kernel) que controla todos los aspectos de un sistema informático que tenga relación directa con los gráficos, tanto a nivel de dispositivos de entrada, proceso o salida; sobre dicho núcleo se podrán construir programas generales que serán transportables, concentrando las rutinas dependientes del hardware en el GKS.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

8.2 PRINCIPIOS Y OBJETIVOS DE ESTANDARD

Las principales razones para la introducción de estandard de gráficos por ordenador son las siguientes:

- Permitir que los programas de aplicación que utilicen gráficos sean fácilmente transportables entre diferentes sistemas informáticos.

- Ayudar a la comprensión y uso de las capacidades gráficas de los ordenadores por parte de los programadores.

- Proporcionar a los fabricantes de equipos gráficos una guía para la adecuada implantación de combinaciones prácticas de capacidades gráficas en un dispositivo.

Para alcanzar dichos objetivos, el GKS fue diseñado de acuerdo a los siguientes requisitos:

- GKS debe incluir las capacidades esenciales para cubrir todos los aspectos gráficos, desde salidas simples hasta aplicaciones altamente interactivas.

- Todo el rango de equipos gráficos, incluyendo dispositivos vectoriales y de

barrido, microfilmadores, pantallas de tubo de almacenamiento, de refresco y de color, deben ser controlables por GKS de una manera uniforme.

- GKS debe proporcionar todas las capacidades requeridas por la mayoría de las aplicaciones, sin llegar a ser un sistema innecesariamente grande y complejo.

8.3 PRINCIPALES CONCEPTOS DEL GKS

REPRESENTACION. Una de las tareas básicas de un sistema gráfico es la generación de imágenes; el concepto correspondiente a esta tarea es el de representación gráfica. Una imagen se compone de primitivas gráficas, y su apariencia visual se controla mediante un conjunto de atributos.

SISTEMAS DE COORDENADAS Y TRANSFORMACIONES. Las primitivas gráficas se crean en uno o varios sistemas de coordenadas dadas por el usuario. Estas primitivas deben representarse en diferentes estaciones de trabajo que tendrán diferentes sistemas de coordenadas del dispositivo.

ESTACIONES DE TRABAJO. Los dispositivos de entrada y salida se agrupan en lo que se denominan "estaciones gráficas" (WorkStations, W.S.). Son manejadas por un operador. La W.S. es un concepto importante y una contribución original de GKS a la metodología del diseño de un sistema gráfico.

ENTRADA. Para comunicarse con el operador de la W.S., el programa de aplicación debe disponer de algún sistema para recibir datos. Además de la entrada de datos gráficos (como coordenadas, etc.), GKS maneja entrada alfanumérica, dispositivos de menú como teclas de función y dispositivos del tipo valor-entre-rango, como potenciómetros. Se procesa la entrada de forma independiente del dispositivo a base de definir dispositivos lógicos.

SEGMENTACION. El procesamiento de partes de una imagen nos lleva al concepto de segmentación. Una imagen se compone de segmentos que pueden ser representados, transformados, copiados o borrados, independientemente unos de otros. Los segmentos pueden ser identificados por el operador y su identificación pasarse al programa. GKS contiene potentes servicios de segmentación, principalmente, proporcionando un almacenamiento de segmentos independiente de la W.S., junto con funciones de copia de segmentos a W.S. o a otros segmentos.

METAFILE. El "metafile" es una forma de almacenar imágenes en ficheros, y transferirlas a otros sistemas. Aumenta considerablemente la flexibilidad del sistema, pues el formato lógico del fichero tampoco depende de las características hardware de los dispositivos gráficos.

LISTAS DE ESTADO. En todo momento GKS está en un estado de operación que se representa por una serie de valores concretos en unas listas de estado. Estos valores los cambian los distintos procedimientos del GKS, llamados por el programa de aplicación. Los conceptos de estado y de listas de estado fueron desarrollados durante el diseño del GKS para permitir una perfecta definición del efecto de cada función y para proporcionar una ayuda al implementador.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

NIVELES. Las funciones del GKS se agrupan en 9 niveles. El nivel mínimo contiene sólo las funciones básicas, el máximo incluye todas la funciones del GKS. Esto permite al implementador el llevar a cabo sólo las funciones necesarias para su rango de aplicaciones, y evita el tener que realizar complejas y costosas simulaciones en sitemas gráficos poco potentes.

MANEJO DE ERRORES. GKS define, con cada función, un número de condiciones de error que pueden darse. El programa de aplicación puede controlar el proceso en caso de error o puede utilizar un tratamiento de errores estandard incluído en el GKS.

DIMENSIONALIDAD. Para la mayoría de las aplicaciones gráficas, un sistema bidimensional, como el que ofrece el GKS, será suficiente. Todas las coordenadas de entrada y de salida están en dos dimensiones. De cualquier manera, hay aplicaciones que requieren una representación 3-D.

8.4 PRIMITIVAS DE DIBUJO

Son las funciones básicas que generan imágenes. Se especifican por su geometría y por la forma en que aparecen en la superficie de dibujo de una W.S. Estos aspectos se controlan mediante una serie de atributos pertenecientes a cada primitiva. Algunos atributos pueden variar de una W.S. a otra (atributos dependientes de la W.S.). En GKS, existen funciones para la creación de primitivas y para la asignación de valores a los atributos. Para algunos de ellos se puede elegir desde programa el que sean o no dependientes de la W.S.

En GKS existen 6 primitivas de dibujo, que son las siguientes:

POLYLINE. Dibuja un polígono cualquiera como un conjunto de líneas rectas. Tiene como parámetros los puntos entre los que se dibujarán las líneas y su número. Los puntos se especifican en coordenadas del mundo absolutas: las coordenadas relativas no están implementadas en GKS.

Es un procedimiento más general que el de dibujar una línea (no definido en GKS) y que se puede realizar simplemente indicando sólo dos puntos. Hay razones para definir la primitiva de esta forma general; es más adecuada para dibujar polígonos, y es conveniente para sistemas de dibujo de planos, en cartografía, o en sistemas de trabajo por lotes; si hay que dibujar varias líneas, es más eficiente agruparlas en unas pocas primitivas que realizar muchas llamadas a un procedimiento de dibujar líneas, pues es más rápido y ahorra el paso de parámetros. Esto se cumple especialmente a la hora de dibujar contornos de figuras más o menos complejas.

POLYMARKER. Genera símbolos de un tipo determinado centrados en posiciones dadas. Estos símbolos se llaman "marcas", y tienen una apariencia específica; se usan para identificar un conjunto de puntos. Sus parámetros son los mismos que POLYLINE; este paralelismo puede ser útil para resaltar los vértices de un polígono dado.

TEXT. Genera una ristra de caracteres en una posición determinada. Esta posición viene expresada en coordenadas del mundo, lo que permite situar fácilmente el

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

texto en relación con otras primitivas de dibujo. GKS controla la goemetría de cada carácter y define atributos que permiten un control preciso del tamaño, forma y orientación de caracteres, caligrafía, espaciado, orientación del texto, etc. Todas estas especificaciones exigen un complejo generador de caracteres, por lo que dentro del GKS se contempla la posibilidad de que el usuario pueda necesitar un texto menos elaborado pero más eficiente en términos de tiempo de cálculo.

FILLAREA. Genera un polígono cerrado que puede estar relleno con un color uniforme, con un patrón o una trama de rayado, o simplemente estar vacío. El objetivo de esta primitiva es la representación de áreas.

El dibujo de áreas sólidas ha sido siempre una opción interesante en sistemas de dibujo, pero la aparición de pantallas de barrido en color ha ocacionado que esta importacia sea aún mayor. Las pantallas en blanco y negro pueden representar áreas pero para distinguir áreas adyacentes es necesario rellenarlas con "patterns" característicos. Las pantallas vectoriales pueden simular el relleno mediante un rayado, pero esto puede restar eficiencia y en estos casos sólo se podrá dibujar el perímetro del polígono (estilo "Hollow"). Todas estas posibilidades se contemplan en GKS.

CELLARRAY. Genera una matriz de casillas rectangulares, cada una con un color determinado. Es la generalización de una matriz de pixels en un dispositivo de barrido; las casillas no tienen porqué corresponderse con los pixels definidos por el hardware del dispositivo. Todas las W.S. deben aceptar esta primitiva, pero aquellas que no puedan simularla adecuadamente pueden limitarse a dibujar los bordes del rectángulo.

GENERALIZED DRAWING PRIMITIVE (GDP). GKS controla las capacidades geométricas especiales de cada W.S. (dibujo de círculos, elipses, curvas de interpolación, arcos, etc). Estos objetos son caracterizados por un identificador, un conjunto de puntos y datos adicionales. GKS aplica todas las transformaciones a dichos puntos pero deja su interpretación a la W.S.

Esta primitiva es una forma estándar de agrupar y aislar las capacidades no estándar de cada sistema computador particular, de forma que al transportar un programa de aplicación de un isstema a otro sólo habrá que modificar las llamadas a GDP para que la salida gráfica sea la misma en ambos sistemas. En cualquier caso, GKS recomienda evitar su uso, para aumentar la compatibilidad de los programas.

8.5 ATRIBUTOS

PICK IDENTIFIER (Identificación de elección). Es un número asignado a las primitivas individuales en un segmento y devuelto por el dispositivo "Pick". El mismo Pick Identifier puede ser asignado a diferentes primitivas.

COLOUR (Color). Se especifica como una terna de valores comprendidos entre 0 y 1, que controlarán las intensidades del rojo, verde y azul.

LINE TYPE (Tipo de línea). Se usan para distinguir diferentes estilos de líneas, como

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

continua, de puntos, etc.

LINE WIDTH SCALE FACTOR (Factor de escala del ancho de línea). El ancho real de una línea viene determinado por el producto de un ancho nominal por este atributo.

MARKET TYPE (Tipo de marca). Es un número especificando la forma del símbolo utilizado para identificar puntos en PolyMarket.

MARKET SIZE SCALE FACTOR (Factor de escala del tamaño de la marca). El tamaño real de una marca viene determinado por el producto de un ancho nominal por este atributo.

TEXT FONT. Un número que selecciona una determinada representación de los caracteres.

TEXT PRECISION (Precisión de Text). Describe el grado de fidelidad con la que se ajusta la posición de los caracteres, su tamaño, orientación y forma, con aquellos pedidos por el programa de aplicación. Hay tres grados; esto es debido a que normalmente los generadores de caracteres disponibles sólo permiten un conjunto restringido de tales características; el programa de aplicación deberá decidir si utilizar las facilidades existentes en el dispositivo o recurrir a generadores software, más potentes pero más lentos.

CHARACTER HEIGHT (Altura del caracter). La extensión vertical.

CHARACTER UP VECTOR. Es un vector que describe la desviación de la orientación de los caracteres con respecto a la vertical.

TEXT PATH (recorrido del texto). Es la dirección de escritura de la secuencia de caracteres. Normalmente, como en este texto, será hacia la derecha, pero también puede ser a la izquierda, arriba o abajo.

CHARACTER SPACING (Espaciado de los caracteres). Es el espacio adicional entre dos caracteres, aparte del que haya definido el diseñador del Text Font.

CHARACTER ALIGNMENT (Alineación de los caracteres). Describe cómo se posiciona el texto respecto al punto de referencia dado en la primitiva "Text".

INTERIOR STYLE (Estilo interior). Indica el estilo en que se rellenará el área definida por FillArea. Tiene cuatro posibles valores; Hollow (no relleno, sólo se dibuja el contorno del polígono), Solid (relleno uniforme), Pattern (relleno con un patrón de dibujo) y Hatch (rayado).

PATTERN SIZE (Tamaño del patrón). Determina las dimensiones del rectángulo básico del patrón.

PATTERN REFERENCE POINT (Punto de Referencia del Patrón). Indica el punto inicial de aplicación del rectángulo de pattern. La esquina inferior izquierda del rectángulo se coloca en el punto de referencia; luego dicho rectángulo es repetido en todas direcciones hasta rellenar el área.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

PATTERN ARRAY (Matriz del patrón). Un pattern se define por una matriz de casillas rectangulares, cada una de las cuales tiene asignada un color.

HATCH STYLE (Estilo del rayado). Selecciona uno de los tipos de reyado definidos por el implementador y presentes en la W.S.

8.6 SISTEMAS DE COORDENADAS Y TRANSFORMACIONES

Cada programa de aplicación procesará datos que pueden estar expresados en sistemas de coordenadas que variarán de una aplicación a otra, incluso el usuario puede desear variar dinámicamente estas coordenadas. Por otro lado, se hace necesario que el núcleo gráfico controle estos aspectos.

Se definen las Coordenadas del Mundo (W.C.), que es un sistema de coordenadas cartesianas independientes del dispositivo usadas por el programa de aplicación para proceso de datos gráficos de entrada y salida.

Otros sistemas de coordenadas no cartesianas (por ejemplo, polares) deberán ser procesados antes de utilizar el GKS.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

8.7 NIVELES DEL GKS

No todos los sistemas gráficos tienen las mismas capacidades, y esto debe tenerlo en cuenta el estardar. Para ello, GKS define 9 niveles diferentes; cada implementación deberá decidir que nivel podrá soportar en función de las capacidades del equipo.

Hay 3 posibles niveles en cuanto a capacidades gráficas de salida, y otros tres de entrada.

8.8 NIVELES DE ENTRADA

- **Nivel (a)**. En este nivel no existen dispositivos de entrada de ningún tipo. Esta situación se da ampliamente en la realidad, en sistemas tales como equipos de cartografía, dibujo de planos, imágenes publicitarias, etc.

- **Nivel (b)**. Los dispositivos de entrada pueden "pedir" datos al exterior (Request Input). Se definen todos los tipos de dispositivos posibles, que podrán o no estar presentes; en cualquier caso, todos los dispositivos estarán controlados por el GKS.

- **Nivel (c)**. Tiene las mismas capacidades del nivel (b), y además permite los modos 'Event Input' y 'Sample Input'. En el primero, se pueden introducir datos al ordenador en cualquier momento, el GKS se encargará de almacenarlos en un buffer de entrada. En el modo 'Sample', el GKS interrogará el estado actual de un dispositivo y devolverá su valor (muestreo). Estos dos modos son necesarios en sistemas interactivos, como en entornos CAD/CAM, simuladores, etc.

8.9 NIVELES DE SALIDA

- **Nivel (0)**. Contiene las capacidades básicas de dibujo, primitivas gráficas, sistemas de coordenadas, recorte, etc.

- **Nivel (1)**. Introduce el concepto de segmento, que es un conjunto de primitivas gráficas, considerándolo como un todo, sobre el que se pueden realizarse varias operaciones: creación, copia, borrado, escalado, etc.

- **Nivel (2)**. Existe, además, el concepto de WISS (Workstation Independent Segment Storage, almacenamiento de segmentos independiente de la estación de trabajo). Es un dispositivo virtual, y como tal, completamente independiente de cualquier medio físico. Un segmento puede ser definido sobre una WISS y de allí ser copiado a cualquier dispositivo; es, por tanto, un medio sencillo de trasladar gráficos entre dispositivos.

8.10 WORKSTATIONS

Las características de un dispositivo físico particular deben de ser transparentes a los programas de aplicación; para conseguir esto, se obliga a que los usuarios del GKS no dibujen en una pantalla o un plotter, ni toman datos de un teclado o una tableta digitalizadora, sino que interactuan con una estación de trabajo. Se definen 6 tipos de WorksStations:

- **OUTPUT**. Son estaciones con capacidades únicamente de salida. Es el tipo de estación de trabajo que se eligirá para asociarlo por ejemplo, a un plotter.

- **INPUT**. Sólo tienen capacidades de entrada, y pueden representar cualquier tipo de sistema de adquisición de datos que no disponga de salida gráfica.

- **OUTIN**. Es el caso más general. Es una estación de trabajo con capacidades de entrada/salida; normalmente, un operador utilizará una tableta digitalizadora o un ratón para introducir dibujos que podrá ver representados en una pantalla de cualquier tipo, teniendo un teclado para entrada de comandos, una impresora gráfica o un plotter y otros dispositivos; todos ellos pueden agruparse en un único dispositivo abstracto e independiente del hardware, la W.S. de tipo OUTIN.

- **WISS**: Fue definida anteriormente al hablar del nivel (2) de salida del GKS. Al ser un dispositivo virtual, una implementación determinada puede elegir el modo de representarla: como una estructura de datos en memoria, un fichero, un mapa de bits de la imagen o cualquier otro medio apropiado, que de cara al usuario cumpla con las características de independencia del hardware.

- **MO, MI**. Incluso el almacenamiento permanente de datos en fichero se realiza de forma independiente de la máquina real en que sea implementado el GKS, definiendo dos tipos particulares de estaciones de trabajo que son en realidad ficheros especiales (Metafiles), tanto de entrada (MI) como de salida (MO). Su estructura y funcionamiento se explican en el siguiente punto.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

8.11 METAFILE

Siguiendo con el principio de independencia del hardware y de transportabilidad de programas y de datos que tiene como objetivo el GKS, se plantea la necesidad de definir algún modo de almacenamiento permanente de gráficos, es decir, especificar el formato del fichero de gráficos. Si lo que se almacenara fuera la imagen, el digujo en sí, se plantearían graves problemas a la hora de transportar dibujos entre sistemas con mapas de bits diferentes, tanto en dimensiones como en el modo de representar cada punto (número de bits empleados, colores disponibles, etc.). Para resolver este problema, lo que el GKS almacena son las primitivas de dibujo que dieron lugar a la imagen. Cada registro del fichero contendrá un campo indicando la primitiva de dibujo, longitud del registro, parámetros de la primitiva, etc.

Además de ser un almacenamiento totalmente independiente del hardware y de permitir el redibujo de imágenes en diferentes dispositivos (pertenecientes o no al sistema computador en que fueron creadas), el Metafile tiene otras ventajas, como son:

- Posibilidad de combinar diferentes fuentes de datos de manera uniforme (por ejemplo, alguien puede diseñar un paisaje de fondo para una escena y guardarlo en Metafile, y combinarlo con diferentes primeros planos guardados en otros Metafiles con sólo redibujar los ficheros).

- Posibilidad de edición posterior de imágenes.

- Fecilidad de interconexión de sistemas de forma estandar y directa.

- Posibilidad de redibujo de imágenes con preproceso selectivo; por ejemplo, se puede dibujar la imagen eliminando las primitivas de dibujo de rectas o de rellenado de polígonos, o aplicar diversas transformaciones geométricas implementadas en GKS a las diferentes primitivas sin necesidad de rellenar algoritmos de proceso de imágenes.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

8.12 SEGMENTOS

Una imagen se construye a partir de primitivas. Después de haber generado y representado una imagen en un display, las primitivas no son accesibles, pero en muchas ocaciones se necesita acceder a ellas, fundamentalmente para edición, redibujo en la misma o en otra W.S (por ejemplo, para "hard copy") o para repetir una subimagen dentro de una imagen en diferentes posiciones y con diferentes atributos de color o tamaño. Para ello, las imágenes deben ser estructuradas, etiquetadas, descritas y almacenadas, y debe disponer de un conjunto de operadores para manipularlas.

Un nucleo como el GKS no puede contemplar todas las posibilidades de un sistema de manipulación y estructuración de imágenes, pero el almacenamiento de segmentos del GKS proporciona herramientas básicas que serán apropiadas para muchas aplicaciones.

Al llamar al procedimiento CREATE_SEGMENT, GKS entra en el estado SGOP (existe un segmento abierto), en el cual todas las primitivas generadas a partir de ese momento, junto con sus atributos asociados, se almacenan en una estructura llamada

Segmento. Por simplicidad se han introducido algunas limitaciones, como que no se puede abrir dos segmentos al mismo tiempo, un segmento no puede hacer referencia a otro, y no se puede reabrir un segmento una vez cerrado con CLOSE_SEGMENT. Pero pueden ser manipulados, y los cambios efectuados se reflejarán en todas las W.S. a las que esté asociado el segmento. Por ejemplo, si se transforma un segmento mediante un escalado, una rotación y/o una traslación, el cambio se verá reflejado en todas las W.S. en las que dicho segmento aparezca representado.

8.13 ENTRADA

El nucleo GKS también controla todos los dispositivos de entrada de datos, lo cual abarca una serie de aspectos de implementación y de interfaz con el usuario muy complejos. Existen dispositivos que proporcionan datos muy diversos, como números enteros, posiciones o pares de números reales, caracteres, etc. Para cubrir esta variedad se han definido seis categorías de dispositivos lógicos:

- **LOCATOR**: devuelve una posición en coordenadas del mundo, es decir, un par de números reales, y el número de transformación aplicada. Dicha transformación es necesaria para convertir los datos obtenidos del dispositivo físico a coordenadas del munto, más apropiadas para ser manipuladas por el programa de aplicación.

- **STROKE**: devuelve una serie de posiciones en coordenadas del mundo. En la práctica funciona y puede ser implementado como una serie de peticiones a un dispositivo de la categoría LOCATOR, pero se han introducido para mantener el paralelismo con el tipo de parámetros que necesitan las primitivas POLYLINE, POLYMARKER y FILLAREA.

- **VALUATOR**: devuelve un número real, comprendido en un cierto rango.

- **CHOICE**: devuelve un entero no negativo, que representa una elección entre una serie de alternativas, como por ejemplo un menú de opciones.

- **PICK**: devuelve el estado del dispositivo PICK (correcto o no), un nombre de segmento y un identificador de PICK. Este dispositivo sirve para identificar primitivas contenidas dentro de segmentos.

- **STRING**: devuelve una ristra de caracteres. Se corresponde en general, con un teclado alfanumérico.

Cada uno de estos dispositivos lógicos puede estar operando en uno de los tres modos posibles:

- **REQUEST**: En este modo, cada vez que el programa de aplicación pide un dato se detiene su ejecución normal y toma el control el GKS, manteniéndolo hasta que el operador introduzca un dato válido o genere una acción de parada, como un código Break.

- **SAMPLE**: Una petición de un dato hace que el GKS interrogue el estado del dispositivo y devuelva su valor actual, sin esperar a que el operador genere ninguna acción.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003

- **EVENT**: El GKS gestiona una cola FIFO de eventos cuyos elementos constan del identificador de un dispositivo lógico y de un valor de entrada. Estos elementos se generan asíncronamente, por medio de una acción del operador. El programa de aplicación podrá tomar el primer elemento de la cola y examinar su contenido.

8.14 COMPONENTES DE UN DISPOSITIVO DE ENTRADA

Para cada dispositivo lógico existen una serie de parámetros que definen su modo de funcionamiento y su interfase de cara al usuario.

- **Interacción**. En los modos SAMPLE y EVENT, los dispositivos están interactuando durante todo el tiempo, pero en modo REQUEST sólo durante la ejecución de una petición.

- **Medida**. El valor devuelto por un dispositivo no se considera como una variable simple, sino como el resultado de un proceso que hace un "mapeo" de los dispositivos físicos a dispositivos lógicos. El estado actual de dicho proceso de medida es el valor lógico devuelto por el GKS. El proceso existe mientras se esté llevando a cabo una interacción.

- **Prompt/Eco**. El "Prompt" es una información que se le da al operador para indicarle que el dispositivo está preparado para su uso. El Eco indica el estado actual del proceso de medida. Para cada categoría de dispositivo de entrada, el GKS permite especificar qué tipo de Prompt y de Eco se desea.

- **Disparo**. Es un dispositivo de entrada, o un conjunto de ellos asociado a un "mapeo del disparo". El operador puede utilizar el disparo para indicar instantes de tiempo significativos. En ese instante, se envía un mensaje al proceso de medida de los correspondientes dispositivos. El efecto de esto depende del modo de operación. Si es REQUEST, el proceso de medida devuelve su valor al programa de aplicación, y termina. Si es SAMPLE, el disparo se ignora, puesto que el programa de aplicación puede muestrear el valor del proceso de medida en cualquier momento. Si es EVENT, el disparo hace que se añada un elemento a la cola de eventos.

- **Reconocimiento**. Cuando se activa un disparo, GKS manda al operador un mensaje que depende de la implementación física concreta del dispositivo. El reconocimiento no es controlable por el GKS.

8.15 MANEJO DE ERRORES

En cada función del GKS se puede presentar un número finito de posibles situaciones de error que hacen que se produzca una llamada al procedimiento ERROR_HANDLING (Manejo de Error). Este procedimeinto puede definirlo el programa de aplicación, lo que le permite un control de la situación, o puede utilizarse el del GKS, el cual se limita a escribir un mensaje de error y la identificación de la función que lo produjo en un fichero de errores.

© Universidad de Las Palmas de Gran Canaria. Biblioteca Digital, 2003