UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SOFTWARE EDUCATIVO PARA LA MATERIA DE GRAFICACION POR COMPUTADORA

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

ENRIQUE RAMIREZ MONTES

Director de Tesis: Ing: Yi Tan Li

MEXICO, D. F 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Indice.

PREFACIO.

Prefacio.

Hoy en día, el uso de gráficos es de vital importancia. Las gráficas tienen bastas áreas de aplicaciones, y por lo tanto, también la Graficación por Computadora las tiene. Las gráficas por computadora se aplican a la Medicina, Botánica, Química, Ingeniería, Astronomía, Física, Arquitectura, Señales de tránsito, etc. Como se puede observar la graficación es de gran importancia en la sociedad.

El proyecto o programa GPC (Graficación por Computadora) ha sido diseñado con la idea de dar las bases y herramientas necesarias para lograr la Graficación en Computadora. Este proyecto trata de ser un complemento o un auxiliar en la enseñanza de las Graficas realizadas por Computadora. El estudiante tiene un recurso más para su aprendizaje en esta área.

El programa GPC fue realizado en Turbo C++ versión 5.00 de Borland. Este proyecto de tesis cuenta con los módulos más fundamentales o básicos de la Graficación. El programa puede manejar desde un punto hasta la iluminación de objetos tridimensionales o desde una línea recta hasta algunas técnicas en la generación de curvas fractales. Por otra parte, el programa GPC ha sido diseñado para operar en computadoras PC con cualquier característica. Por ejemplo, se puede arrancar el programa en tarjetas del cualquier tipo, con un mínimo de memoria de 640k bytes, e inclusive, el programa puede operar en diskette. El programa GPC no exige equipos caros para su operación.

Este proyecto cuenta con ayudas, las cuales dan una idea general de lo que hace cada módulo, es decir, cada punto o tópico que componen a este proyecto tiene su ayuda en específico. En algunas ocasiones se da el valor de los parámetros o variables que

son de vital importancia en la generación de algunos gráficos, e inclusive, se da la oportunidad de que el usuario modifique estos valores, para que de esta manera se logre apreciar la repercusión que tienen algunas variables en la gráficas, obteniendo así, un concepto más profundo del tópico o módulo de la graficación que se esté analizando. En la modificación de algunas de estas variables se aprecian en ocasiones cambios sorprendentes e interesantes en la gráficas generadas. Para apreciar estos cambios le invito a hacer uso del programa GPC.

Para fortalecer más el concepto de la Graficación por Computadora, este proyecto cuenta con el módulo de pseudocódigo. El pseudocódigo nos da la idea general en la que algunos algoritmos del proyecto operan. Es primordial la generalidad de los pseudocódigos mostrados en este proyecto, ya que con esto se obliga a investigar al usuario o estudiante en algunas otras fuentes. Para lograr así un aprendizaje bueno en esta área. Por otra parte, obsérvese que en ningún momento se dan los fuentes del programa GPC, por que de esta manera no se lograría el objetivo de este trabajo. En pocos casos no es suficiente la información que se dan en las ayudas y pseudocódigos solicitados debido a los recursos de memoria que se tienen en computadoras PC.

La parte más importante de este proyecto se encuentra en el módulo "Demo", el cual nos da una presentación automática de lo que el sistema o proyecto GPC puede hacer. Basta con situarnos en Demo y seleccionarlo para que el sistema nos dé una presentación de los tópicos de graficación seleccionados sin oprimir una sola tecla. La integración de ejemplos, ayudas, pseudocódigos y demo otorgan un concepto bueno sobre la área que estamos trabajando. El proyecto GPC ha sido enfocado para ser un software educativo en la materia de la Graficación Por Computadora.

El trabajo escrito que acompaña a el proyecto GPC, cuenta con cinco capítulos, los cuales hacen referencia a algunos de los tópicos de la Graficación Por Computadora. Además, este trabajo escrito cuenta con dos apéndices, el A y el B. El apéndice A nos habla sobre el Ambiente de C y la Graficación por Computadora. El apéndice B nos da una lista de las rutinas empleadas en esta tesis, así como una breve explicación de la función o tarea que desempeñan dentro del sistema , incluyéndose un diagrama

de bloques, el cual da un panorama más amplio de la operación del sistema.

Este trabajo escrito trata de abarcar el contenido de la materia Graficación por Computadora impartida en nuestra facultad. En algunos casos se cubre más de lo que se marca en el temario de esta materia, obteniéndose así, un recurso más para nuestro objetivo didático . Cualquier persona puede recurrir a este documento, para apoyarse durante el proceso del aprendizaje de esta área. Este escrito trata de ser breve y explícito en la mayoría de los puntos referenciados. Se espera que este trabajo satisfaga las necesidades de todas las personas, así como, el logro del objetivo propuesto.

CAPITULO I.

INTRODUCCION.

Breve Historia de la Graficación.

Las primeras computadoras surgieron a finales de los 50's y a principios de los 60's, fue aparente que estas máquinas iban a crecer por sus resultados, y además los medios de desplegado de la información gráfica iban a ser necesarios.

En la primavera de 1963 Ivan Sutherland, mostró un sistema de computación gráfico interactivo, conocido como Sketchpad. Este sistema utiliza una pluma linterna que apunta a posiciones en el despliegue para definir elementos de las gráficas, tales como líneas, arcos y polígonos. Una vez dibujados, estos elementos pueden ser movidos, más grandes o más pequeños, copiados o modificados utilizando lo que se conoce como arquitectura gráfica de un vector de refresco. Este sistema histórico marcó el estándar para futuros sistemas gráficos interactivos. Posteriormente, Sutherland, se unío con Dave Evans, fundando la Compañía pionera llamada Evans & Sutherland (E&S).

Desafortunadamente, el alto costo del hardware necesario para el soporte de estos sistemas como el Sketchpad, inhibío su uso y únicamente se emplearon en la investigación y en algunas universidades. A mitad de los 60's, sistemas comerciales con vector de refresco comenzaron a aparecer, pero aún costaban cientos de miles de dólares.

Con la introducción de las minicomputadoras se incrementó la necesidad para el desarrollo de sistemas de despliegues interactivos a bajo precio y eventualmente llegó en la forma de tubos de almacenamiento CRT producidos por Tektronix Inc. Esto abrió la puerta para los sistemas gráficos interactivos, lo cual, significó un serio comienzo en el desarrollo de programas para computadoras y sistemas gráficos. El tubo de almacenamiento CRT, sin embargo, no es un verdadero sistema de despliegue dinámico.

A mediados de los 60's, algunos trabajos se hicieron para el desarrollo de los sistemas de despliegue ráster, basados en pixeles, lo cual dio una alta calidad en blanco y negro y en despliegues de multisombreado, este sistema requería de grandes cantidades de memoria en las computadoras. A principios de los 70's, el desarrollo de hardware y de memoria integrada comenzó a más bajo precio y a menos costo en sistemas de almacenamiento de imágenes de pixeles, y las tecnologías ráster comenzaron a aparecer en los mercados comerciales. La introducción de estos sistemas de memoria fueron paralelos al desarrollo de los microprocesadores y de los controles lógicos, en los generadores gráficos para las tecnologías ráster y vector. Los sistemas de refresco comenzaron a encogerse en tamaño y costo. Esta tendencia continúa hasta nuestros días con los sistemas de despliegue gráficos a color.

El desarrollo del software para gráficos en computadoras se veía, mientras tanto reflejado en el desarrollo del hardware. No obstante el nuevo hardware requería de nuevos algoritmos de software, poco han cambiado en los conceptos básicos y en las técnicas de interacción desde el sistema Sketchpad de Sutherland. Sin embargo, a principios de los 70's, los investigadores de Xerox Palo Alto Research Centre (PARC) empezaron a considerar el uso de la interfaz gráfica con el sistema de cómputo, utilizando imágenes en lugar de texto para comunicarse con el usuario. Este trabajo permitió el desarrollo de sistemas tales

como Smalltalk-80, el cual es un ambiente de cómputo completo, dicho sistema utiliza una interfaz natural entre el usuario y la computadora. La historia de las gráficas por computadora ha sido restringida por el costo y por la tecnología. Los seres vivos viven en un mundo de imágenes, no es difícil imaginar los ilimitados usos de las gráficas. Inicialmente fueron empleados en el diseño y en sistemas de control, en donde los datos podían ser desplegados usando los primeros sistemas de refresco de tipo vector. Los sistemas de cómputo generados para la simulación de vuelos permitío el desarrollo de nuevas técnicas, las cuales son encontradas en fuentes comerciales de despliegue personal. En cada nuevo desarrollo en la tecnología de gráficas hay siempre una lista de aplicaciones listas para ser usadas.

Areas de Influencia.

Las aplicaciones de gráficas generadas por computadora parecen crecer cada día más. Algunas aplicaciones representan materias completas requiriendo técnicas y algoritmos complejos. Algunas otras se pueden realizar con equipo barato mientras otras requieren de sistemas de varios millones de dólares.

Las gráficas por computadora son utilizadas en la Ingeniería, Arquitectura, Química, Administración, Industria, Gobierno, Arte, Entretenimiento, Publicidad, Educación, Investigación, Medicina, etc.
La graficación por computadora se ha aplicado en las diversas áreas de la Ingeniería, por ejemplo, para el diseño de puentes y carreteras, en sistemas robóticos, así como en sistemas eléctricos y electrónicos.

Desde hace varios años, el uso más extenso de la graficación por computadora ha sido como auxiliar en el diseño. Generalmente conocido como CAD (DAC, traducido al español, Diseño Asistido por Computadora), los métodos de diseño con la ayuda de la computadora ofrecen poderosas herramientas.

Elementos de un Ambiente Gráfico Típico.

Los elementos que conforman a un ambiente gráfico son el hardware y el Software.

Hardware.

Los elementos que caracterizan al hardware son las tecnologías utilizadas de Rastreo al Azar y de Rastreo con Rastreador, los Dispositivos de E/S y las tarjetas de Procesadores Gráficos.

Tecnologías de Rastreo al Azar y Rastreo con Rastreador.

Cuando se opera con una unidad de despliegue con rastreo al azar, un CRT (tubo de rayos catódicos) tiene el haz de electrones dirigido solamente a las partes de la pantalla donde se trazará la figura, es decir, el haz de electrones recorrerá la figura definida. Los monitores de rastreo al azar dibujarán la figura, línea por línea y por esta razón también se conocen como despliegues vectoriales. Las secuencias de las líneas componentes de una figura pueden ser dadas en cualquier orden. Cuando se trabaja con figuras curvas, se hace una división de la figura, en líneas rectas mas pequeñas. Una graficadora con pluma opera en forma similar y es un ejemplo de un dispositivo de rastreo al azar de copia dura.

Las tecnologías de rastreo con rastreador hacen un recorrido en todas las partes de la pantalla, subiendo y bajando la intensidad del haz de electrones, para que de esta manera coincida con la definición de la imagen. La renovación de un sistema de rastreo con rastreador, generalmente, se lleva a una tasa de 30 a 60 cuadros (imágenes) por segundo.

Dispositivos de E/S.

Un dispositivo de E/S en un medio por el cual se puede dar u obtener información de un sistema. Los dispositivos de E/S se dividen en dispositivos de despliegue, dispositivos de copia dura y dispositivos de entrada.

Dentro de los dispositivos de despliegue tenemos los tubos de rayos catódicos de renovación, monitores CRT de color, tubos de almacenamiento con vista directa, despliegues en panel de plasma, monitores LED y LCD, dispositivos láser y monitores tridimensionales. Los dispositivos de copia dura son las impresoras y las graficadoras. Dentro de los dispositivos de entrada tenemos a los teclados, paneles de tacto, plumas linterna, tabletas de gráficas, palancas de mando, esfera de control, ratón y sistemas de voz.

Para la manipulación de algunos dispositivos de E/S existen comandos. Por ejemplo, el comando WRITE, puede colocar en un monitor un conjunto de datos.

Tarjetas y Procesadores Gráficos.

Algunos sistemas de cómputo en donde se realizan tareas de graficación cuentan con dos o más procesadores. En los sistemas de graficación, además del CPU se cuenta con un procesador de despliegue en donde se pueden realizar tareas como las de manejo de colores, manejo de coordenadas, manejo de estilos y tamaño de caracteres, estilos de líneas (continuas, punteadas, dobles, delgadas, gruesas, etc).

Entre las funciones que podemos encontrar en un procesador de despliegues es la de interactuar con los dispositivos de E/S.
En el caso de sistemas CRT de renovación, el procesador de despliegues puede ser solicitado, para renovar la pantalla con la suficiente frecuencia para eliminar la fluctuación. La definición de la imagen puede conservarse en una área de almacenamiento de renovación. En muchos sistemas el área de renovación puede asignarse a un procesador adicional, conocido como controlador de despliegues.

Las tarjetas o adaptadores gráficos que son instalados en una microcomputadora determinan las capacidades gráficas máximas en un sistema. En general, hay algunos tipos principales de adaptadores gráficos que son empleados en computadoras personales. Estos adaptadores gráficos son el VGA, EGA, MCGA y el CGA. En todos estos adaptadores gráficos pueden generarse una diver-

sidad de selecciones de los modos gráficos con una clasificación de 2 hasta 256 colores, así como la variación de la resolución de un despliegue de 320x200 a 640x480.

Software.

Los elementos característicos del software en un sistema de graficación son el Kérnel Gráfico, Las Funciones Gráficas Avanzadas y Los Estándares Gráficos.

Kérnel Gráfico.

El Sistema Estándar del Kérnel Gráfico (GKS) consiste de un conjunto de 209 funciones gráficas ordenadas en una estructura de 3 niveles de salidas (del 0 al 2) y de 3 niveles de entrada (de la 'a' a la 'c'). Una implementación GKS puede ajustarse a cualquier nivel de entrada y a cualquier nivel de salida, la combinación de sus funciones definen un nivel válido de GKS.
Una implementación GKS del nivel 2c tiene todas las funciones definidas.

Las funciones incluyen a las del despliegue e interacción gráfico, además de aquellas que manejan casi todo el ambiente de hardware. Cada definición de función exactamente especifica la acción y el estado final del sistema después de su uso.

Funciones Gráficas Avanzadas.

Las funciones avanzadas permiten realizar funciones más complejas en los sistemas gráficos. Entre las más importantes tenemos la de medios tonos, estilos de líneas, copia de páginas, dibujo de gráficas en páginas ocultas o escondidas y gráficas en bloques.

Los medios tonos se utilizan en el rellenado de áreas. Se pueden crear diversos patrones para las operaciones de rellenado.
Los estilos de líneas son logrados por estas funciones gráficas avanzadas.
En adaptadores como el VGA o EGA se dispone de más de una página gráfica, algunos programas con animación en 3D operan

moviendo la imagen de una página a otra, a este movimiento se le llama copiado de página.

El dibujo en páginas escondidas es posible. En tarjetas como la VGA o la EGA, debido a que ofrecen múltiples páginas. Se puede dibujar en páginas que no estén desplegadas en el monitor con esta función gráfica avanzada.

Las gráficas en bloques son usadas en animaciones para mover títulos a través de la pantalla.

Estándares Gráficos.

A mitad de los 70's, algunas organizaciones estándares nacionales e internacionales comenzaron una investigación para un sistema gráfico Estándar. En ese tiempo y actualmente, los métodos y funciones usados para el dibujo en varios sistemas gráficos eran muy diferentes. Este trabajo fue, y aún es coordinado por la International Standards Organization (ISO) en cooperación con las organizaciones nacionales de estandarización de los países participantes, los cuales incluyen a el West German Deutsches Institute Fur Normungs (DIN), al British Standards Institute (BSI) y al American National Stan dards Institute (ANSI).

Eventualmente, sin embargo, ISO decidío concentrarse en el sistema gráfico Kérnel (GKS), puesto al frente por la DIN y sujetado a una revisión técnica internacional. GKS pasó por varias etapas de modificación antes de que alcanzara el status actual que tiene como estándar internacional.

CAPITULO II.

PRIMITIVAS GRAFICAS.

Generación de un Punto a Nivel Memoria de Video.

El desplegado de puntos o pixeles en algunas tarjetas se logra mediante la estrategia Multiplano por Pixel (mpp) o bien por la estrategia Multibit por Pixel (mbp).

En la figura 1 se muestra la técnica Multiplano por Pixel, en la cual se utilizan cuatro planos para almacenar una figura o una gráfica. Cada plano tiene una porción de la figura, de acuerdo a los colores primarios que conforman a la gráfica inicial. Los colores primarios usados son el rojo, verde, azul y nivel de intensidad.

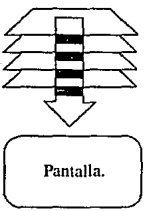


Fig 1. Mapa de memoria Multiplano por Pixel.

Simultáneamente, el controlador de video lee los bits correspondientes de los cuatro planos, y usa la suma para determinar cual de los colores disponibles debe de ser desplegado en esa localidad de la pantalla. Un plano de bits puede exhibir un 0 o un 1 en una localidad o punto, por lo que el número de colores disponibles será de 16 (de 0 a 15). Debido a que el máximo número representado en binario por la suma de los cuatro planos será 16.

En aplicaciones avanzadas se pueden modificar los tonos de los colores desplegados, mediante la modificación de las paletas. Por ejemplo, en la iluminación de objetos se pueden modificar las intensidades de los colores primarios utilizados para generar, de esta manera, sombras, medios tonos o escala de grises. Para que de esta forma se tenga un efecto de iluminación muy real.

Algunas otras tarjetas utilizan la estrategia llamada Multibit por Pixel (mbp), tal y como se puede observar en la figura 2. Solamente un plano de bits es utilizado, comúnmente llamado mapa de bits. Para gráficos de 320x200 con 4 colores, 2 bits por pixel son requeridos. Esto es por que 2 bits pueden expresar 4 diferen-

tes colores, tal y como lo son 00, 01, 10, 11. Un byte de 8 bits controla 4 pixeles en la memoria de video. Para un modo de 640x200 con 2 colores, un bit por pixel es requerido porque un simple bit puede expresar 2 colores diferentes 0 ó 1 (apagado o encendido).

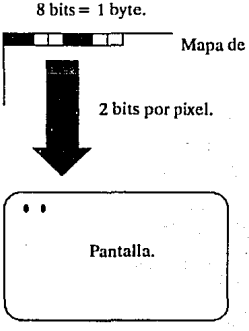


Fig 2. Mapa de memoria Multibit por Pixel.

Por ejemplo, en máquinas de 640k bytes en RAM, la memoria de video se encuentra en seguida de los 640K bytes. Generalmente 256k bytes de memoria son utilizados para fines de video. Si dividimos los 256k bytes de la memoria de video disponible en 4 bloques de 64k bytes obtenemos los 4 multiplanos de las diferentes páginas, en caso de utilizar la estrategia de Multiplano por Pixel.

Generadores Por DDA, Parametrización y Bresenham para:

Líneas.

La generación de una línea recta por el método del Analizador Diferencial Digital (DDA) se apoya fundamentalmente en la ecuación:

Deltay = m*Deltax.

En otras palabras a todo incremento de "x", le corresponde un incremento de "y".
Para entender mejor este método analicemos una línea recta con pendiente positiva y menor a 1. Si suponemos que el punto P1 está a la izquierda del punto P2, y si damos a "x" incrementos unitarios, "y" podría crecer aplicando el criterio de la ecuación 1, de esta manera tenemos:

$y_{i+1} = y_i + m \Rightarrow x_{i+1} = x_i + 1.$

Ahora si la línea recta tiene una pendiente positiva y mayor a 1, y nuevamente, si el punto P1 está a la izquierda del punto P2, la variable a incrementarse sería "y", porque de esta manera se podría manejar una línea recta con pendiente positiva y mayor a 1, resumiendo lo anterior tenemos:

$x_{i+1} = x_i + 1/m \Rightarrow y_{i+1} = y_i + 1.$

Este análisis podría generalizarse para otros casos.

El método paramétrico para líneas rectas se apoya en la ecuación de la recta para dos puntos la cual es:

$y = m*(x-x_1) + y_1.$

Después de una serie de operaciones tenemos que:

$x = u*(x_2-x_1) + x_1.$
$y = u*(y_2-y_1) + y_1.$

Donde u va desde 0 hasta 1. Obteniendo a "x" y a "y" de una manera paramétrica.

Las líneas Bresenham son las más eficientes debido a que su generación se apoya en el uso exclusivo de aritmética entera. Otros métodos hacen uso de números reales, además de realizar redondeos, lo cual utiliza demasiado tiempo. Por otra parte, en el redondeo de números se tienen diversos criterios. Los cuales, si observamos una línea recta a detalle, es decir, muy cercanamente, podría tener discontinuidades en su trazado. En caso opuesto, la línea recta podría tener información de sobra en ciertos pixeles, lo cual consume tiempo.

Las líneas Bresenham deciden un pixel u otro en el trazado, por la dirección que marca el parámetro P_i. El parámetro P_i es el eje principal en la generación de las líneas Bresenham. P_i estará siempre en función del valor del parámetro anterior P_i, es decir:

$P_{i+1} = f(P_i)$.

La deducción del algoritmo para líneas rectas por el método de Bresenham se obtiene de la figura 3. La cual es una línea recta con pendiente positiva y menor a uno. Posteriormente se generaliza este algoritmo para rectas con otras características. Cuando se realiza la generalización del algoritmo, por ejemplo, se toma el punto P_1 como el punto P_2, o el punto P_2 como el punto P_1, o se intercambian los valores de x por los de y, o los de y por los de x, solo por mencionar algunos ejemplos.

Manejando la diferencia d1 y d2, mostrados en la figura 3, obtenemos P_i, es decir $P_i = d_1 - d_2$. Haciendo una extensión obtenemos un parámetro P_{i+1} en función de P_i, los mecanismos empleados, utilizan solo aritmética entera.

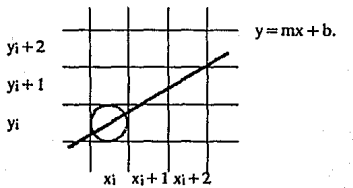


Fig 3. Sección de la retícula de la pantalla.

Las ecuaciones obtenidas después de una serie de cálculos son las siguientes:

$P_{i+1} = P_i + 2*Dy - 2*Dx*(y_{i+1} - y_i)$ ... 1.
$P_1 = 2*Dy - Dx.$ ... 2.

De donde Dx y Dy son delta de "x" y delta de "y" respectivamente.
Por ejemplo, si manejemos el parámetro P_i de una manera gráfica, para la figura 3, suponiendo que la pendiente es positiva y menor que uno, en el punto X_{i+1}, si $P_i < 0$, implica que $d_1 < d_2$, y por lo tanto se tomaría el punto $(x_i + 1, y_i)$, en caso contrario se tomaría el punto $(x_i + 1, y_i + 1)$ por que $d_1 = d_2$.
Si observamos el funcionamiento de este algoritmo únicamente opera para líneas rectas con pendiente positiva menor a 1, para una pendiente mayor a 1, "x" se manejará como "y" y "y" como la variable "x".

Las ecuaciones que rigen el funcionamiento de este algoritmo son las ecuaciones 1 y 2. Como se puede observar estas ecuaciones manejan aritmética entera, lo cual hacen un algoritmo mas rápido y por lo tanto mas eficiente.

Círculos.

La generación de los círculos DDA, en muchos libros, son obtenidas directamente de la fórmula general del círculo la cual es:

$$x^2+y^2=r^2.$$

Posteriormente de haber calculado "x" y "y", se hace un co rrimiento hacia el centro del círculo, es decir, se suma el punto central $P_c(x_c,y_c)$ a "x" y a "y" obtenidos.

Es suficiente, con calcular una octava parte del círculo, debido a que un círculo ofrece simetría sobre los ejes cartesianos, es decir ofrece simetría por el eje "x" y por el eje "y".

Los círculos paramétricos son generados por las ecuaciones paramétricas del círculo, las cuales son:

$$x = r*\cos(\theta).$$
$$y = r*\text{sen}(\theta).$$

El incremento de θ debe de ser de 1/r, con este incremento se asegura el trazo del círculo sin discontinuidades. Nuevamente, el cálculo del círculo se puede hacer en una octava parte, debido a la simetría del círculo.

Los círculos Bresenham operan de manera semejante a las líneas. De acuerdo a la figura que en seguida se muestra:

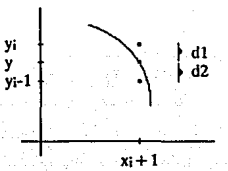


Fig 4. Diferencia de coordenadas en pixeles a elegir.

Se resta d_1 y d_2 para obtener P_i, luego se hace una extensión de P_i para obtener P_{i+1}, de esta forma se tiene que:

$$P_{i+1} = P_i + 4*x_i + 6 + 2*(y_{i+1}^2 - y_i^2) - 2*(y_{i+1} + y_i)$$
$$P_1 = 3 - 2*r$$

Para este método es conveniente que el punto inicial sea (0,r) = (x,y), y que el punto final sea cuando x = y, de esta manera aseguramos el cálculo, por lo menos, de una octava parte del círculo, con lo cual es suficiente, para trazar el círculo completo, de acuerdo a la simetría antes mencionada.

CAPITULO III.

GRAFICACION EN 2D.

Transformaciones 2D

En la graficación por computadora es importante el manejo de las figuras de trabajo, como lo son círculos, líneas, polígonos, elipses, entre otros. Entre las transformaciones fundamentales tenemos el desplazamiento, el escalamiento, la rotación y el espejo. Con estas herramientas se pueden hacer diseños interesantes de máquinas, aviones, casas, etc.

Coordenadas Homogéneas.

Las transformaciones mencionadas para 2D, generalmente, se realizan con matrices debido a la facilidad que muestran en su manejo. En un sistema de traslación, las ecuaciones son:

$x' = a*x + c*y + m.$
$y' = b*x + d*y + n.$

donde a,b,c,d,m,n son constantes, p(x,y) es el punto a trasladar y p'(x',y') es el punto resultante. Desafortunadamente, no sería posible introducir las constantes de traslación m,n a una matriz general de transformación de 2x2, no habría espacio. Con la introducción de las coordenadas homogéneas es posible introducir estas constantes de traslación, con lo cual tenemos lo siguiente:

$$[x'] \quad [a\ b\ 0]$$
$$[y'] = [x,y,1] * [c\ d\ 0]$$
$$[1] \quad [m\ n\ 1]$$

Las coordenadas homogéneas facilitan y agilizan las transformaciones en las figuras, diseños o cuerpos de trabajo.

Traslación.

La traslación es una transformación fundamental empleada para cambiar de posición a un cuerpo. Una traslación en 2D puede tener componentes sobre el eje "x" o sobre el eje "y", o bien sobre ambos. Las ecuaciones que representan la traslación son:

$$x' = x + tx.$$
$$y' = y + ty.$$

Al par (tx,ty) se le conoce como vector de cambio o vector de traslación. La traslación en coordenadas homogéneas tendría la siguiente forma:

$$[x'] \quad [1\ 0\ 0]$$
$$[y'] = [x,y,1] * [0\ 1\ 0]$$
$$[1] \quad [tx\ ty\ 1]$$

Escalamiento con Respecto a un Punto.

El escalamiento es una transformación que nos amplifica o nos reduce una figura. El escalamiento debe tener un punto de referencia, este punto puede estar en cualquier lugar de nuestro sistema coordenado. Por ejemplo, si el punto de referencia está sobre un vértice del polígono a amplificar, el polígono incrementa su tamaño y tiende a moverse, excepto, el vértice que fue

tomado como referencia. Las ecuaciones que representan el escalamiento de un punto con respecto a un punto son:

$$x' = x*sx + (1-sx)*xf.$$
$$y' = y*sy + (1-sy)*yf.$$

De donde sx y sy son el factor de amplificación y xf, yf son el punto de referencia.
La escalación tiene la siguiente forma en coordenadas homogéneas.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [x,y,1] * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ (1-sx)*xf & (1-sy)*yf & 1 \end{bmatrix}$$

En realidad, para llevar a cabo la escalación con respecto a un punto de referencia, lo primero que se hace es llevar ese punto de referencia al origen, después se realiza la amplificación, y posteriormente el punto de referencia se regresa a su lugar de origen.

La forma simplificada de las transformaciones utilizadas, para la realización de la escalación con respecto a un punto de refe rencia, es la siguiente:

$$T_R = T(-xf,-yf) * S(sx,sy) * T(xf,yf).$$

De donde:
T_R: Es la transformación resultante.
T: Es la transformación de traslación.
S: Es la transformación de escalación con respecto al origen.
A esta notación simplificada también se le conoce con el nombre de Concatenación de Transformaciones, la cual en breve se verá.

Rotación con Respecto a un Punto.

La rotación con respecto a un punto de referencia es una transformación, que gira un cuerpo con respecto a un punto de referencia elegido.
Las ecuaciones que representan la rotación de un punto son las siguientes:

$x' = x_R + (x-x_R)*\cos(\theta)-(y-y_R)*\text{sen}(\theta)$.
$y' = y_R + (y-y_R)*\cos(\theta)+(x-x_R)*\text{sen}(\theta)$.

de donde x_R, y_R es el punto de referencia con respecto al cual se va a rotar. θ es el ángulo de rotación.
La representación de estas ecuaciones en coordenadas homogéneas son:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [x,y,1]* \begin{bmatrix} \cos(\theta) & \text{sen}(\theta) & 0 \\ -\text{sen}(\theta) & \cos(\theta) & 0 \\ (1-\cos(\theta))*x_R + y_R*\text{sen}(\theta) & (1-\cos(\theta))*y_R - x_R*\text{sen}(\theta) & 1 \end{bmatrix}$$

Para realizar la rotación con respecto a un punto de referencia, lo primero que se hace, es trasladar el punto de referencia al origen, posteriormente se procede a girar o a rotar la figura, el cuerpo o punto que estemos trabajando. Después de esto se traslada el punto de referencia a su lugar de origen.

La forma simplificada de las transformaciones utilizadas, para la realización de la rotación con respecto a un punto de referencia, es la siguiente:

$T_R = T(-x_R,-y_R)*R(\Theta)*T(x_R,y_R)$.

De donde:
T_R: Es la transformación resultante.
T: Es la transformación de traslación.
R: Es la transformación de rotación con respecto al origen.
A esta notación simplificada también se le conoce con el nombre de Concatenación de Transformaciones, la cual en breve se verá.

Espejo.

El espejo es una transformación que produce la reflexión de una figura o de un punto con respecto a un eje arbitrario. Este eje arbitrario ser paralelo a cualquiera de los ejes cartesianos. El

espejo se puede hacer sobre el eje x, y, sobre el origen o bien sobre un eje arbitrario. Cuando se hace un espejo con respecto al origen, en realidad se hace un espejo simultáneamente con respecto al eje "x" y "y".

Por ejemplo, si queremos hacer un espejo con respecto al eje "x" las ecuaciones serían.

$x' = x.$
$y' = -y.$

En coordenadas homogéneas sería:

$$\begin{matrix}[x'] \\ [y'] = [x,y,1] * \\ [1]\end{matrix} \begin{matrix}[1 & 0 & 0] \\ [0 & -1 & 0] \\ [0 & 0 & 1]\end{matrix}$$

Para realizar la reflexión con respecto al eje "y", o al origen se utilizan otras ecuaciones.

Concatenación de Transformaciones.

La concatenación de transformaciones también se conoce como transformaciones compuestas, eslabonamiento o composición de matrices, debido a que las transformaciones ofrecen propiedades interesantes. Todas estas propiedades se logran mediante el producto de las matrices de transformación.

Por ejemplo, si queremos trasladar un punto, una distancia d_1 y posteriormente, si queremos trasladar este punto una distancia d_2 la ecuación podría escribirse como:

$$T(dx_1,dy_1) * T(dx_2,dy_2) = T(dx_1 + dx_2, dy_1 + dy_2).$$

Porque:

$$\begin{matrix}[1 & 0 & 0] \\ [0 & 1 & 0] \\ [dx_1 & dy_1 & 1]\end{matrix} * \begin{matrix}[1 & 0 & 0] \\ [0 & 1 & 0] \\ [dx_2 & dy_2 & 1]\end{matrix} = \begin{matrix}[1 & 0 & 0] \\ [0 & 1 & 0] \\ [dx_1 + dx_2 & dy_1 + dy_2 & 1]\end{matrix}$$

Relación entre Sistemas Coordenados.

En el contexto de un sistema gráfico hay tres sistemas coordenados de interés, el mundial o coordenadas de usuario, un pseudoespacio en coordenadas normalizadas y coordenadas de máquina. Las coordenadas mundiales son aquellas, en la cual la aplicación es dada o especificada, por ejemplo, la ubicación de una casita en una montaña, un árbol, una silla, etc. Son generalmente dadas en números de punto flotante. Las coordenadas de la máquina son entendidas por la fuente gráfica en particular como lo son pixeles, puntos direccionables, pulgadas, centímetros, etc. Estas pueden ser especificadas por números enteros. El espacio normalizado es un pseudoespacio usado para obtener la independencia de la fuente. Las coordenadas normalizadas son dadas en números de punto flotante.

La transformación mundial a coordenadas normalizadas y de coordenadas normalizadas a coordenadas de la máquina involucran simplemente traslación y escalación.

En el siguiente tema se describe el cambio del sistema coordenado mundial a coordenadas de máquina. Mediante el uso de una ventana para el sistema coordenado mundial y de un puerto para el sistema coordenado de la máquina.

La ventana se utiliza en el sistema coordenado mundial para recortar el área que se quiere desplegar, y el puerto se utiliza en el sistema coordenado de la máquina, para ubicar la gráfica de la ventana antes mencionada.

Transformación Ventana-Puerto.

La frontera rectangular del área de interés en nuestro sistema de coordenadas mundiales es llamada ventana. El correspondiente cambio en el área de la pantalla es conocido como puerto. En otras palabras, tal y como se mencionó anteriormente, se utiliza la ventana para definir que es lo que se quiere desplegar, y utilizamos el puerto para especificar sobre la pantalla el lugar donde queremos colocar la imagen. Para entender mejor este concepto veamos la siguiente figura:

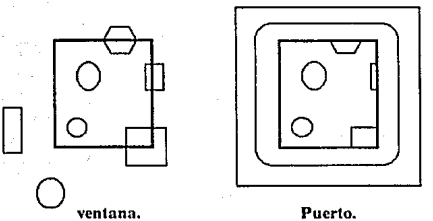


Fig 5. Transformación Ventana-Puerto.

Sí se realiza la aplicación, el programa debe de checar la visibilidad antes de que se manden los datos a la pantalla, las coordenadas de los puntos pueden quedar fuera de la pantalla. Por lo tanto, se requiere de un recorte antes de que se dibujen dichos datos, además de eliminar las porciones de la gráfica que están fuera de la ventana durante el recorte.

Después de haber efectuado el recorte de la figura dentro de los límites de la ventana, el siguiente paso es mapear la imagen a la porción de la pantalla o puerto. Para realizar la transformación Ventana-Puerto utilicemos la figura 6 que en seguida se muestra:

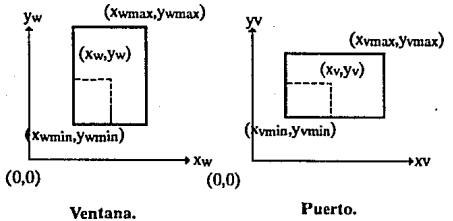


Fig 6. Mapeo Ventana-Puerto.

El punto (x_w,y_w) mapéa al punto (x_v,y_v), de tal forma, se retiene una posición relativa. La transformación de coordenadas ventana a coordenadas de puerto es equivalente a trasladar (x_w,y_w) una distancia (x_{wmin},y_{wmin}), después se realiza un escalamiento, y posteriormente, se realiza la traslación de un sistema coordenado mundial a el nuevo sistema coordenado, en este caso es el puerto.

La forma simplificada de las transformaciones sería:

$$T_R = T(-x_{wmin},-y_{wmin}) * S(s_1,s_2) * T(x_{vmin},y_{vmin}).$$

De donde:
T: es la transformación de traslación.
S: es la transformación de escalamiento con respecto al origen.
T_R: es la transformación resultante.
Obteniendo de esta manera:

$$x_v = s_1 * (x_w - x_{wmin}) + x_{vmin}.$$
$$y_v = s_2 * (y_w - y_{wmin}) + y_{vmin}.$$

Para preservar las proporciones, los factores de escalación s_1 y s_2 deben de ser iguales a:

$s1 = (x_{vmax} - x_{vmin})/(x_{wmax} - x_{wmin})$.
$s2 = (y_{vmax} - y_{vmin})/(y_{wmax} - y_{wmin})$.

Recortes.

El recorte es el proceso en el cual se extrae una porción de datos de una ventana o de una región.

Los algoritmos de recorte pueden ser implementados en hardware o software. Cuando son implementados en software son más lentos.

Recorte Sobre Regiones Rectangulares.

La figura 7 muestra una escena para una ventana de recorte regular rectangular. Una ventana de recorte regular rectangular está definida si los lados están alineados al despliegue de la máquina. El propósito de un algoritmo de recorte es determinar cuales líneas, puntos o porciones de líneas son retenidas para el despliegue. todo lo demás es desechado.

En el recorte de gran cantidad de líneas o de puntos, la eficiencia de los algoritmos de recorte es de interés particular. En muchos casos gran cantidad de puntos o de líneas se encuentran en el interior o en el exterior de la ventana de recorte. No obstante es importante aceptar rápidamente una línea como ab o un punto como p o desechar una línea como ij o un punto como q, tal y como se puede apreciar en la figura 7.

Un punto está en el interior de la ventana de recorte si:

$x_L <= x <= x_R$ y $y_B <= y <= y_T$

De donde:

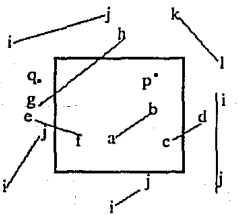
x_L, x_R: son los límites izquierdo y derecho de la ventana, respectivamente.

y_B, y_T: son los límites inferiores y superiores de la ventana, respectivamente.

Las líneas son interiores a la ventana de recorte y por lo tanto visibles si ambos puntos terminales están en el interior de la ventana, por ejemplo, la línea ab. Sin embargo si los puntos finales de la línea son exteriores a la ventana, la línea no es necesariamente completamente exterior a la ventana. Si ambos puntos

terminales de la línea se encuentran completamente a la derecha, o a la izquierda, o arriba o completamente debajo de la ventana, la línea es completamente exterior a la ventana y por lo tanto invisible. Esta prueba eliminaría a la línea ij, y aceptaría a la línea gh, la cual es parcialmente visible o a la línea kl, la cual es totalmente invisible.

cima



fondo **izquierda** **derecha**

Fig 7. Ventana de recorte bidimensional.

Algunos algoritmos para probar la visibilidad de la líneas se apoyan en la técnica de Dan Cohen y de Ivan Sutherland. Esta técnica utiliza dos códigos de 4 bits para indicar cual de las 9 regiones contienen el punto final de la línea. Los códigos de 4 bits son mostrados en la figura 8.

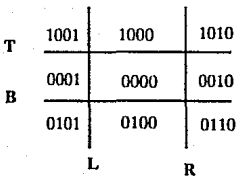


Fig 8. Mapa de códigos para los extremos de una línea.

- El primer bit se pone en 1, si el extremo de la línea está a la izquierda de la ventana.
- El segundo bit se pone en 1, si el extremo de la línea está a la derecha de la ventana.
- El tercer bit se pone en 1, si el extremo de la línea está a la debajo de la ventana.
- El cuarto bit se pone en 1, si el extremo de la línea está a la arriba de la ventana.

De otra manera, los códigos de los puntos finales son cero, sí la línea se encuentra dentro de la ventana.

El algoritmo de la subdivisión de la línea de Sutherland Cohen divide a la línea con un vértice de la ventana. En contraste no checa para ver si el punto de intercepción está dentro de la ventana, pero trata de aceptar o rechazar los dos puntos resultantes usando los códigos de los puntos finales de acuerdo a la figura 9. La línea P_1P_2, inmediatamente muestra una dificultad con esta simple técnica. Si P_1P_2 es recortado contra el segmento izquierdo de la ventana, los dos nuevos segmentos son P_1P_1' y $P_1'P_2$. Los códigos finales para estos segmentos indican que ambas pueden

estar parcialmente visibles. Por lo que ninguna puede ser desechada como invisible o aceptada como visible. La llave para el algoritmo Sutherland-Cohen es siempre saber que alguno de los puntos finales está afuera de la ventana. Por lo que el segmento desde este punto final a la intercepción puede ser desechado como invisible. El algoritmo entonces procede con el resto de la línea. En efecto esto sustituye el punto final original.

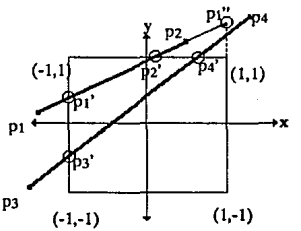


Fig 9. Recorte paramétrico en dos dimensiones.

El algoritmo de la subdivisión del punto medio evita un cálculo directo, debido a una búsqueda binaria de la intercepción por la división de la línea en su punto medio. El algoritmo el cual es un caso especial del algoritmo de Sutherland-Cohen, fue propuesto por Sproull y Sutherland para hardware. La implementación en hardware es más lenta que usando el cálculo directo de la intercepción con el límite de la ventana. La implementación en hardware es rápida y eficiente por la arquitectura en paralelo que puede ser usada y la suma o división por 2 es muy rápida.

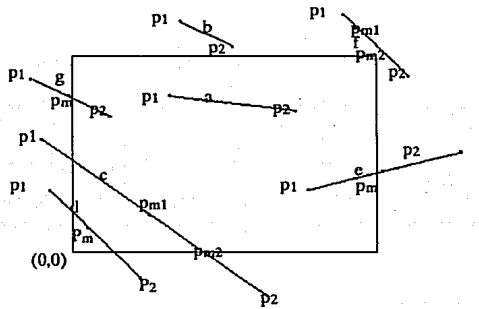


Fig 10. Subdivisión en el punto medio.

El algoritmo usa el código del punto final de la línea y pruebas asociadas para identificar inmediatamente líneas totalmente visibles como la línea a en la figura 10 y líneas trivialmente invisibles como la línea b. Las líneas de la c a la g de la figura 10 son subdivididas a partes iguales. Estas pruebas son aplicadas a cada mitad hasta que la intercepción con un lado de la ventana es encontrada o hasta que la longitud del segmento dividido es infinitesimal(un pixel por ejemplo).

Recorte Sobre Regiones Poligonales Mayores a Cuatro Lados.

Para muchos objetivos o propósitos la ventana de recorte no va a ser siempre rectangular. La ventana de recorte, por ejemplo, puede estar rotada o puede tener más de cuatro lados, inclusive existen algoritmos que trabajan con ventanas de recorte con tres lados. Cyrus y Beck trabajaron sobre un algoritmo para ventanas no rectangulares y con cualquier número de lados. Este algoritmo para ver si un punto está adentro o afuera de la ventana utiliza el

producto "cruz". Además este algoritmo para encontrar la intercepción de la recta con la ventana, utiliza la ecuación paramétrica de la recta.

Por otra parte, Sutherland y Hodgman desarrollaron un algoritmo interactivo, el cual evita la generación y almacenamiento de los polígonos intermedios generados. Tan pronto como un vértice es cortado contra una cara de la ventana, este algoritmo se llama recursivamente para recortarse contra la siguiente cara.
La idea general del recorte de Sutherland y Hodgman es:

- Desde i = 1 hasta n vértices del polígono.
- Si p_{i-1} y p_i están dentro de la ventana se salva p_i.
- Si p_{i-1} y p_i están afuera de la ventana, no se salva nada.
- Si p_{i-1} está adentro y si p_i está afuera, calcular I y salvar I.
- Si p_{i-1} está afuera y si p_i está adentro, calcular I y salvar I, p_i.
- Fin Desde i.

De donde:
I: es la intercepción obtenida del lado del polígono (p_i-p_{i-1}) contra el lado de la ventana.
Es muy importante que la ventana de recorte sea convexa. Además los vértices deben de ser dados en sentido manecillas del reloj. Un ejemplo de como opera el algoritmo de recorte gráficamente lo podemos ver en la siguiente figura:

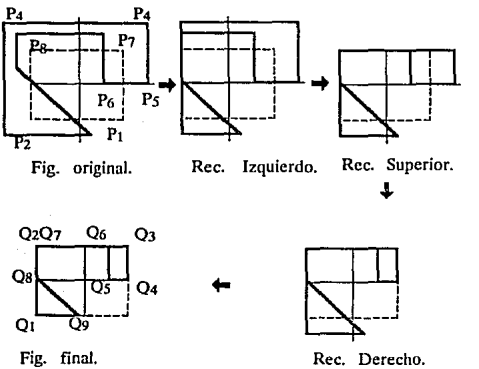


Fig 11. Representación esquemática de un recorte.

El algoritmo de Weller-Atherton es capaz de recortar polígonos cóncavos con hoyos interiores. El algoritmo recorta al polígono y a la ventana, mediante una lista de vértices. Los límites exteriores de los polígonos son descritos en sentido manecillas de reloj y los límites interiores u hoyos en sentido antimanecillas de reloj.

Diseño de Curvas.

Las primitivas de salida más básicas entre los sistemas de graficación son los puntos, las líneas y los polígonos. Estos son suficientes para la realización de curvas o superficies. Para lograr curvas se requiere de gran cantidad de información.

En las características importantes de las curvas tenemos: la propiedad de la convexividad, la invarianza en la traslación, la invarianza en la rotación y escalación, la curvatura, las condiciones finales en curvas cerradas, entre otras.

La convexividad en las curvas consiste en que si los puntos de control forman un polígono convexo, la curva generada estará dentro del área que ocupa el polígono o los puntos de control.

La invarianza en la traslación no cambia el aspecto de la curva definida, cuando se realiza la traslación de los vértices o puntos de control.

La invarianza en la rotación y escalación no cambia el aspecto de la curva definida cuando se realiza la rotación y/o la escalación de los vértices o puntos de control.

La curvatura en un punto P, en una curva definida paramétricamente Q(u), el círculo que tiene la primera y segunda derivada iguales a las de la curva es llamado círculo osculatorio. El centro y el radio de este círculo son llamados centro de curvatura c(u) y radio de curvatura r(u), respectivamente. Las curvatura c(u) en este punto es recíproca a 1/r(u). Por lo que si el círculo osculatorio tiene un radio grande, la curvatura es pequeña, tal y como nuestra intuición nos los dice. Para entender mejor este concepto, obsérvese la siguiente figura.

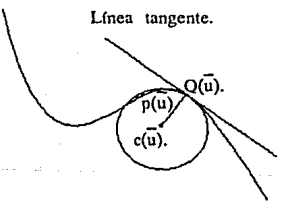


Fig 12. Círculo Osculatorio.

El vector de curvatura K(u) tiene una magnitud igual a la curvatura y apunta de P hacia el centro de curvatura.

$$Q^{(1)}(u)xQ^{(2)}(U)/|Q^{(1)}(u)|^3.$$

tienen magnitud igual a la curvatura. Sin embargo, este vector es perpendicular al plano que contiene el círculo osculatorio (plano osculatorio). El producto cruz con:

$$Q^{(1)}(u)/|Q^{(1)}(u)|.$$

Resulta en un vector de la misma longitud ubicado dentro del plano osculatorio, el cual es un vector de curvatura.

$$K(u)=(Q^{(1)}(u)xQ^{(2)}(u))xQ^{(1)}(u)/|Q^{(1)}(U)|^4.$$

De lo anterior se induce que:

- Si la segunda derivada del vector es cero, entonces la curvatura es cero.
- Si la primera y segunda derivada del vector son diferentes de cero y además son linealmente dependientes (colineales), entonces la curvatura es cero.
- Si el primer y segundo vector derivativo es linealmente independiente (no colineales), entonces la curvatura es diferente de cero.

Las curvas generalmente son abiertas, en muy pocas ocasiones son cerradas. En una curva cerrada los puntos extremos se juntan.

Hermite.

El método de Hermite es un algoritmo que obtiene curvas a partir del conjunto de datos "X,Y " de donde "X,Y", definen el contorno de la curva a ser dibujada. Este contorno debe de ser una función, debido a que el algoritmo utiliza derivadas.

La derivada utiliza métodos numéricos. Para la realización de una derivada en métodos numéricos se requiere de un espaciamiento continuo o constante en la variable X. Para lograr el espaciamiento continuo se hace una división constante o continua en el contorno de la curva. Después de haber obtenido a f'(X) se procede con lo siguiente:

Paso 1:
Para $i = 0,1,...,n$ seguir los pasos 2 y 3
Paso 2: Tomar $Z_{2*i} = X_i$;
$Z_{2*i+1} = X_i$;
$Q_{2*i,0} = f(X_i)$;
$Q_{2*i+1,0} = f(X_i)$;
$Q_{2*i+1,1} = f'(X_i)$;
Paso 3: Si $i <> 0$ entonces tomar
$Q_{2*i,1} = (Q_{2*i,0} - Q_{2*1-1,0})/(Z_{2*i} - Z_{2*i-1})$
Paso 4: Para $i = 2,3,...,2*n+1$
Para $j = 2,3,...,i$ Tomar $Q_{i,j} = (Q_{i,j-1} - Q_{i-1,j-1})/(Z_i - Z_{i-j})$
Paso 5 : Salida $(Q_{0,0}, Q_{1,1}, ..., Q_{2*n+1,2*n+1})$.

Del paso 1 al 3 se inicializa la columna 0,1 de la matriz Q utilizada, también, se inicializa el vector Z. Posteriormente en el paso 4, se calculan los elementos de una matriz triangular inferior, para obtener de esta manera los coeficientes del polinomio interpolante de Hermite. Estos coeficientes se toman de la matriz Q para cuando "i" igual a "j".
De donde el polinomio interpolante de Hermite es el siguiente:

$$H(X) = Q_{0,0} + Q_{1,1}(X-X_0) + Q_{2,2}(X-X_0)^2 + Q_{3,3}(X-X_0)^2(X-X_1) + Q_{4,4}(X-X_0)^2(X-X_1)^2 + ... + Q_{2*n+1,2*n+1}(X-X_0)^2(X-X_1)^2 ....(X-X_{n-1})(X-X_n).$$

Bezier.

Una curva de Bezier es determinada por la definición del contorno, el cual puede ser un polígono. Este polígono puede ser o no ser una función.
Algunas de las propiedades de las curvas de Bezier son:

- Las funciones de generación son reales.
- El grado del polígono que define la curva es menor en uno.
- La obtención de los puntos del polígono tienen una secuencia y la generación de la curva también la sigue.
- El primer y el último punto de la curva coincide con el primer y el último punto del polígono respectivamente.

- Si el polígono definido es convexo, la curva se encontrará en la parte interna del mismo. El polígono puede ser abierto y cerrado, si el polígono es abierto se traza una línea imaginaria del punto inicial al punto final y de esta manera se sabe si un polígono es cóncavo o convexo.

La fórmula para obtener una curva a partir de los puntos del polígono o puntos de control es:

$$P(t) = \Sigma B_i * J_{n,i}(t) \quad 0 <= t <= 1$$

$J_{n,i}(t)$ es la función básica de orden n. De donde n es el grado de la función básica y por lo tanto es uno menor que el número de puntos que componen al polígono. Generalmente el número de puntos del polígono va de 0 a n, obteniendo así n+1 puntos de control y por lo tanto el orden de la función básica es n.

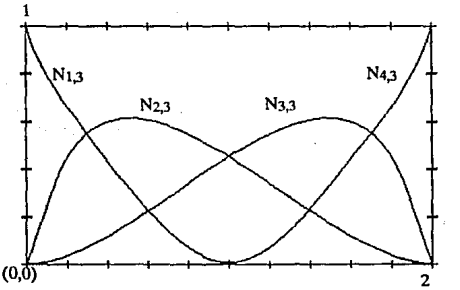


Fig 13. Funciones básicas para Bspline.

Cuando se realiza una curva para un polígono cerrado, la curva también es cerrada. Es decir, el punto inicial y el punto final van a ser los mismos, en la curva y en el polígono.

Bspline.

Para definir una curva, se necesita nuevamente, un polígono, el cual va a marcar el contorno de la curva. La fórmula general para obtener una curva por el método Bspline es:

$$p(t) = \Sigma B_i * N_{i,k}(t) \quad T_{min} <= t <= T_{max} \quad 2 <= k <= n+1$$

de donde:
B_i : son las coordenadas "x" y "y" del polígono que definen la curva.
$N_{i,k}$: son la funciones normalizadas Bspline.
Para calcular la función normalizada Bspline de orden k (realmente de orden k-1) $N_{i,k}(t)$ se obtiene por fórmulas de recursión, las cuales son:

$$N_{i,1}(t) = \begin{cases} 1 & \text{Si } X_i <= t <= X_{i+1} \\ 0 & \text{c.o.c} \end{cases}$$

$$N_{i,k}(t) = (t-X_i) * N_{i,k-1}(t)/(X_{i+k-1}-X_i) + (X_{i+k}-t) * N_{i+1,k-1}(t)/(X_{i+k}-X_{i+1})$$

Los valores de X_i son elementos de un vector, el cual cumple la siguiente relación:

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

En el caso de que en la función normalizada Bspline se obtenga 0/0 se ha adoptado la convención de que es igual a 0.
Algunas de las propiedades de las curvas Bspline son:

- La sumatoria de las funciones normalizadas siempre van a ser igual a uno, en otras palabras:

$$\Sigma N_{i,k}(t) = 1.$$

- El valor de cada función normalizada es positivo o cero, para todos los parámetros. Excepto para k = 1 cada función tiene un valor máximo.
- El orden máximo de la curva es igual al número de puntos que definen al polígono.
- La curva generalmente sigue la definición del polígono.
- Si se trabaja con polígonos convexos, la curva se encontrará en la parte interna del mismo.

La convexividad de la curva Bspline es mas notoria que la de las curvas Bezier.

El vector X generalmente se obtiene por las siguientes ecuaciones:

$X_i = 0 \qquad [1 <= i <= k$
$X_i = i-k \qquad [k+1 <= i <= n+1$
$X_i = n-k+2 \qquad [n+2 <= i <= n+k+1$

Para una curva Bspline en donde k = 3 y n + 1 = 4 tenemos que X es igual a:
X = [0,0,0,1,2,2,2]

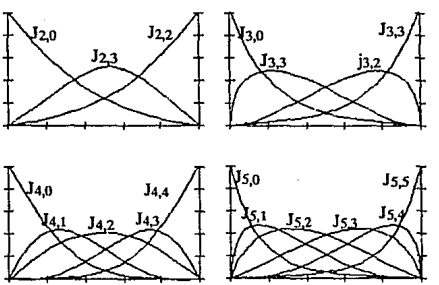


Fig 14. Funciones normalizadas de Bezier.

Las funciones normalizadas gráficamente se observan en la figura anterior:

Betaspline.

Las curvas Betaspline son un área nueva en la Graficación por Computadora. Las curvas Betaspline manejan 2 parámetros, los cuales son β_1 y β_2. β_1 y β_2 generalmente deben de ser positivos o iguales a cero (β_1, $\beta_2 >= 0$). Las fórmulas para obtener una curva Betaspline son:

$b_{-0}(u) = 1/\delta^*(2u^3)$.
$b_{-1}(u) = 1/\delta^*(2^*\beta_1{}^2u^2(3-u) + 2(\beta_1u(3-u^2) + \beta_2u^2(3-2u) + 2(1-u^3)$
).
$b_{-2}(u) = 1/\delta^*(2\beta_1{}^3u((1-u)(2-u)+1) + 2\beta_1{}^2(u^3-3u^2+2) + 2\beta_1(u^3-3u+2) + \beta_2(2u^3-3u^2+1))$.
$b_{-3}(u) = 1/\delta^*(2\beta_1{}^3(1-u)^3)$.
$\delta = \beta_2 + 2\beta_1{}^3 + 4\beta_1{}^2 + 4\beta_1 + 2 <> 0$.
$Q_i(u) = V_{i-3}b_{-3}(u) + V_{i-2}b_{-2}(u) + V_{i-1}b_{-1}(u) + V_ib_{-0}(u)$

De donde:
U : varía de 0 a 1.
i : va de 3 a n.
V_i : Son los vértices del polígono (x,y).
Q_i : son los valores de la curva (x,y).
δ : es una constante.
b(u) : son las curvas normalizadas BetaSpline.

La suma de las curvas normalizadas es igual a 1 en otras palabras:

$b_{-0}(u) + b_{-1}(u) + b_{-2}(u) + b_{-3}(u) = 1$.

La modificación de los valores β_1, β_2 modifican la forma de la curva, en algunas ocasiones la curva se acerca a los vértices del polígono y en algunas otras se aleja.
Si el polígono definido es convexo, la curva estará dentro del polígono.

β_1, β_2 pueden estar definidos para cada uno de los puntos del polígono, es decir cada punto tendrá su β_1 y β_2, lo cual hace más flexible a la curva que estemos trabajando.

Rellenado de Areas.

El rellenado de áreas es necesario y fundamental en el diseño de gráficas. En el diseño nos podemos encontrar con la necesidad de llenar curvas cerradas. Si el contorno es una curva, puede ser aproximada a líneas, y de esta manera a un polígono cerrado. La mayoría de los algoritmos operan sobre polígonos.

Uno de los métodos más sencillos es examinar en un monitor, si un pixel está adentro o afuera del polígono, estas técnicas son muy lentas. Existen otras técnicas como la de barrido y la de la semilla, las cuales son más eficientes que las antes mencionadas

Rellenado por el Método de Barrido.

Esta técnica se apoya en el hecho de que los pixeles adyacentes tienen las mismas características. Para un monitor gráfico los pixeles adyacentes en una línea de barrido tienen las mismas características. Estas características cambian únicamente cuando la línea de barrido encuentra un vértice del polígono. La línea de barrido generalmente es una línea horizontal que barre al polígono completamente.

Esta línea de rastreo puede interceptar al polígono por cualquiera de sus vértices, en una figura como un cuadrado, la línea de rastreo la interceptaría en dos lados. En figuras más complejas la línea de rastreo podría interceptar en más de dos veces a un polígono. Para ejemplificar lo anterior veamos la siguiente figura.

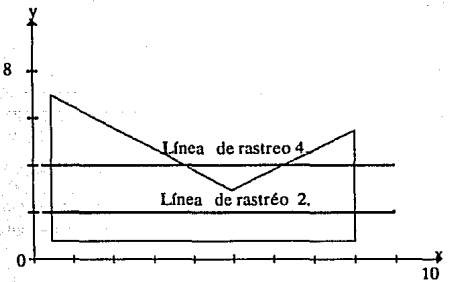


Fig 15. Area sólida en rastréo.

La línea de rastreo 2, intercepta al polígono en dos ocasiones en $x = 1$ y $x = 8$, dividiendo a la línea de rastreo en tres regiones.

Por otra parte las intersecciones para la línea de rastreo 4, no son necesariamente, determinadas de izquierda a derecha. Por ejemplo, si el polígono es especificado por la lista de vértices $P_1P_2P_3P_4P_5$ y la lista de aristas por los pares sucesivos de vértices P_1P_2, P_2P_3, P_3P_4, P_4P_5, P_5P_1, las intersecciones de los vértices en la línea de rastreo 4 será determinada en x igual a 8, 6, 4, 1. En otras palabras se determina la intersección del polígono, de acuerdo a la secuencia de la definición del polígono. Después de haber obtenido las intersecciones se pueden ordenar descendentemente o ascendentemente, por ejemplo, 1,4,6,8.

Para determinar la intensidad, el color o el encendido de los pixeles en una línea de rastreo, las intersecciones son consideradas por pares. Para cada intervalo formado por un par de intersecciones el polígono se rellena en ese intervalo.

Otros algoritmos emplean el concepto de relleno hasta aquí mencionado. Además de ordenar los vértices de menor a mayor en cuanto al eje "y", obteniendo por otra parte a y_{min} y y_{max}, de donde la línea de barrido hará su recorrido desde y_{min} hasta y_{max}.
Por otro lado, se tiene información de la pendiente de cada arista del polígono. Cuando la línea de rastreo hace el barrido sobre el polígono, se cambia la posición de "x" y de "y" de acuerdo a la pendiente que se tiene en cada arista. Además de procesar siempre intersecciones pares.

Existen muchos patrones de rellenado, con lo cual se puede dar lugar al arte por computadora. Otro tipo de rellenado puede darse en la generación de pixeles de colores aleatorios, lo cual da efectos especiales.

Este método funciona únicamente sobre figuras poligonales. Si una curva no está dada de manera poligonal, el rellenado no puede llevarse a cabo. Este algoritmo es muy eficiente ya que se tiene información necesaria y suficiente para calcular las intersecciones de la línea de rastreo con el polígono. La línea de rastreo es una línea horizontal, que realiza un barrido en un sentido vertical, en incrementos unitarios, para que de esta manera, se rellene el polígono completamente. Si el incremento de la línea de rastreo o de barrido es menor a 1 y mayor 0 se tendría información de redundancia. Pero si el incremento de la posición de la línea de rastreo o de barrido es mayor a 1, la figura presentará discontinuidades en el relleno del polígono.

Rellenado por el Método de la Semilla.

Este método supone que por lo menos un pixel en el interior del polígono existe, posteriormente, este algoritmo trata de encontrar otro pixel para ser iluminado.
Una región puede ser definida por interiores o por límites. Una región definida por interiores tiene un color o un valor, y los pixeles exteriores tienen otro. Si una región es definida por límites entonces los pixeles que definen el límite tienen un valor o un color único. Ninguno de los pixeles interiores puede tener el valor de los límites. Pero si embargo los pixeles exteriores pueden tener el valor de la frontera o del límite, las regiones

definidas por interiores o límites pueden ser rellenadas por técnicas de 4 conectadas u 8 conectadas.

Si una región se define como 4 conectada, luego cada pixel en la región puede ser alcanzada por una combinación de movimientos en solo 4 direcciones, como lo son la izquierda, derecha, arriba, abajo, para una región 8 conectada, cada pixel en la región puede ser alcanzado por una combinación de movimientos, los cuales pueden ser 2 en sentido horizontal, 2 en sentido vertical y 4 en direcciones diagonales o inclinadas. Un algoritmo 8 conectado puede llenar una región 4 conectada, pero un algoritmo 4 conectado no puede llenar una región 8 conectada. Para polígonos complejos es importante utilizar algoritmos 8 conectados, por que así, puede rellenarse cualquier rincón del polígono, algunos ejemplos de regiones definidas por interiores se muestran en la figura 16.

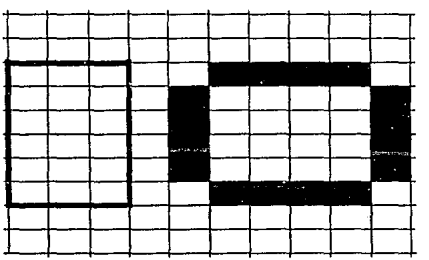


Fig 16. Tipos de definiciones de áreas.

Como se puede ver en la figura 17, cada subregión de la región 8 conectada es 4 conectada. Para pasar de una región a otra se requiere de un algoritmo 8 conectado. Sin embargo, si cada una

de las subregiones es una región separada 4 conectada, cada una puede ser llenada con un valor o color separado, luego usando un algoritmo 8 conectado causa que ambas regiones sean incorrectamente llenadas con un solo color o valor.

La figura 17 muestra una región definida por límites, la cual es una región 8 conectada.

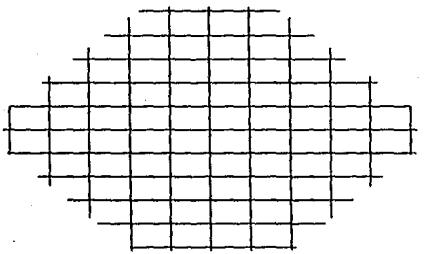


Fig 17. Región definina por límites 4 y 8 conectada.

Un método sencillo para rellenado por semilla puede ser obtenido, usando un stack. Un stack puede ser definido por un arreglo o por un espacio de almacenamiento, en el cual algunos valores pueden ser secuencialmente ingresados o del cual pueden ser secuencialmente removidos. Algunos valores pueden ingresarse o extraerse utilizando algunas técnicas conocidas como lo son PUSH y POP. Un stack puede ser definido en términos del que el primero que entra es el último que sale, también conocido como stack PUSH-DOWN, a este método también se le conoce como método FILO. Este algoritmo puede enunciarse de la siguiente manera:

- Llevar a un PUSH la semilla inicial.
- Mientras el stack no este vacío.
- Hacer un POP en el stack.
- Realizar el encendido, el color o el sombreado del pixel.
- Para cada uno de los pixeles adyacentes 4 conectados al pixel actual, checar si es un pixel de la frontera o si ya ha sido iluminado, si es así, ignorarlo. En cualquier otro caso hacer un PUSH.
- Ir al paso 2.

Los algoritmos 8 conectados funcionan de manera similar al algoritmo anterior 4 conectado. Los algoritmos 8 conectados funcionan en 8 direcciones, las cuales son izquierda, derecha, arriba, abajo y 4 en direcciones inclinadas. El algoritmo anterior 4 conectado puede cambiarse a un algoritmo 8 conectado, modificando el punto 5.

Este algoritmo es ineficiente y por lo tanto no recomendable. Como se puede observar, para figuras con dimensiones grandes, el stack de la computadora se saturaría, si se utiliza el llamado recursivo de este algoritmo. En una computadora PS/2 30286 IBM aproximadamente el stack oscila entre 200 y 300 (utilizando un modelo de memoria mediano). Con esto se quiere decir que la figura a rellenar puede tener entre 200 y 300 pixeles coloreados, lo cual es un obstáculo grande. Para polígonos grandes este método no funcionaría.

Ahora si utilizamos un vector de almacenamiento como stack, la memoria de la máquina tendería a saturarse nuevamente.
Para evitar este problema se utiliza un algoritmo de llenado por el método de la semilla. Utilizando una línea de rastreo, la cual va a poner en el stack un punto ya sea en la línea superior o inferior. Este algoritmo es aplicable a regiones definidas por límites, la cual puede ser convexa o cóncava, además, puede contener uno o más hoyos internos. La región exterior a la región adyacente definida por límites no puede contener pixeles con otro color o valor de acuerdo al que se está utilizando para el rellenado de la región o del polígono. Este algoritmo puede enunciarse de la siguiente manera:

- Se realiza un PUSH con el pixel semilla.
- Mientras el stack no esté vacío.
- Realizar un POP.
- El ancho del polígono que contiene el pixel actual es rellenado hacia la derecha y a la izquierda, a lo largo de la línea de barrido hasta que el límite es encontrado.
- El algoritmo memoriza el extremo izquierdo y el extremo derecho (X_L,X_R).
- En el rango de $X_L <= X <= X_R$ la línea de rastreo hace un barrido en la parte superior e inferior inmediata, si se detecta que es una región interna se realiza un PUSH.
- ir al paso 2

Este algoritmo es inicializado colocando la semilla en el stack y termina cuando el stack está vacío.

Fractales.

Algunas aplicaciones requieren del uso de la geometría fractal. Disciplinas como la Geografía, la Topología, la Paleontología, la Medicina, la Química requieren del uso de la graficación fractal. Por ejemplo, la Geografía requiere del uso de la geometría fractal, para la realización de mapas, con imágenes tomadas desde un satélite. La Topología requiere a la geometría fractal, para la representación de superficies, montañas, caminos, ríos, lagos, etc. La Paleontología se apoya en la geometría fractal, para la representación de posibles fósiles encontrados en las capas de la tierra. La medicina puede hacer uso de los fractales, por ejemplo, para la representación de tejidos, membranas, etc. Algunos elementos químicos son radiactivos, la radiactividad, puede ejercerse en una área no regular, la geometría fractal puede dibujar el contorno de esta área no regular. Las radiaciones ejercidas por estos elementos químicos pueden producir efectos sobre superficies, las representaciones de las superficies alteradas por la radiación, pueden ser representadas mediante el uso de los fractales. La superficie alterada puede lograrse con un mínimo de detalle o con un máximo de detalle, e inclusive pueden aplicarse técnicas de iluminación, supresión de líneas o áreas ocultas para obtener una definición más realista de esta superficie dañada por la radiación. Las técnicas fractales fueron iniciadas por Mandelbrot. Con la geometría de Euclides no se podría

representar la mayoría de aplicaciones que utilizan fractales, la geometría Euclidiana no podría representar, por ejemplo, la superficie que comprende el estado de Durango, la geometría fractal, sí podría.

Dentro de los fractales se maneja un parámetro llamado "dimensión". La dimensión de una figura se calcula por la siguiente fórmula:

$$D = Ln(N)/Ln(1/S).$$

de donde:
N: es el número de divisiones de la figura.
S: es el escalamiento o división de la figura.
Para entender el concepto de dimensión utilicemos la figura 18 que a continuación se presenta.

Fig 18. Producción de una curva fractal.

La figura (a) es la figura inicial, la figura (b) es una modificación de la figura (a), la figura (b) está compuesta por líneas de la figura (a), las cuales fueron escaladas a S = 1/3, el número de líneas que componen los lados modificados de la figura (a) en la figura (b) son N = 4. Y por lo tanto si analizamos la longitud de la figura (b) con respecto a la de la figura (a), la longitud de la figura (b) es 4/3 más grande que la longitud de la figura (a). Y por lo tanto la dimensión de esta figura es:

$$D = Ln(4)/Ln(3) = 1.26.$$

La dimensión de un figura bidimensional está comprendida entre 1 y 2, la dimensión de una figura tridimensional está comprendida entre 1 y 3. Para que una figura tenga dimensión 1 se requiere que N = 1/S, de acuerdo a la fórmula de la dimensión tenemos que:

$$D = Ln(1/S)/Ln(1/S) = 1.$$

Para que una figura tenga dimensión 2, se requiere que $N = 1/S^2$. De acuerdo a la fórmula de la dimensión tenemos que:

$$D = Ln(1/S^2)/Ln(1/S) = 2 * Ln(1/S)/Ln(1/S) = 2.$$

Después de haber analizado el concepto de dimensión, se puede decir que una figura fractal puede tener dimensión fraccionaria. La realización de fractales requieren de funciones de transformación, las cuales pueden dar lugar a variaciones regulares o al azar. Un ejemplo de variación regular se da en la figura 18. Un ejemplo de variación al azar se da en la figura 19.

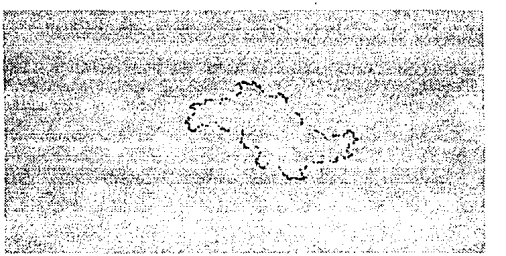


Fig 19. Curva fractal Mandelbrot con generación al azar.

Si observamos una montaña a distancia y si obtenemos su re presentación mediante los fractales, tiene un mínimo de detalle, pero si nos acercamos un poco más el detalle se incrementa, pero si aún nos acercamos un pocos más el detalle se incrementa aún todavía más, pudiendo visualizar las salientes, e inclusive algunos árboles de la montaña. Cuando el detalle es mínimo, la longitud de la figura de la montaña es mínima, pero si el detalle es considerable, la longitud de la montaña se incrementa, debido al detalle que compone a la figura.

La mayoría de los fractales utilizan funciones de transformación, algunas de estas funciones de transformación utilizan números imaginarios. Una de las funciones más comunes de transformación descubierta por Mandelbrot es la siguiente:

z = 1/2*[1±SQR(1-4*z'/Lambda)].

De donde:
z' : es un número imaginario, que puede ser mapeado a los ejes cartesianos.
Lambda : es un número constante complejo.
z : es un número complejo que puede ser mapeado a los ejes cartesianos.

La fórmula anterior opera de una manera interactiva. La generación de imágenes fractales requiere de una gran cantidad de iteraciones. Siempre el punto siguiente estará en función del punto anterior, esto se puede visualizar mejor en la siguiente ecuación:

$P_1 = F(P_0)$ $P_2 = F(P_1)$ $P_3 = F(P_2)$.

Con los fractales podemos llevar a cabo superficies, volúmenes, sólidos, entre otros. Algunos métodos para generar estos objetos emplean números cuaterniones. Para lograr un realismo en las aplicaciones, se procede a utilizar técnicas de iluminación para obtener brillantez y color en la imagen de trabajo, así como algunos métodos para la supresión de superficies ocultas.
Los elementos componentes de una figura elaborada con fractales pueden ser líneas, puntos, polígonos, etc.

Las técnicas de Von Koch Snowflake para la generación de fractales se construyen utilizando la siguiente metodología. Se comienza con un iniciador, el cual puede ser una línea recta o un polígono. Cada lado del iniciador es entonces substituido por el generador, el cual es un conjunto de líneas conectadas. Las cuales forman una ruta desde el comienzo hasta el final de la línea a ser substituida (Generalmente los puntos del generador son una red cuadrada o una red compuesta de triángulos equiláteros). Posteriormente, cada segmento de la figura nueva es substituida por una versión escalada de el generador, este proceso continua indefinidamente. Por supuesto no se puede continuar el proceso infinitamente, y si se realiza, el resultado no es interesante, debido a que el detalle puede estar lejos de la opreciación en el alcanze en la resolución de nuestro monitor. En la práctica generalmente se realiza de 2 a 16 repeticiones.

En el proyecto GPC se implementa una curva Snowflake de trece segmentos, la cual tambien fué descubierta por Mandelbrot. La dimensión de la curva Snowflake es de 2. Para esta curva puede haber cuatro elecciones en la posición del generador, el cual debe de ser cuidadosamente seleccionado para cada nivel y para cada segmento. Por otro lado, también debe de asegurarse de que la curva no se está intersectando o sobreponiendo.

En la generación de las curvas fractales se emplea muy frecuentemente un parámetro llamado nivel (level), el cual nos indica el número de veces que se substituirá un generador en cada iniciador para cada interacción.

La curva Pharaoh es el nombre dado por Mandelbrot a una figura creada por círculos usando una función inversa. Si se observa el proyecto o programa de este trabajo. Los dos círculos principales nos dan una referencia para la generación de los otros círculos. Los cuales son mapeados por inversión a dos líneas inferiores y superiores. Los otros círculos son tangentes a cualquier otro. Los cuales se pueden extender hacia la izquierda o hacia la derecha tanto como se desee. El número de círculos es puesto en un parámetro, en un loop.

CAPITULO IV.

GRAFICACION EN 3D.

Graficación en 3D.

La graficación en 3D, obtiene un realismo aceptable en las figuras de graficación que se están procesando, por ejemplo, tenemos letras tridimensionales (con fondo y sombra), cubos, entre otros. Con la graficación en 3D se pueden realizar aplicaciones interesantes, por ejemplo, la representación de montañas, presas, la trayectoria de un automóvil, etc.

Transformación entre Sistemas Coordenados.

En la mayoría de las aplicaciones de 3D, se opera de una manera aislada, la cual es irrealista. La mayoría de las aplicaciones reales requieren de la transformación de sistemas coordenados. Por ejemplo, en el diseño de un tornillo, para un sistema de ruedas delanteras de un autobús, se puede manejar la construcción y diseño del tornillo en un sistema de coordenadas 1, por otra parte el sistema de ruedas delanteras del camión se

puede manejar en un sistema coordenado 2, por otra parte el camión pertenece a otro sistema coordenado 3, cuando el tornillo es terminado, se ensambla al sistema coordenado de llantas delanteras, se puede realizar una traslación para que coincidan los orígenes coordenados y de esta manera el ensamble es realizado, la escala puede ajustarse mediante una escalación en caso de ser necesaria, además el sistema de ruedas puede ser rotado, en caso necesario, para que de esta manera coincida con el tornillo diseñado. Este ejemplo da una idea entre la relación que pueden tener los sistemas coordenados, además de las transformaciones necesarias para ser ajustados.

Representación de Objetos en 3D.

El ingreso o la captación de la información en 2D es muy fácil. Basta con colocar un cursor en una posición determinada y seleccionarla, para obtener así, un punto o un vértice del objeto. Para definir el polígono, basta situar el cursor en la pantalla y dar un <INTRO> , obteniendo así los vértices o las aristas que conforman al polígono. Para 3D este concepto cambia en gran medida, para obtener o definir un vértice de un objeto en 3D, es difícil, ya que no se puede apreciar con exactitud la ubicación del cursor en ancho, alto, y profundidad.

Por ejemplo, si quisiéramos definir los vértices de un cubo, sería difícil. Obteniendo quizás un cubo deforme, debido a la difícil apreciación en ancho, alto y profundidad en los vértices. Por otro lado, es difícil ingresar la información, sobre conectividad del vértice definido, con los demás vértices. La mejor forma de solucionar este problema, es el trabajar con volúmenes predefinidos. Para la definición de volúmenes complejos se puede utilizar la sobreposición de los volúmenes predefinidos, además de aplicar técnicas de supresión de líneas y superficies ocultas, para obtener así, una mejor definición del volumen que se este procesando. Estos volúmenes predefinidos pueden representarse por una lista de vértices, una de aristas, y/o una de polígonos planos.

En la realidad, se requiere del diseño de motores, tornillos, relojes, carrocerías para aviones, etc. Para lograr esto se aplica un método de aproximación, el cual consiste en la definición de caras

poligonales planas. En la figura que en seguida se muestra se puede apreciar este método.

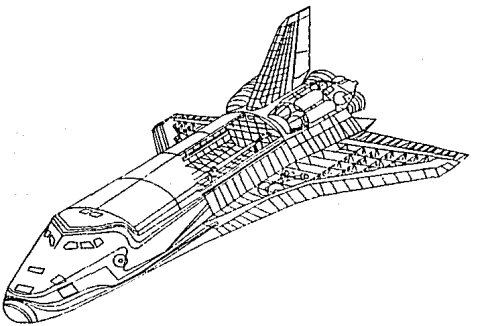


Fig 20. Figura tridimensional compuesta de caras planas.

El objeto sólido de la figura 20, está representado como un conjunto de superficies poligonales.

La información de las superficies de los sólidos se puede almacenar en tablas, las cuales pueden contener información para el procesamiento o información para el despliegue.

La información para el procesamiento nos ofrece, datos de propiedades geométricas del objeto, así como algunos datos en los atributos del objeto. Dentro de las propiedades geométricas del objeto podemos encontrar coordenadas y parámetros de fronteras, para identificar la orientación en el espacio, de las superficies poligonales. Toda esta información se puede almacenar en una lista de vértices, en una lista de aristas, y/o en una lista de polígonos. En la información de atributos del objeto podemos encontrar color, escala de grises, sombreado, entre otros.

En la figura que en seguida se muestra, se puede apreciar un objeto tridimensional, formado por dos superficies poligonales planas, formadas con seis aristas y cinco vértices.

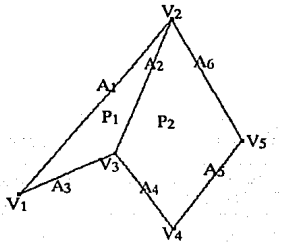


Fig 21. Superficie poligonal de un cuerpo en 3D.

Las tablas de vértices, aristas y polígonos son:

Vértices.

V_1	(x_1,y_1,z_1)
V_2	(x_2,y_2,z_2)
V_3	(x_3,y_3,z_3)
V_4	(x_4,y_4,z_4)
V_5	(x_5,y_5,z_5)

Aristas.

A_1	V_1,V_2
A_2	V_2,V_3
A_3	V_3,V_1
A_4	V_3,V_4
A_5	V_4,V_5
A_6	V_5,V_2

Polígonos.

P1	A_1, A_2, A_3
P_2	A_2, A_4, A_5, A_6

La información esencial se encuentra dentro de la tabla de vértices. Sin esta tabla no se podrían dibujar las superficies que conforman al volumen.

La tabla de aristas, tiene la información necesaria para dibujar una superficie, al igual que la tabla de polígonos.
Con la tabla de aristas se dibujaría una sola vez cada arista de la superficie poligonal, mientras que con la tabla de polígonos se dibujarían dos veces algunas aristas.

En algunas aplicaciones es recomendable utilizar tablas de aristas, y en algunas otras tablas de polígonos, según las necesidades de las aplicaciones. La tabla de vértices es fundamental, debido a que las tablas de aristas y polígonos están en función de la tabla de vértices.

Transformaciones 3D.

Las transformaciones 3D son similares a las transformaciones en 2D. En las transformaciones de 3D podemos tener traslación, escalación, rotación, espejo, entre otras.

Traslación.

La traslación nos sirve para cambiar de posición a un punto. Y por lo tanto, se puede aplicar la traslación a un plano y en consecuencia a un volumen. La representación en coordenadas homogéneas de la traslación, está definida por la siguiente representación matricial:

$$[x',y',z',1] = [x,y,z,1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$$

En otras palabras tendríamos las siguientes ecuaciones:

$x' = x + Tx,$
$y' = y + Ty,$
$z' = z + Tz,$

de donde:
Tx,Ty,Tz: son constantes de traslación.
x,y,z: es un punto a trasladar.
x',y',z': es el punto resultante de la traslación.

Si un volumen está definido por un conjunto de aristas, se efectúa la traslación de cada una de las aristas. Por otro lado, si un polígono está definido por un conjunto de caras, o planos, se realiza la traslación de cada uno. Obteniendo así la traslación del volumen.

Si Ty y Tz son iguales a cero, y si Tx es diferente de cero, se realizará una traslación paralela o sobre el eje x. Similarmente, podría realizarse la traslación sobre cualquiera de los ejes y, z.
Para realizar una traslación en una dirección no paralela a uno de los ejes, se requiere que por lo menos dos constantes de traslación sean diferentes de cero. Los valores de las constantes de traslación pueden ser positivos o negativos.

Cuando se realiza el diseño de software, la traslación en 2D puede aprovecharse para la traslación en 3D, la traslación en 2D se realiza en un plano XY, la traslación en 3D se puede realizar en los planos XY, YZ y XZ. Como se mencionó anteriormente la traslación de un punto P_1, a un punto P_2, puede ser no paralela a estos planos, entonces la traslación se logrará mediante una combinación de traslaciones en cada uno de los planos antes mencionados.

Escalamiento con Respecto a un Punto.

La escalación con respecto a un punto de referencia, como se mencionó anteriormente, nos amplifica o desamplifica un volumen.

La representación de la escalación en coordenadas homogéneas está dado por la siguiente representación matricial.

$$[x',y',z'] = [x,y,z,1] \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ (1-Sx)Xf & (1-Sy)Yf & (1-Sz)Zf & 0 \end{bmatrix}$$

De donde:
Sx,Sy,Sz: es el factor de amplificación.
Xf,Yf,Zf: es el punto de referencia.
x,y,z: Es el punto a escalar.
x',y',z': Es el punto escalado.

Para lograr la escalación con respecto a un punto de referencia se traslada el punto a escalarse hacia el origen, posteriormente se realiza la escalación con respecto al origen y por último se regresa el punto de referencia junto con el punto escalado a su origen, en otras palabras, se puede enunciar de la siguiente manera:

T(-Xf,-Yf,-Zf)S(Sx,Sy,Sz)T(Xf,Yf,Zf).

Si el factor de escalación es unitario, la figura después de esta transformación queda igual. Ahora, por ejemplo, si el factor de escalación es 0.9, la figura se desamplifica en una décima fracción del volumen original. Por otro lado si el factor de escalación es 1.1 el volumen se incrementa en una décima parte.

Si los factores de escalación Sx,Sy,Sz son diferentes, el volumen tiende a distorsionarse. Por el contrario, si Sx,Sy,Sz son iguales, el volumen se escala sin modificar su apariencia original.

Rotación con Respecto a un Eje.

La rotación en 3D nos permite girar un volumen con respecto a un eje cartesiano o con respecto a un eje arbitrario. Para realizar una rotación sobre el eje x, se realiza la modificación de los valores Y,Z de los vértices de un volumen arbitrario, los valores de la variable X conservan su valor.

Para realizar la rotación con respecto a "y" se modifican los valores X,Z de los vértices a ser rotados, y ahora, los valores de la variable Y permanecen constantes.

Para realizar la rotación con respecto al eje "z" se modifican los valores X,Y de los vértices del volumen a ser rotados, y ahora, los valores de Z permanecen constantes.

El sistema matricial en coordenadas homogéneas para la rotación con respecto al eje "x", el eje "y" y el eje "z", son respectivamente las siguientes:

$$[x',y',z'] = [x,y,z,1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \operatorname{sen}(\theta) & 0 \\ 0 & \operatorname{sen}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x',y',z'] = [x,y,z,1] \cdot \begin{bmatrix} \cos(\theta) & 0 & -\operatorname{sen}(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \operatorname{sen}(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x',y',z'] = [x,y,z,1] \cdot \begin{bmatrix} \cos(\theta) & \operatorname{sen}(\theta) & 0 & 0 \\ -\operatorname{sen}(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De donde:
θ: es el ángulo de rotación.
x,y,z: es el punto a ser rotado.

x',y',z': es el punto resultante de la transformación.

Cuando se efectúa la rotación con respecto a uno de los ejes antes mencionados, es como si, una persona se ubicara sobre ese eje, y se procediera a la rotación sobre el plano que se tiene enfrente, la rotación con respecto a los ejes y utilizando esta idea se puede apreciar en la siguiente figura:

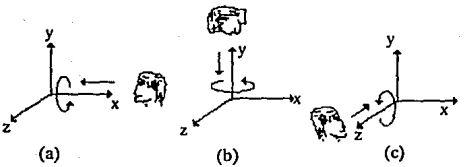


Fig 22. Rotación con respecto a los ejes cartesianos.

Cuando se realiza la rotación con respecto a un eje que no es paralelo a ninguno de los ejes cartesianos se procede con los siguiente pasos:

- Se traslada el objeto de manera que el eje de rotación coincida con uno de los ejes paralelos.
- Se efectúa lo rotación.
- Se traslada el objeto de manera que el eje de rotación se devuelva a su posición original.

En la realización de software se aplica la rotación de 2D en 3D. La rotación en 3D utiliza el mismo concepto utilizado en 2D. La rotación en 3D a fin de cuentas se realiza sobre un plano.

La rotación de un volumen se logra mediante la rotación de cada uno de sus caras o bien mediante la rotación de cada una de las aristas que conforman al volumen.

Espejo con Respecto a un Plano.

Consiste en la reflexión de un volumen con respecto a un plano. La reflexión o el espejo puede realizarse sobre los planos X, Y o Z. Si la reflexión se realiza con respecto al plano Z se tiene la siguiente representación matricial en coordenadas homogéneas:

$$RFz = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Los valores de los vértices en X,Y se mantienen constantes, el valor Z cambiará de signo.

Las representaciones matriciales en coordenadas homogéneas para el plano X y Y son las siguientes respectivamente:

$$RFx = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$RFy = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Concatenación de Transformaciones.

Al igual que en 2D la concatenación de transformaciones se realiza mediante el producto de las transformaciones en forma matricial. El sistema que definen las matrices, nuevamente está dado en coordenadas homogéneas a lo cual, como ya se había mencionado se le conoce como matriz compuesta. Al producto

de las matrices se le conoce como Eslabonamiento o Composición de matrices. La traslación doble de un punto en el espacio puede realizarse mediante la composición de las matrices de cada uno de las traslaciones. Tal y como se puede ver en las siguientes representaciones matriciales:

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ Tx_1 & Ty_1 & Tz_1 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ Tx_2 & Ty_2 & Tz_2 & 1\end{bmatrix}=$$

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ Tx_1+Tx_2 & Ty_1+Ty_2 & Tz_1+Tz_2 & 1\end{bmatrix}$$

En otras palabras:

$$T(Tx_1,Ty_1,Tz_1)T(Tx_2,Ty_2,Tz_2)=T(Tx_1+Tx_2,Ty_1+Ty_2,Tz_1+Tz_2)$$

Proyecciones.

Las proyecciones más importantes son la proyección paralela y la proyección perspectiva. El uso de la proyección paralela se da en el dibujo profesional, en el diseño, en la construcción de piezas, etc. Debido a que la proyección paralela mántiene las dimensiones relativas de un objeto. Por otro lado, la proyección en perspectiva no mantiene las dimensiones relativas del objeto.

La proyección paralela esta compuesta por la proyección paralela ortogonal y por la proyección paralela oblicua. La proyección paralela ortogonal se da cuando la dirección de la proyección del objeto de trabajo es perpendicular al plano de proyección. La proyección paralela oblicua no mantiene esta perpendicularidad.

Proyección Perspectiva.

Cuando salimos a carretera en un automóvil, el final de la carretera, se observa, como si la carretera se juntara o como si las orillas de la carretera se unieran en un punto, a esta proyección se le conoce como proyección perspectiva. Al punto de convergencia se le conoce como Punto de Fuga. Los puntos de fuga principales son paralelos a cualquiera de los ejes cartesianos. Por lo tanto la proyección perspectiva puede contar con uno, dos o tres puntos de fuga principales. En la siguiente figura se tiene un ejemplo de proyección perspectiva con uno y dos puntos de fuga principales. La figura 23.a es la figura inicial, la

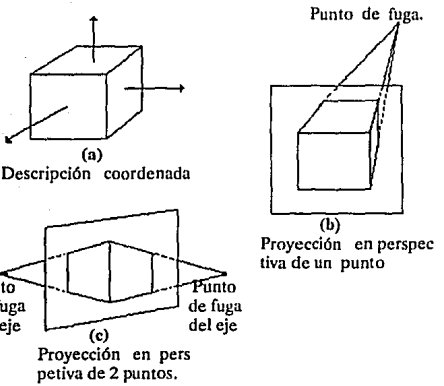


Fig 23. Proyección con diferentes tipos de fuga.

figura 23.b es la figura inicial en perspectiva con un punto de fuga principal paralelo al eje z. La figura 23.c contiene dos puntos de fuga principales, los cuales se encuentran sobre el eje z y el eje x.

Para obtener las ecuaciones que rigen la proyección central perspectiva utilicemos la siguiente figura:

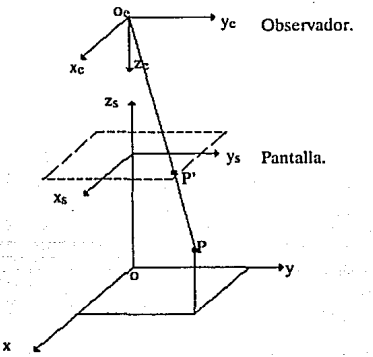


Fig 24. Proyección central perspectiva.

En la figura anterior se puede apreciar un punto P(x,y,z) en coordenadas mundiales, un punto $P_s(x_s,y_s,z_s)$ en coordenadas de la pantalla y un punto $P_c(x_c,y_c,z_c)$, el cual es el punto, en el cual se encuentra parado el observador. Para simplificar los cálculos el observador se encuentra sobre el eje z del sistema de coordenadas mundiales, y además el eje z del sistema de coordenadas mundiales coincide con el eje z del sistema de coordenadas del observador.

Por otro lado se maneja un parámetro D y un d, de donde D es la distancia entre el origen del sistema de coordenadas mundiales y el origen del sistema de coordenadas del observador, por otra parte, d es la distancia del origen del sistema de coordenadas del observador al plano de la pantalla. Si obtenemos una vista de perfil de la figura 24, obtenemos la siguiente figura.

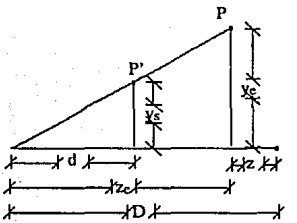


Fig 25. Proyección de la figura anterior.

Por triángulos semejantes obtenemos:

$y_s/d = y_c/z_c = => y_s = d*y_c/z_c$.
$x_s/d = x_c/z_c = => x_s = d*x_c/z_c$.

De la figura 25 observamos que: $(x_c,y_c,z_c) = (x,y,D-z)$, y por lo tanto:

$x_s = d*x/(D-z)$.
$y_s = d*y/(D-z)$.

Si D es igual a infinito, y $D = d$, obtenemos una proyección paralela, sobre el plano XY.

Proyección Paralela.

La proyección en paralelo es una herramienta esencial en el dibujo, en general. Las proyecciones en paralelo se usan para visualizar las caras de un volumen o de un objeto con respecto a planos. Los cuales se conocen como vistas. Cuando la proyección en paralelo es ortogonal al plano de proyección, se le conoce como proyección paralela ortogonal, si la proyección no es ortogonal se obtiene una proyección oblicua. Los planos de visión son tres: plano de visión lateral, plano de visión superior y plano de vista frontal. A los planos de visión frontales y laterales se le conocen como elevaciones, y a los planos de visión superior se le conocen como plantas. Estos planos son también conocidos con el nombre de proyección isométrica. Cuando en un plano de proyección se visualiza mas de una cara, se le conoce como proyección axonométrica.

En la siguiente figura se puede observar una proyección isométrica:

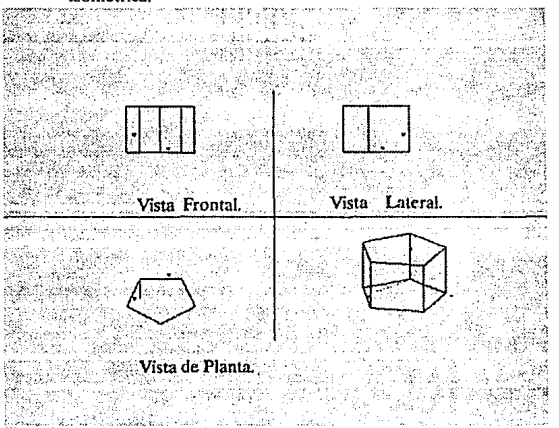


Fig 26. Proyeccion Isométrica de un volumen.

Para realizar una proyección en el plano de visión frontal, se le asigna una valor cero a la variable Z, para obtener una proyección en el plano de visión lateral se procede a igualar los valores de X en el volumen iguales a cero. Por otra parte para obtener una proyección sobre el plano de vista superior se procede a igualar a cero los valores de Y, en el volumen. Suponiendo que los ejes cartesianos están definidos de la siguiente manera: "x" hacia la derecha, "y" hacia arriba y "z" dirigido hacia nosotros. Por lo que la normal del plano de proyección de vista lateral será paralela al eje "x", la normal del plano de visión frontal será paralela al eje "z", y la normal del plano de visión superior será paralela al eje "y". Todas las normales antes mencionadas van en sentido positivo, sobre los ejes cartesianos.

Recorte 3D.

Las ventanas utilizadas en el recorte tridimensional, por lo general son paralelepípedos rectangulares, como por ejemplo una caja, un cubo, etc. Por otro lado, los volúmenes que determinan la ventana pueden ser aprovechados utilizando una proyección perspectiva.

Esos volúmenes cuentan con seis lados, el izquierdo, el derecho, la cima, el fondo, la parte frontal y la parte trasera.

Al igual que en el recorte 2D, las líneas son totalmente visibles o trivialmente visibles, empleando códigos en los puntos extremos o finales de la línea recta a ser recortada. Para el recorte tridimensional se utiliza un registro para un código de 6 bits para los extremos de la línea recta.

- El primer bit se pone en 1, si el extremo de la línea está a la izquierda del volumen.
- El segundo bit se pone en 1, si el extremo de la línea está a la derecha del volumen.
- El tercer bit se pone en 1, si el extremo de la línea está debajo del volumen.
- El cuarto bit se pone en 1, si el extremo de la línea está arriba del volumen.

- El quinto bit se pone en 1, si el extremo de la línea está enfrente del volumen.
- El sexto bit se pone en 1, si el extremo de la línea está atrás del volumen.

De otra manera el bit se pone en ceros. Tal y como se mencionó en el recorte para dos dimensiones, nuevamente aquí, si los códigos de los puntos extremos son cero, ambos extremos son visibles y por lo tanto la línea es visible. También si nos es cero, la intercepción lógica bit por bit de los códigos en los extremos de la línea recta es totalmente invisible. Si la intercepción lógica es cero, la línea puede ser parcialmente o totalmente invisible. En este caso es necesario determinar la intercepción de la línea con la ventana de recorte.

Para determinar los códigos de los extremos de la recta para una ventana de recorte, se usa una extensión del algoritmo de 2D. Cuando se tiene un volumen de recorte en perspectiva se requieren otras consideraciones adicionales. Para realizar un recorte tridimensional se puede utilizar el algoritmo tridimensional de la subdivisión en el punto medio, además de utilizar el código en los extremos de la línea.

Por otra parte se puede realizar un recorte tridimensional, utilizando el algoritmo de Cyrus-Beck. Este algoritmo no representa ninguna restricción excepto que la ventana debe ser convexa. En lugar de utilizar k vértices se utilizan n planos. todos los valores utilizados en este algoritmo, tienen 3 componentes x,y,z. La extensión del producto punto es aplicable para visibilidad.

Líneas y Superficies Ocultas.

La creación de imágenes realistas en 3D, requieren del uso de las líneas y superficies ocultas. Para la creación de estas imágenes, se han desarrollado algoritmos que identifican que líneas o que superficies son visibles, parcialmente visibles o invisibles. Para que posteriormente se desplieguen las porciones visibles del objeto o para eliminar la porciones que son ocultas a nuestra vista.

En la mayoría de los algoritmos, hay dos suposiciones prácticas. La primera es que si un objeto tiene caras curvas, se usaran segmentos de líneas para aproximarlas a polígonos. La segunda es que todas las superficies son planas. Lo cual quiere decir que los límites de los polígonos en las superficies se encuentran en el mismo plano.

Cabe mencionar, que la primera suposición no es aplicable al algoritmo Eliminación de Superficies en Areas, debido a la naturaleza de su operación. Este algoritmo genera superficies, utilizando una función de la forma $y = f(x,z)$.

Eliminación de Superficies en Objetos Convexos.

Para la realización u obtención de figuras reales se requiere del uso de la eliminación de algunas líneas o superficies. Para realizar esto existen gran cantidad de algoritmos, los cuales determinan que líneas o que superficies son visibles, parcialmente visibles o totalmente invisibles. La mayoría de los algoritmos asumen que un volumen está compuesto de líneas, las cuales pueden aproximarse a polígonos, los cuales en algunos casos se utilizarán como bandera en la eliminación de superficies.

Si contamos con un solo objeto, y si este objeto es convexo podemos solucionar la visibilidad de las caras mediante las normales de los polígonos o de las caras que conforman el volumen. La normal se obtiene mediante el producto cruz de dos vértices que forman cualquier lado de la cara del volumen, los vectores son obtenidos de la diferencia existente entre tres vértices. Un cubo, una pirámide, un octaedro son ejemplos de volúmenes convexos.

Observemos el sólido de la figura 27 que a continuación se muestra. La figura (a) tiene ocho vértices, doce aristas y seis caras. Cada cara nos da una bandera obtenida de los límites poligonales. Si observamos al cubo desde un punto P, solo dos o tres de las caras no son visibles, las caras son ocultas. Para evitar el dibujo de las caras se requiere de una prueba la cual nos dirá si es una cara o superficie visible o invisible.

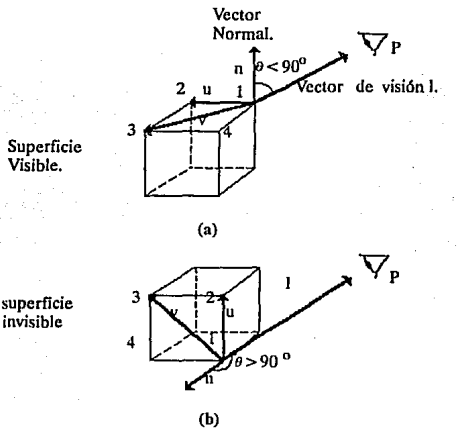


Fig 27. Normal y vector de visión en la visibilidad.

Para determinar la visibilidad de una cara, introduzcamos dos vectores de orientación, asociados con cada una de las caras o superficies, los cuáles son el vector normal a la superficie y un vector de visión. Como ya mencionó el vector normal se obtiene de dos aristas de la cara, y el vector de visión se obtiene de un punto P a cualquier vértice de la arista del volumen. El punto P es el sitio en el cual se observa al volumen. Además se requiere de un ángulo θ, entre el vector normal n y el vector de visión. Si el ángulo se encuentra entre 0 y 90, la superficie es visible y debe de ser desplegada, si es mayor a 90 la cara es oculta y por lo tanto es invisible.

La ecuación básica utilizada en la prueba de visibilidad es:

$$\theta = \cos^{-1}(n \cdot l / (|n| * |l|)).$$

De donde:
n: es el vector normal.
l: es el vector de visión.
θ: es el ángulo entre n y l.
n: se calcula de: uxv.
u,v: son las aristas de la cara, en forma vectorial.
Para una cara visible θ se encuentra entre 0 y 90, por lo que n·l va a ser mayor a cero. En otro caso n·l será menor a cero y por lo tanto la superficie será oculta.

Eliminación de Superficies en Areas.

Existen algoritmos para la supresión de líneas o superficies ocultas. Generalmente, cada algoritmo se ajusta a un caso. Por ejemplo, en la eliminación de superficies o líneas de volúmenes cóncavos existen algoritmos ya desarrollados para estos casos. Para la supresión de caras en un ambiente en donde existen más de un volumen se han desarrollado algoritmos especiales.

En la eliminación de superficies se eliminan las áreas no visibles, como por ejemplo, las generadas por funciones de la forma y = f(x,z). Uno de los algoritmos más comunes es el desarrollado por Watkins, también conocido como el algoritmo de la Máscara. Este algoritmo basa su operación en dos premisas fundamentalmente. 1) las líneas de en frente son dibujadas antes de que se dibujen las líneas posteriores. 2) Una línea o posición de línea es enmascarada (oculta), si esta se encuentra en una región limitada por líneas previamente dibujadas. Tal y como se puede apreciar en la figura.

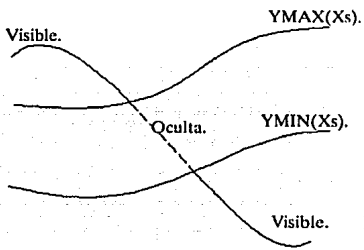


Fig 28. Figura que muestra la técnica de la máscara.

La información de las líneas previamente dibujadas se almacenan en dos vectores YMAX[I] y YMIN[I]. Los cuales son inicializados a cero y a Ysmax respectivamente. De donde Ysmax es la coordenada máxima vertical de la pantalla o del monitor que en ese momento se este utilizando.

Estos arreglos son utilizados para identificar cada coordenada horizontal en la pantalla.

YMAX[I] va a contener información sobre las coordenadas verticales máximas para cada posición de Xs.

YMIN[I] por otra parte va a tener información sobre las coordenadas verticales mínimas para cada posición de Xs.

Un ejemplo de la supresión de superficies se puede observar en la siguiente figura.

Fig 29.Superficie oculta generada por una funcion y = f(x,z)

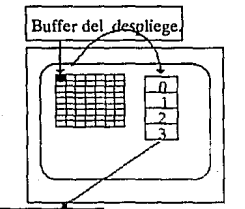
CAPITULO V.

TEORIA DEL COLOR.

Generación de Paletas de Color.

En el modo gráfico 320x200, podemos especificar dos parámetros relacionados para el color, el background y la paleta. El background consiste en los pixeles que se encuentran apagados.

La paleta se selecciona entre los dos conjuntos disponibles de 3 colores que trabajaran con los subsecuentes pixeles a ser encendidos. A cada pixel en la pantalla se le asigna un número entre 0 y 3. La figura 30 muestra como este código del tipo numérico corresponden a los colores en la pantalla de estas dos diferentes paletas. De esta tabla podemos observar que el color 0 está determinado por el color del background actual, pero los otros números tienen dos posibilidades de acuerdo a la elección de la paleta. El búffer del despliegue genera un código para el color (de 0 a 3) para cada pixel. Luego, el "Registro para Selección de Color" determina el background para los pixeles apagados y la paleta (entre 0 y 1) para los pixeles que hemos encendido.



Código	Paleta 0	Paleta 1
0	Background.	Background
1	Verde.	Cyan.
2	Rojo.	Magenta.
3	Amarrillo.	Blanco.

Fig 30. Selección de color en el modo 320x200.

En el modo gráfico 640x200 a cada pixel encendido se le está permitido tener únicamente un color como foreground. El búffer del display determina si se enciende o no se enciende un pixel. Luego el "Registro para Selección de Color" selecciona el color para el foreground para cada pixel encendido.

El "Registro para Selección de Color" es un registro de 6 bits de solo escritura con una dirección del puerto de I/O en &H3D9 (en decimal equivale a un 985). Este registro puede ser escrito nuevamente utilizando el comando OUT del 8088 de I/O.
La descripción para el "Registro de Selección de Color" se da en la siguiente figura.

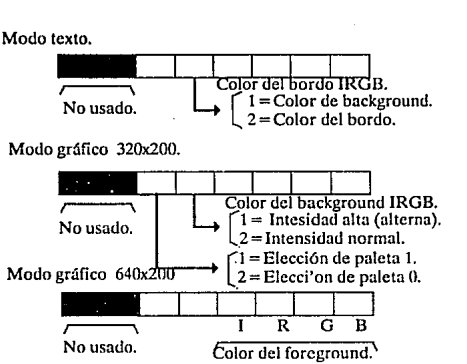


Fig 31. Descripción del registro para elección del color.

Para elegir cualquiera de las paletas se llama una rutina del ROM BIOS, la cual es conocida como "Coloca Color de la Paleta" (Set Color Palette), Para invocarla se realiza la interrupción 10H, la cual corresponde a las interrupciones del VIDEO. En seguida se da el valor de los registros para realizar esta interrupción:

Rutina Set Color Palette.

Interrupción 10H	Video.
Registro de entrada	AH = 0BH
	BH = 1 (para cambiar la paleta).
	BL = número de paleta.
Registros de Salida	Ninguno.

Iluminación de Objetos.

La iluminación de objetos requiere del uso de la proyección en perspectiva, de los algoritmos de las superficies ocultas, además del uso de algunos métodos para el sombreado e

iluminación de las superficies visibles. Como se mencionó anteriormente, una superficie es visible o invisible a un observador, de acuerdo a los criterios utilizados en el método para iluminación de superficies.

La iluminación de superficies se logra mediante el coloreado en posiciones relativas sobre la superficie, utilizando algunas propiedades o leyes de la óptica. La iluminación de las superficies se logra mediante la utilización de la escala de grises de un computador. En algunos casos iluminando con tonos fuertes, las áreas de la superficie, y en algunos otros iluminando con tonos bajos.

La intensidad de la luz depende de los tipos de fuentes de luz en la vecindad de la superficie y de las características de la superficie del objeto. Algunas veces las superficies son brillosas y en algunas ocasiones las superficies son opacas.

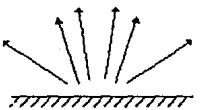
Las fuentes de luz pueden ser emisoras y reflectoras. Como ejemplo de fuentes emisoras de luz tenemos a los focos, lámparas, al sol, etc. Las fuentes reflectoras de luz generalmente son superficies iluminadas, como por ejemplo, las paredes de un cuarto, la luna, algunas estrellas, etc. Generalmente, una fuente reflectora de luz es iluminada por una fuente emisora de luz, tal y como sucede con la luna y el sol, respectivamente.

Las múltiples fuentes reflectoras de luz que intervienen en la iluminación de una superficie dan lugar a luz ambiente o luz de fondo.
Las dimensiones de una fuente de luz pueden ser grandes o pequeñas con respecto al objeto o superficie a iluminar. Si las dimensiones son grandes se tiene una fuente de iluminación distribuida, por ejemplo, el sol, un reflector gigante en un patio, etc. Si las dimensiones de la fuente son pequeñas se tiene una fuente de punto, por ejemplo, un foco en un cuarto, una lámpara de mano en un pasillo, etc.
Cuando se realiza el sombreado de una área se toman en cuenta los tipos de fuentes de luz, fuentes de luz ambiente y fuentes de punto.

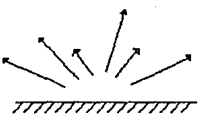
La reflexión difusa obtenida de una superficie se produce de la aspereza de la superficie. Una superficie mate produce principal-

mente reflexiones difusas, de manera que la superficie parezca igualmente brillante desde todas las direcciones de observación.

La reflexión especular se da en las superficies brillosas y se generan por las fuentes de iluminación de punto, las cuales crean toques de luz o manchas brillantes. En la siguiente figura se da un ejemplo de reflexión difusa y especular.



Reflexiones difusas de una superficie.



Reflexión especular superimpuesta en reflexiones difusas.

Fig 32. Reflexión difusa y especular.

La reflexión difusa es creada generalmente por los efectos de luz ambiente. La luz ambiente o luz de fondo es el resultado de múltiples reflexiones de objetos muy próximos. En la reflexión difusa se utiliza una intensidad uniforme en todas direcciones. Cuando la luz ambiente ilumina una superficie, esta produce un reflejo, el cual produce una iluminación uniforme desde cualquier ángulo o punto de observación. Si un objeto o superficie es expuesto a luz ambiente, la fórmula que expresaría la reflexión difusa en cualquier punto de la superficie sería:

$$I = K_d * I_a.$$

De donde:
K_d: es el coeficiente de reflexión o reflectividad, comprendido entre 0 y 1.
I_a: es la intensidad uniforme de iluminación o luz ambiente.
I : es la reflexión obtenida en la superficie.

La reflectividad K_d de una superficie, oscila entre 0 y 1, para superficies opacas la reflectividad tiende a cero, para superficies brillantes el coeficiente de reflexión tiende a uno.

La reflexión difusa originada por una fuente de iluminación se basa en la ley de los cosenos de Lambert, la cual dice que la intensidad de luz reflejada está en función del ángulo de incidencia. El ángulo de incidencia se forma entre el vector normal de la superficie y el vector dirigido hacia la fuente de luz. Una superficie perpendicular recibe más luz que una no perpendicular. La ley de los cosenos de Lambert es:

$$\cos(\theta) = N \cdot L.$$

Aplicando la ley anterior tenemos que:

$$I = K_d * I_p * (N \cdot L)/(d + d_0).$$

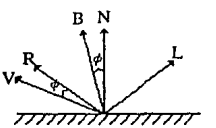
De donde:
N : es la normal unitaria de la superficie.
L : es un vector unitario en dirección a la fuente de iluminación.
K_d: es el coeficiente de reflexión.
d : es la distancia de la fuente de luz a un punto en la superficie.
I_p: es la intensidad de la fuente.
d_0: es una constante utilizada para evitar la indeterminación en la última ecuación cuando d es igual a cero.
θ : es el ángulo de incidencia.

La reflexión difusa total de una superficie, se obtiene mediante la suma de la iluminación por luz ambiente y por la iluminación por una fuente de punto, la cual está regida por:

$$I = K_d * I_a + K_d * I_p/(d + d_0) * (N \cdot L).$$

La reflexión especular se observa en ciertos ángulos de visión. Una superficie brillosa refleja la luz incidente sin importar los valores de reflectividad. Por ejemplo, una fuente de punto de color blanco nos va a producir una mancha blanca sobre la superficie en donde se observe la reflexión especular. En relación con un reflector ideal (espejo perfecto) el ángulo de incidencia y el ángulo de reflexión especular son iguales.

La reflexión especular se puede observar en la siguiente figura:



El vector B está a lo largo de la bisectriz del ángulo formado entre los vectores V y L.

Fig 33. Relación de la variables en la reflexion espec.

De donde:
R : es el vector unitario en dirección de la reflexión especular.
L : es un vector unitario que apunta hacia la fuente de luz.
V : es un vector unitario en dirección del observador.
N : es el vector normal unitario de la superficie.

Los objetos reales exhiben la reflexión especular en una diversidad de posiciones en torno al vector R. Las superficies brillosas tienen un intervalo de reflexión más amplio.

Un método para modelar la reflexión especular fue creado por Phong Bui Toung, y llamado modelo de Phong, el cual hace la intensidad de la reflexión especular proporcional al $\cos^n(\theta)$. El valor asignado a n nos va a determinar el tipo de superficie que se observará. Una superficie muy brillante se modela con una valor grande de n (200 o más) y se utilizan valores menores en superficies más opacas (hasta 1). Para un reflector perfecto n tiende a infinito. En superficies ásperas, como por ejemplo plástico, n recibe un valor próximo a 1.

La reflexión especular también depende del ángulo de incidencia. En general, disminuye cuando el ángulo de incidencia aumenta. La fórmula que rige la reflexión especular está dada por:

$$I = K_d * I_a + I_p/(d + d_0) * [K_d * (N \cdot L) + w(\theta) * \cos^n(\phi)].$$

Para fines prácticos $w(\theta)$ toma valores comprendidos entre 0 y 1, por otro lado $\cos(\phi)$ se puede calcular con V·T, obteniendo de esta forma:

$$I = K_d * I_a + I_p/(d + d_0) * [K_d * (N \cdot L) + K_s * (V \cdot R)^n].$$

Algunas consideraciones que se pueden hacer cuando d es muy grande es que (N·L) y (V·R) son constantes, debido a que N y L son casi paralelos y V·R siempre van a ser constantes.
Existen otros métodos para realizar la iluminación como el de Torrance y Sparrow.

APENDICES.

Apéndice A: El Ambiente de C y la Graficación Por computadora.

Las ventajas de C para la Graficación son la Versatilidad, el Poder y la Velocidad.

La Versatilidad de C se encuentra en el manejo de memoria y en el control de los procesos. Es una elección sabia en la programación de gráficos.

El manejo de memoria es de vital importancia debido a que las imágenes gráficas son almacenadas como bloques de datos en memoria, y por que los valores numéricos usados para el despliegue de las imágenes gráficas son frecuentemente almacenados, como base de datos en la memoria.

La versatilidad de C para el manejo de la memoria involucra la habilidad de organizar la memoria de la computadora de diferentes maneras, además de encontrarse con diferentes necesidades en los programas de Graficación. Algunos programas utilizan grandes cantidades de datos de código ejecutable y muy poca cantidad de datos. C puede acomodar estas necesidades. Dentro de Tubo C++ tenemos los siguientes modelos: Tiny, Small, Compact, Medium, Large y Huge. Estos modelos pueden solucionar los casos antes mencionados.

La versatilidad de C en el manejo de la memoria también incluye la habilidad de mover rápidamente el contenido de un bloque de memoria a otro bloque. Esta habilidad es de gran importancia en las animaciones.

El control en el proceso es de gran importancia por que los programas de graficación pueden saltar a subrutinas, emplear

loops y contadores, y aceptar el ingreso de la información por teclado. Las instrucciones para el control del proceso del lenguaje C son ricas y variadas.

El Poder de C está contenido en las características del lenguaje. Las cuales son: bajo, mediano y alto nivel.

Un lenguaje de bajo nivel consisten en un lenguaje de programación cuya sintaxis muestra la operación de los registros internos en el hardware de la computadora. Frecuentemente, se dice que un lenguaje de bajo nivel opera a nivel de bit. Un lenguaje de bajo nivel ofrece potencia áspera, además de que el código fuente es difícil de leer. Si se usa un lenguaje de bajo nivel, los programas pueden hacer cualquier cosa de lo que el hardware pueda hacer.

Un lenguaje de nivel medio nos da un conjunto de instrucciones que parecen menos precisas en cuanto a la operación de los registros. A un lenguaje de nivel medio también se le conoce como lenguaje a nivel de bytes. Los programas escritos en lenguajes de medio nivel, tales como Pascal, DBase, etc, son fáciles de leer.

Un lenguaje de alto nivel utiliza una sintaxis mediante palabras extraídas del inglés. Una instrucción causará que una computadora realice una serie de funciones de bajo y mediano nivel.

Un lenguaje de alto nivel es fácil de usar, también es conocido como lenguaje de rutinas-orientadas. C es un lenguaje híbrido, y cae en tres categorías de programación: bajo, medio y alto nivel. Esta característica ofrece muchas ventajas al programador de Graficación, especialmente, cuando muchas de las instrucciones son orientadas a gráficos.

La Velocidad de C nos produce un código de corrida rápida, debido a la relación existente con el lenguaje ensamblador. Se ofrece una variedad de rutinas especializadas llamadas funciones. La especialización significa velocidad, lo cual es el alma de los programas de Graficación.

El ambiente típico para la programación utilizando dos drives y trabajando con DOS, además de utilizar QuickC de Microsoft, consiste de algunos pasos. En el primero, la computadora es inicializada mediante un disco con DOS, el segundo paso, con-

siste en colocar un disco en el drive A con QuickC (QB.EXE). El editor de QuickC es invocado del disco en el drive A. En el tercer paso, el disco que contiene los archivos #include son colocados en el drive A, de donde QuickC obtiene los archivos solicitados durante el proceso de compilación.

El disco que contienen los archivos fuentes QLB, en el último paso, son colocados en el drive B. El archivo QLB es cargado a la memoria alta en la RAM cuando por primera vez QuickC es invocado. Las rutinas de graficación que realizan el dibujo de imágenes se encuentran en la librería QLB.

Al igual que en QuickC, consideremos un sistema de dos drives. El primer paso a seguir en un ambiente típico de programación para Turbo C es cargar DOS en el drive A. El segundo paso consiste en arrancar Turbo C en el drive A, mediante TC.EXE, posteriormente, TC.EXE invoca a su editor. El disco de librerías conteniendo algunos archivos #include, librerías para graficación, librerías matemáticas son colocadas en el drive A. Estas rutinas son obtenidas para Turbo C durante la fase de compilación.

Como se mencionó anteriormente, C es capaz de soportar diferentes tipos de ambiente en el manejo de memoria para los programas de graficación. Estos ambientes en la memoria son llamados Modelos de Memoria. Los dos modelos comúnmente utilizados en la realización de programas para graficación son el modelo Mediano y el modelo Compacto. Como también se mencionó anteriormente otros modelos son también disponibles.

El modelo mediano nos da un máximo de 64K bytes para datos y un máximo de 64K bytes en el stack, por otro lado nos da un tamaño limitado en el código ejecutable. Por default, el tamaño del stack para QuickC es 2K y para Turbo C es de 4K. La versión 1.00 para QuickC puede únicamente operar en el modelo mediano. Turbo C puede ser configurado a través de su menú de ventanas para operar en los modos antes mencionados, incluyendo el modelo mediano.

El modelo compacto limita el código ejecutable a 64K bytes, pero nos permite un ilimitado tamaño en los datos. El stack máximo es nuevamente de 64K. Este modelo de memoria tiene

aplicaciones especializadas. Nuevamente Turbo C puede ser configurado utilizando sus menús. La versión 2.00 de QuickC soporta todos los modelos de memoria en un ambiente de programación integrado.

Apéndice B: Algunas Rutinas del Proyecto GPC.

Este software educativo se encuentra formado por decenas de rutinas, a continuación se da una breve descripción de las rutinas más importantes, así como la descripción generalizada de la operación del programa en un diagrama de bloques.

- InicializaModo: Rutina que nos pone en modo gráfico.
- InicializaCursor: Rutina que obtiene la imagen del cursor, para usos posteriores.
- Presentación: Rutina que nos da la presentación inicial en el sistema.
- ReporteDelStatus: Nos da información del estado en como se encuentra operando el sistema, como por ejemplo: modo gráfico, tipo de tarjeta, resolución, cuantos colores disponibles existen, etc.
- Menu: Este módulo es la parte central de este sistema. Nos maneja todas las funciones del sistema como lo son ayudas, pseudocódigo, ejemplos, etc.
- ColocaVentanaActiva: Rutina que nos dibuja las ventanas de trabajo, la ventana actual la marca con un color más resaltado.
- DibujaBordo: Rutina que dibuja el bordo del puerto o ventana actual.
- CambiaEstiloDeTexto: Rutina que cambia a un estilo nuevo de texto.
- LlenaPuerto: Rutina que nos borra el contenido de la ventana actual, iluminándolo de un color.
- TodoElPuerto: Módulo que activa una ventana del tamaño de la pantalla.
- PuertoDeTrabajo: Rutina que activa una ventana para trabajo. En esta ventana se realizan todos los dibujos.
- RutinaCursor. Obtiene un punto P(x,y).

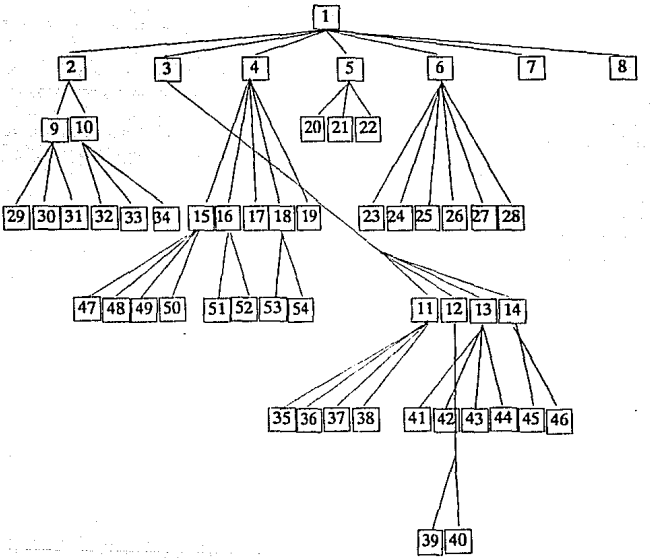
- Poligono: Este módulo obtiene datos sobre polígonos en 2D.
- DibujaPoligono: Dibuja un polígono en la ventana actual.
- LineaDDA: Este módulo dibuja una línea por el método del Analizador Diferencial Digital.
- LineaBRESENHAM: Rutina que dibuja una línea por el método de Bresenham.
- LineaParametrica: Módulo que dibuja una línea recta utilizando la ecuación paramétrica de la línea recta.
- CirculoBRESENHAM: Dibuja un círculo utilizando el método de Bresenham.
- CirculoDDA: Dibuja un círculo utilizando el método del Analizador Diferencial Digital.
- Traslación: Este módulo realiza el cambio de posición de un polígono.
- Traslada: Rutina utilizada por la traslación en 2D y 3D.
- Escalamiento: Este módulo realiza la amplificación de un polígono con respecto a un punto de referencia.
- Rotacion: Realiza el giro de un polígono con respecto a un punto de referencia.
- Espejo2D: Esta rutina nos realiza el espejo de un polígono con respecto al eje "x","y" y origen.
- Rellenado: Llena un polígono por el método de la línea de rastreo.
- RellenadoPorSemilla: Llena un polígono utilizando una semilla.
- Bezier: Esta rutina dibuja una curva, por el método del ingeniero Francés Bezier.
- Curva: Este módulo obtiene el contorno de la curva, en forma de líneas rectas, para que posteriormente cualquiera de los métodos que dibujan curvas la realicen. Obtiene los puntos de control de la curva.
- DibujaCurva: Esta rutina dibuja el contorno o la unión entre los puntos de control.
- Bspline: Dibuja una curva utilizando el método Bspline.
- Hermite: Dibuja una curva Hermite utilizando métodos numéricos.

- BetaSpline: Dibuja una curva BetaSpline utilizando los parámetros β_1 y β_2.
- RecorteSutherland: Realiza el recorte de un polígono utilizando como ventana un rectángulo o un polígono mayor a 2 lados.
- Volumen: Este módulo obtiene un volumen de varios volúmenes predefinidos.
- DibujaVolumen: Dibuja el volumen con el que se está trabajando, en el puerto o ventana activa.
- Rotación3D: Realiza el giro de un volumen sobre cualquiera de los tres ejes cartesianos x,y o z.
- Traslación3D: Efectúa el cambio de posición en dirección de los ejes cartesianos x,y o z.
- DibujaEjes3D: Módulo que dibuja los ejes cartesianos para 3D.
- Espejo3D: Este módulo realiza el espejo del volumen con respecto a los planos x, y y z.
- Escalacion3D: Realiza el incremento en el tamaño de un volumen, lo amplifica.
- Paralela: Realiza la proyección paralela de un volumen. Obtiene el isométrico de un volumen.
- Perspectiva: Realiza la proyección perspectiva de un volumen, situando al observador sobre el eje z.
- ColorDeTrazo: Rutina que cambia el color en el trazo de polígonos, curvas, líneas, círculos, etc.
- ObtenImagen: Rutina que obtiene la imagen de la ventana actual.
- LiberaImagen: Rutina que coloca una imagen grabada en la ventana actual.
- Clip3D: Recorte de una línea recta en tercera dimensión, teniendo como ventana un paralelepípedo rectangular.
- Info: Da información de ayudas y pseudocódigos.
- LineasOcultas: Realiza la supresión de líneas ocultas en volúmenes.
- Salida: Rutina que nos da la salida del sistema GPC.
- Impresión: Esta rutina nos manda a impresora el contenido del puerto de trabajo.

- Mandelbrot: Esta rutina obtiene una curva fractal, a partir de la inversa de una función compleja.
- Snowflake: Rutina que genera una curva fractal utilizando un iniciador y un generador.
- Pharaoh: Nos genera una curva fractal utilizando una función inversa, además de la utilizanción de círculos que utiliza para su generación.

En la siguiente página se da el diagrama de bloques generalizado del programa GPC. Por razones de visibilidad se utilizan códigos asociados a cada bloque.

Diagrama de bloques del proyecto GPC.



De donde los símbolos asociados a los bloques son los siguientes:

1: Rutina principal.
2: Primitivas gráficas..
3: Dos dimensiones.
4: Tres dimensiones.
5: Curvas realizadas por fractales.
6: Utilerias del programa GPC.

7: Demostración del sistema.
8: Salida del sistema.
9: Líneas.
10:Círculos.
11: Transformaciones en 2D.
12: Recorte.
13: Curvas.
14: Rellenado.
15: Transformaciones.
16: Proyecciones.
17: Recorte en 3D.
18: Eliminación de superficies.
19: Iluminación de volúmenes.
20: Fractal por el método de Mandelbrot.
21: Fractal por el método de snowflake.
22: Fractal por el método de pharaoh.
23: Obtención de un polígono.
24: Obtención de un volumen.
25: Obtención de una curva.
26: Información del sistema.
27: Limpiar pantalla.
28: Selección de color.
29: Línea DDA.
30: Línea por el método paramétrico.
31: Línea por el método de Bresenham.
32: Círculo por el método DDA.
33: Círculo por el método paramétrico.
34: Círculo por el método de Bresenham.
35: Desplazamiento en 2D.
36: Escalamiento en 2D.
37: Rotación en 2D.
38: Espejo en 2D.
39: Recorte rectangular.
40: Recorte poligonal.
41: Curva Hermite.
42: Curva por el método de Bezier.
43: Curva por el método Bspline.
44: Curva por el método Betaspline.
45: Rellenado por el método de la línea de barrido.
46: Rellenado por le método de la semilla.
47: Desplazamiento en 3D.
48: Escalación en 3D.

49: Rotación en 3D.
50: Espejo en 3D.
51: Proyección paralela.
52: Proyección en perspectiva.
53: Eliminación de caras en volúmenes.
54: Eliminación de superficies en áreas.

BIBLIOGRAFIA.

Libros.

* Illumination and Color In Computer Generated Imagery.
Ray Hall.
Springer-Verlay.

* An Introduction to Splines for Use in Computer Graphics & Geometric Modeling.
Richard H. Bartels, John C. Beatty, Brian A. Barsky.
Forwards by Pierre Bezier & A. Robin Forrest.

* Mathematical Elements for computer Graphics.
David F. Rogers, J. Alan Adams.
Second Edition.
Mc. Graw-Hill Publishing Company.

* Gráficas Por Computadora.
Donald Hearn, M. Pauline Baker.
Prentice-Hall Hispanoamericana, s,a.
Impreso en México.

* High Performace Cad Graphics in C.
Lee Adams.
Published by Wind Crest.
First edition/Second Printing.

* Principles of Interactive Computer Graphics.
Newman, W. M., Sproull R.F.
Mc. Graw Hill.
N.Y. 1979.

* Fundamentals of Interactive Computer Graphics.
Foley. J. V., Van Dam A.

The System Programming Series 1981.

* Introduction to Interactive Computer Graphics.
Scott J. E.
Wiley, 1982.

* Procedural Elements for Computer Graphics.
David F. Rogers.
Mc. Graw Hill, 1985.

* Tutorial: Computer Graphics.
John C. Beatty & Kellogs Booth
IEEE Computer Society Press.
1982 EUA.

* Computer Graphics.
Francis S. Hill, Jr.
Department of Electrical & Computer Engineering
University of Massachusetts.
Macmillan Publishing Company New York.
Collier Macmillan Publishers.
London.

* Computer Graphics Systems & Concepts.
Rod Salmon & Mel Slater.
Addison-Wesley Publishing Company.
First Printed 1987.
Printed in Great Britain by The Bath Press, Avon.

* Computer Graphics: A Revolution In Design.
R. A. Siders, D. R. Drane, L. W. Ebrhardt, P. Ghent,
R. G. C. Hanna.
American Management Association New York.

* Interactive Microcomputer Graphics.
Chan S. Park.
Addison-Wesley, USA.

* Course Notes.
Introduction to Ray Tracing Siggraph '88.
Andrew S. Glassner.
1987.

* Tablas Matemáticas.
Arquímides Caballero C.
Lorenzo Martínez C.
Jesús Bernardez G:
Vigésima Octava Edición.
Editorial Esfinge S.A.
Colima 220-503 México 7 D.f 1983.

* Análisis Numérico.
Richard L. Burden.
J. Douglas Faires.
Grupo Editorial Iberoamericano.
Escuela Superior de Física Matemáticas-IPN.
Impreso en México.

* The Waite Grup's Microsoft C Programing for the PC
Robert Lafore.
Revised Edition.
Howard W.Sams & Company.

Revistas.

* IEEE Computer Graphics and Aplications.

* ACM Computing Surveys.

* ACM Siggraph.