Instituto Tecnológico de Morelia

"José María Morelos y Pavón"

GRAFICACIÓN

Martha Azucena Fernández Saucedo

Asesor: Rogelio Ferreira Escutia

Trabajo realizado para obtener el título de:

Ing. en Sistemas Computacionales

Morelia, Mich. Agosto de 2007

A mi familia

Índice

Índice de figuras

Capítulo 1

Introducción

La infografía, es decir, los gráficos por computadora, continua siendo una de las áreas más excitantes y de más rápido crecimiento de la tecnología moderna. Los métodos infográficos se aplican de forma rutinaria en el diseño y la mayoría los productos, en los simuladores para actividades de programación, en la producción de películas, anuncios de televisión, vídeos musicales, en el análisis de los datos, en los estudios científicos, en las intervenciones médicas, los vídeo juegos y en muchísimas otras aplicaciones.

En la actualidad, se utiliza una gran variedad de técnicas y de dispositivos de hardware en diversas áreas de aplicación y existen muchas técnicas y hardware en proceso de desarrollo también. Buena parte de las investigaciones actuales en el campo de la informática están relacionadas con la mejora de la efectividad, el realismo y la velocidad de generación de imágenes.

La tendencia actual en los gráficos por computadora consiste en incorporar mejores aproximaciones de los principios físicos dentro de los algoritmos gráficos a fin de simular mejor las complejas interacciones existentes entre los objetos y el entorno.

El objetivo del texto presente es introducir al estudiante a los conceptos básicos, en teoría y práctica, de la graficación mediante la explicación detallada de los principios de esta.

1. Introducción a la graficación por computadora

Objetivo. *El estudiante conocerá los antecedentes de la graficación así como los principales formatos de almacenamiento*.

Los gráficos por computadora se han convertido en una potente herramienta para la producción rápida y económica de imágenes. Prácticamente no existe ninguna tarea en la que la representación gráfica de la información no pueda aportar alguna ventaja y por tanto, no sorprende encontrar gráficos por computadora en muchos sectores.

Aunque las primeras aplicaciones de ciencia e ingeniería requerían equipos caras y aparatosos, los avances en la tecnología informática han hecho los gráficos interactivos una herramienta muy útil. Actualmente los gráficos por computadora se usan en campos tan diversos como las ciencias, las artes, la ingeniería, los negocios, la industria, la medicina, las administraciones públicas, el entretenimiento, la publicidad, la educación, la formación y en aplicaciones caseras.

1.1.Breve historia de la graficación

Las bases

A lo largo de la historia han sucedido importantes eventos que han sentado las bases para las gráficas por computadora. Estos descubrimientos merecen ser mencionados así como sus autores:

- Euclides (300- 250 A.C.), su fórmula de geometría provee una base para los conceptos gráficos.
- Filippo Brunelleschi (1377 - 1446), arquitecto y escultor, es reconocido por su uso de la perspectiva.
- Rene Descartes (1596-1650), hizo aportaciones a la geometría analítica, en particular, el sistema de eje coordenado que provee una base para describir la localización y forma de objetos en el espacio.
- Gottfried Wilhelm Leibniz (1646 - 1716) e Issac Newton (1642 - 1727), inventaron el calculo que permite la descripción de sistemas dinámicos.
- James Joseph Sylvester (1814 - 1897), inventó la notación matricial. Los gráficos son hechos con matrices.
- I. Schoenberg, descubrió los splines, un tipo fundamental de curva.
- J. Presper Mauchly (1919 - 1995) y John William Mauchly (1907 - 1980), construyeron la computadora ENIAC
- Alexander S. Douglas desarrolla en 1952 Nought and crosses. El juego era una versión computerizada del tres en raya que se ejecutaba sobre el EDSAC.

Los 1960s

Se puede decir que la historia comienza con el Proyecto Whirlwind y el sistema computacional SAGE que fue diseñado para apoyar el estado de alerta militar. El Proyecto Whirlwind inició como un esfuerzo para construir un simulador de vuelo y SAGE para proveer un sistema de defensa aéreo en los Estados Unidos como protección contra un ataque nuclear. La estación de trabajo SAGE tenía un monitor vectorial y lápiz luminoso que los operadores usaban para dibujar planes de vuelo sobre las regiones de los Estados Unidos. En la actualidad, el Museo de Computación de Boston (Boston Computer Museum) exhibe una estación de trabajo SAGE. El monitor es una pantalla de radar con un recuadro alrededor de la región que esta siendo escaneada. Los lápices luminosos son como los viejos taladros de metal. En la exhibición “Hollywood and Vine” de IBM en Kingston New York se muestra una computadora SAGE.

Figura 1.1.Computadora de uso militar SAGE

Además de el inicio de la era de las primeras computadoras de tubos de vacío, los 1940s vieron nacer el transistor en los Laboratorios Bell (Bell Labs) en 1947. En 1956 la primera computadora de transistores se construyo en el MIT.

IBM, Sperry-Rand, Burroughs y otras pocas compañías de computadoras existían a principios de los 1960s. Las computadoras tenían unos pocos kilobytes de memoria, ni hablar de sistemas operativos o monitores que desplegaran gráficos.

Los periféricos eran tarjetas perforadas de Hollerit, impresoras de líneas y plotters de papel en rollo. Los únicos lenguajes de programación eran ensamblador, FORTRAN y Algol. Las funciones gráficas y los calendarios Snoopy eran los únicos gráficos hechos hasta entonces.

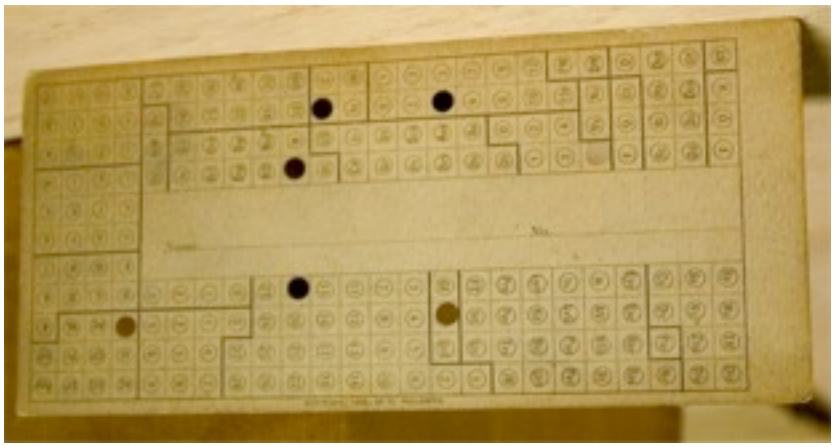


Fig. 1.2.Tarjeta perforada de Hollerit

En 1963 Ivan Sutherland presentó su artículo Sketchpad en el Summer Joint Computer Conference. Sketchpad permitía el diseño interactivo en un monitor de gráficos de vectores con un lápiz luminoso como dispositivo de entrada. La mayoría de la gente ubica este hecho en la historia como el origen de las gráficas por computadora. Sketchpad se considera el abuelo de los programas CAD actuales y era muy avanzado para su época al usar una interfaz gráfica de usuario, GUI.

Fig. 1.3.Ivan Sutherland trabajando en el MIT

El sistema Sketchpad fue creado en el Laboratorio Lincoln del MIT sobre un ordenador TX-2, una de las mejores máquinas de entonces pues contaba con 320Kb de memoria base y 8Mb de memoria externa en forma de cintas magnéticas. En cuanto a dispositivos gráficos, estaba dotado con un monitor de siete pulgadas 1024×1024, además de un puntero óptico y un remedo de ratón.

Jack Bresenham nos enseñó como dibujar lineas en un dispositivo raster a mediados de los 1960s. Después extendió su método para dibujar círculos. Larry Roberts señaló la utilidad de las coordenadas homogéneas, las matrices 4 x 4 y los algoritmos de detección de líneas ocultas. Stevens Coons introdujo las superficies paramétricas y desarrolló los primeros conceptos de diseño geométrico asistido por computadora.

El trabajo de Pierre Bézier en curvas paramétricas también se hizo público. Andrew Appel desarrolló en IBM algoritmos para superficies ocultas y sombras que fueron precursoras de el trazado de rayos (ray tracing). La transformada rápida de Fourier fue descubierta por Cooley y Tukey. Este algoritmo nos permite entender mejor las señales y es fundamental para el desarrollo de técnicas de anti-alisado.

En 1963, Douglas Englebart inventó el Mouse en los laboratorios de Xerox PARC. Este mouse usaba dos ruedas perpendiculares entre ellas: la rotación de cada rueda era trasladada en movimiento a lo largo de un eje en el plano.

Fig. 1.4.Primer mouse sujetado por su inventor, Douglas Englebart

Años más tarde, la corporación Evans & Sutherland y General Electric comenzaron a construir simuladores de vuelo con gráficos rasterizados de tiempo real.

En 1966 Ralph Baer empezó a desarrollar junto a Bob Tremblay y Ted Dabney un proyecto de videojuego llamado Fox and Hounds dando inicio al videojuego doméstico. Este proyecto evolucionaría hasta convertirse en la Odyssey, el primer sistema doméstico de videojuegos lanzado en 1972.

El algoritmo de sombreado fue creado por Gouraud y Phong en la Universidad de Utah (University of Utah) cuando en 1968 la universidad le pidió a David Evans que dirigiera un programa de investigación en gráficos por computadora; además del sombreado, otros algoritmos se desarrollaron tales como el de iluminación.

Phong también introdujo un modelo de reflexión que incluía rayos de luz especulares. Se hicieron demostraciones de animación basada en cuadros clave para gráficos en 3D . Xeroc PARC desarrolló un programa de dibujo o editor de mapa de bits llamado Markup para la computadora Alto.

Ed Catmull introdujo el rendereo paramétrico, el algoritmo de z-buffer y mapeo de textura. En esa misma época BASIC, C y Unix fueron desarrollados en los laboratorios Dartmouth y Bell.

Los 1970s

A principios de los 1970s el Sistema de Imágenes de Evans & Sutherland era una computadora high-end de gráficos. El primer microprocesador para computadoras fué creado en Intel en 1971, este era de 8 bits e inicio la serie de los “8”: 8088, 8086, 80186, etc.

Los videojuegos como arcade nacieron en 1971 cuando Nolan Bushnell comenzó a comercializar Computer Space, una versión de Space War, en Estados Unidos, aunque es posible que se le adelantara Galaxy War otra versión arcade de Space War aparecida a principios de los 70 en el campus de la universidad de Stanford.

La eclosión de los videojuegos llegó con Pong, un videojuego muy similar Tennis for Two y diseñador por Al Alcorn para Bushnell. El juego se presentó en 1972 y fue la piedra angular del videojuego como industria. Durante los años siguientes se implantaron numerosos avances técnicos en los videojuegos (destacando los microprocesadores y los chips de memoria), se pusieron a la venta juegos como Space Invaders (Taito) o Asteroids (Atari) y sistemas como el Atari 2600.

Fig. 1.5.Ipod con el juego Pong

Fig. 1.6.Consola de video juegos Atari 2600

La Altair 8800 de MITS fue un microcomputador diseñado desde 1975, basado en el CPU Intel 8080A. Vendido como un kit a través del la revista Popular Electronics, los diseñadores proyectaron vender solamente algunos cientos a los aficionados, y fueron sorprendidos cuando vendieron sobre diez veces esa cantidad en el primer mes. Hoy en día, la Altair es ampliamente reconocida como la chispa que condujo a la revolución del computador personal de los años siguientes: El bus de computador diseñado para la Altair se convirtió en un estándar de facto conocido como el bus S-100. El primer lenguaje de programación para la máquina fue el Altair BASIC, escrito por Bill Gates y Paul Allen, quienes inmediatamente después fundarían Microsoft.

Fig. 1.7.Computadora Altair 8800

En 1976 la computadora Apple I fue el primer éxito comercial de la computación personal. El Apple I fue uno de los primeros computadores personales, y el primero en

combinar un teclado con un microprocesador y una conexión a un monitor. El Apple I fue diseñado por Steve Wozniak originalmente para uso personal. Un amigo de Steve Wozniak, Steve Jobs, tuvo la idea de vender el computador. Fue vendido como el primer producto de Apple, comenzando en abril de 1976. Su precio al por menor era US$666.66. Cerca de 200 unidades fueron producidas.

Fig. 1.8.Computadora Apple I

Es común pensar que el primer filme que usó gráficas por computadora es "2001: Una odisea en el espacio" (1968), la cual pretendía mostrar cómo las computadoras se volverían más gráficas en el futuro. Sin embargo, los efectos por computadora en esa película fueron animados a mano y las secuencias de efectos especiales fueron producidas con óptica y efectos convencionales.

Quizá el primer uso de las gráficas por computadora, específicamente la ilustración de gráficas por computadora fue en "Futureworld" (1976), que incluyó la animación de un rostro y mano humanos producida por Ed Catmull y Fred Parke en la Universidad de Utah (University of Utah).

Los 1980s

La IBM PC comenzó a ser vendida en agosto de 1981. La frase "computadora personal" era de uso corriente antes de 1981, y fue usada por primera vez en 1972 para denominar al Xerox PARC's Alto. Sin embargo, debido al éxito del IBM PC, lo que había sido

un término genérico llegó a significar específicamente una computadora compatible con las especificaciones de IBM.

La PC original fue un intento de IBM para entrar en el mercado de los ordenadores domésticos, entonces dominado por el Apple II de Apple Computer y varias máquinas con CP/M. En lugar de utilizar el proceso de diseño normal de IBM, el cual ya había fallado en el diseño de una computadora económica (como el IBM 5100), se reunió a un equipo especial para descartar las restricciones de la compañía e ingresar rápidamente al mercado. Al proyecto se le dio el nombre código de Project Chess (proyecto ajedrez).

Fig. 1.9.Primera computadora personal IBM

Las computadoras con Mouse, monitores rasterizados y red ethernet se hicieron estándar en el área académica, las ciencias y la ingeniería.

Cerca de 1983,Jim Blinn introdujo los conceptos de mapeo de textura tales como el bump mapping. El bump mapping (Correlación de relieve), una técnica de perturbación normal(la dirección hacia donde apunta un polígono) solía simular superficies desiguales o arrugadas y con relieve.

Fig. 1.10.Jim Blinn

Se introdujeron los Binary space partitioning o Particionado Binario del Espacio (BSP), que es un método para subdividir recursivamente un espacio en elementos convexos empleando hiperplanos. Esta subdivisión permite obtener una representación de la escena mediante un árbol conocido como Árbol BSP. Estos se utilizan en renderizados y en las aplicaciones CAD.

Loren Carpenter comenzó a explorar fractales en gráficos por computadora a principios de los 1980s. El es un cofundador de los Estudios de Animación Pixar. En 1982 John Warnock desarrolló el Postcript aunque Warnock es más conocido por ser cofundador de Adobe Systems Inc., se retiró en el 2001 de la dirección de Adobe.

Fig. 1.11.John Warnok, cofundador de Adobe

En 1982 se lanzó la tarjeta Hércules, la primera tarjeta de video. Esta tarjeta se podía

visualizar gráficos y textos simultáneamente. En modo texto, soportaba una resolución de 80x25 puntos. En tanto que en los gráficos lo hacía con 720x350 puntos, dicha tarjeta servía sólo para gráficos de un solo color. La tarjeta Hércules tenía una capacidad total de 64k de memoria video RAM. Poseía una frecuencia de refresco de la pantalla de 50HZ.

En 1983 Jobs se dio cuenta de que el futuro eran los iconos y los entornos agradables, y los patrocinó creando el Apple Lisa, primer ordenador comercial con interfaz gráfica y ratón, y posteriormente en 1984 el primer Apple Macintosh.

El 28 de junio de 1985, un par de años después del primer aviso de Windows en noviembre de 1983 se lanzo Microsoft Wndows 1.0. En la caja azul se anunciaba una interface gráfica de Windows con las ventanas embaldosadas. La estrategia de venta de Microsoft para Windows era proporcionar un nuevo ambiente de desarrollo y un nuevo entorno de software en el que se utilizan imágenes de mapa de bits y un ratón.

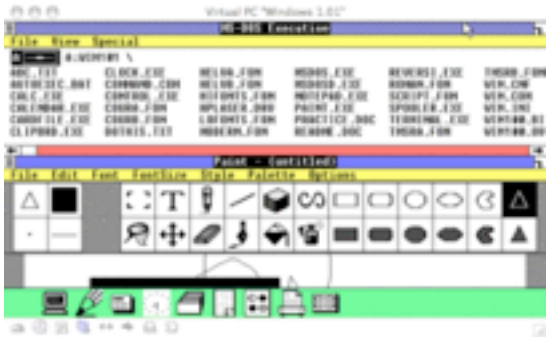


Fig. 1.12.Ventana en la primera versión de Microsoft Windows

El aloritmo radiosity fue creado por Donald Greenberg y sus colegas en la Universidad de Cornell. Radiosity una técnica para la iluminación global que usa la teoría de transferencia de radiación para simular la iluminación (reflejada) indirecta en escenas con superficies difusas.

En 1989 Adobe Photoshop comenzó a ser comercializado siendo hoy una de las aplicaciones clásicas para el manejo de imágenes. Aunque en su primera versión, solo estaba disponible para Macintosh. Photoshop es uno de los productos más vendidos por Adobe quien recientemente compró a Macromedia.

Fig. 1.13.Programa Adobe Photoshop

Unix y X window eran las plataformas de elección para la programación en C y C++, pero MS-DOS comenzó a crecer.

La tarjeta VGA (Video Graphics Array) fue creada en 1987 por IBM. Las estaciones de trabajo de Silicon Graphics que soportaban rasterizado de dibujo de líneas en tiempo real se convirtieron en el deseo de los desarrolladores de gráficas. Los circuitos VLSI (Very Large Scale Integration) para procesadores gráficos y procesamiento paralelo se convirtieron en áreas de investigación importantes a finales de los 1980s.

Los 1990s

Unix, X y Silicon Graphics GI eran el sistema operativo, sistema de ventanas y la Interfaz de Programación de Aplicaciones (API, Application Programming Interface) que los desarrolladores de gráficos utilizaban a principios de los 1990s.

En 1991 las computadoras Hand-held se inventaron en HP (HewlettPackard). En 1992 OpenGL se convirtió en un estándar de APIs gráficas. OpenGL es una biblioteca gráfica desarrollada originalmente por Silicon Graphics Incorporated (SGI). OpenGL significa Open Graphics Library, cuya traducción es biblioteca de gráficos abierta.

Las gráficas rasterizadas sombreadas comenzaron a introducirse en las películas. Las computadoras aún no soportaban gráficos 3D y la mayoría de los programadores escribía software para ser convertidos por escaneo o rasterizados y utilizaban algoritmos de remoción de superficies ocultas así como trucos de animación de tiempo real.

Mosaic fue creado en 1993 por Marc Andressen en la Universidad de Illinois (University of Illinois) en el Centro Nacional para Aplicaciones Científicas (NCSA). ue el primer navegador gráfico disponible para visualizar páginas web.

Fig. 1.14.Imagen del primer navegador gráfico Mosaic

Con el surgimiento de Mosaic, se popularizó el uso de la Web a pesar de que fue creada años antes. La Web nació alrededor de 1989 a partir de un proyecto del CERN, en el que Tim Berners-Lee construyó el prototipo que dio lugar al núcleo de lo que hoy es la World Wide Web. La intención original era hacer más fácil el compartir textos de investigación entre científicos y permitir al lector revisar las referencias de un articulo mientras lo fuera leyendo.

En esta época, se comenzaron a utilizar los estándares de MPEG para la compresión de video. Los sistemas dinámicos que permiten animación con colisiones, gravedad, fricción y causa y efecto también comenzaron a usarse

En 1996 sale al mercado Nintendo 64; en su primera semana de ventas en Japón, Nintendo 64 vendió 500.000 consolas y durante su periodo de vida alrededor de 30.000.000.000 consolas.

También fueron lanzados varios periféricos: un micrófono que permitía jugar un juego con funciones de reconocimiento de voz, uno que permitía leer cartuchos de Game Boy,

una unidad de Disquetes y un cartucho que permitía capturar audio y video a los disquetes. Sin embargo el mayor problema de estos periféricos fue el poco soporte que tuvieron.

Fig. 1.15.Super Mario 64, popular videojuego para la consola Nintendo 64

Super Mario 64 programado por Shigeru Miyamoto (para Nintendo 64) es considerado por muchos el mejor juego de plataformas de todos los tiempos y su sistema de juego es la base de la mayor parte de los juegos de plataformas 3D de hoy en día. Este juego creó el primer sistema de control de cámaras en un juego en 3 dimensiones.

Fig. 1.16.Shigeru Miyamoto, creador de Supermario 64

Las tarjetas gráficas tales como la 3DFX de Nvidia se comenzaron a popularizar en 1995. Las tarjetas Nvidia utilizan el puerto AGP inventado por Intel en 1996 como solución a los cuellos de botella que se producían en las tarjetas gráficas que usaban el bus PCI.

El formato de DVD se anuncio en septiembre de 1995; la especificación oficial fue desarrollado por un consorcio de 10 compañías: Hitachi, JVC, Matsushita, Mitsubishi, Philips,

Pioneer, Sony, Thomson, Time Warner, y Toshiba . Al inicio, existían varias propuestas de diferentes compañías y debido a la incompatibilidad se realizó el estándar.

La realidad virtual y el VRML (Virtual Reality Modeling Language) se convirtieron áreas de investigación importante. Los PDAS, Palms y los flat panel se introdujeron al final de los 1990s con gran éxito.

2000 - a la fecha

En la actualidad la mayoría de las personas que trabajan con gráficos utilizan computadoras de grandes capacidades: discos duros de terabytes, tarjetas gráficas aceleradoras de video con memoria en gigabytes, mouse óptico y memoria RAM en el orden de los gigas. También son muy utilizadas las computadoras Macintosh especialmente en lo relacionado a efectos especiales y gráficos de animación.

Fig. 1.17.La computadora PowerMac es popular para las aplicaciones de animación

Los procesadores ahora tienen núcleo doble que dota a las aplicaciones de recursos que permiten hacerlas más sofisticadas.

Aunque las cámaras digitales aparecieron en los 1990s, es hasta ahora que comienzan a popularizarse, existiendo una gran variedad en cuanto a marcas, precios y características.

Fig. 1.18.Cámara digital

Los PDAs son ahora un cliente importante en cuanto aplicaciones de software. Es ahora común el uso de agendas electrónicas, juegos en los teléfonos celulares, llamadas de video, etc.

En general, el hardware y software utilizado en lo que va de esta década es la popularización de tecnologías creadas anteriormente pero que en el pasado no se habían difundido, eran costosas o no estandarizadas.

1.2.Aplicaciones

Diseño Asistido por Computadora

En los procesos de diseño se hace un uso importante de las gráficas por computadora, en particular, para sistemas de ingeniería y arquitectura, sin embargo, en la actualidad casi todos los productos se diseñan por computadora. Los métodos CAD se utilizan diariamente en el diseño de automóviles, aeronaves, embarcaciones, naves espaciales, computadoras, telas, construcciones, software y otros muchos productos.

El diseño asistido por computadora, abreviado DAO pero más conocido por las siglas inglesas CAD (Computer Aided Design), se trata básicamente de una base de datos de entidades geométricas (puntos, líneas, arcos, etc) con la que se puede operar a través de una interfaz gráfica. Permite diseñar en dos o tres dimensiones mediante geometría alámbrica, esto es, puntos, líneas, arcos, splines, superficies y sólidos para obtener un modelo numérico de un objeto o conjunto de ellos.

La base de datos asocia a cada entidad una serie de propiedades como color, capa, estilo de línea, nombre, definición geométrica, etc., que permiten manejar la información de forma lógica. Además pueden asociarse a las entidades o conjuntos de estas otro tipo de propiedades como el costo, material, etc., que permiten enlazar el CAD a los sistemas de gestión y producción.

En el caso de algunas aplicaciones de diseño, los objetos se despliegan primero en forma de armazón que muestra la forma general y sus características internas. Los despliegues del armazón permiten que los diseñadores vean con rapidez los efectos de ajustes interactivos para diseñar formas. Las siguientes figuras muestran armazones de figuras diseñadas en un programa CAD.

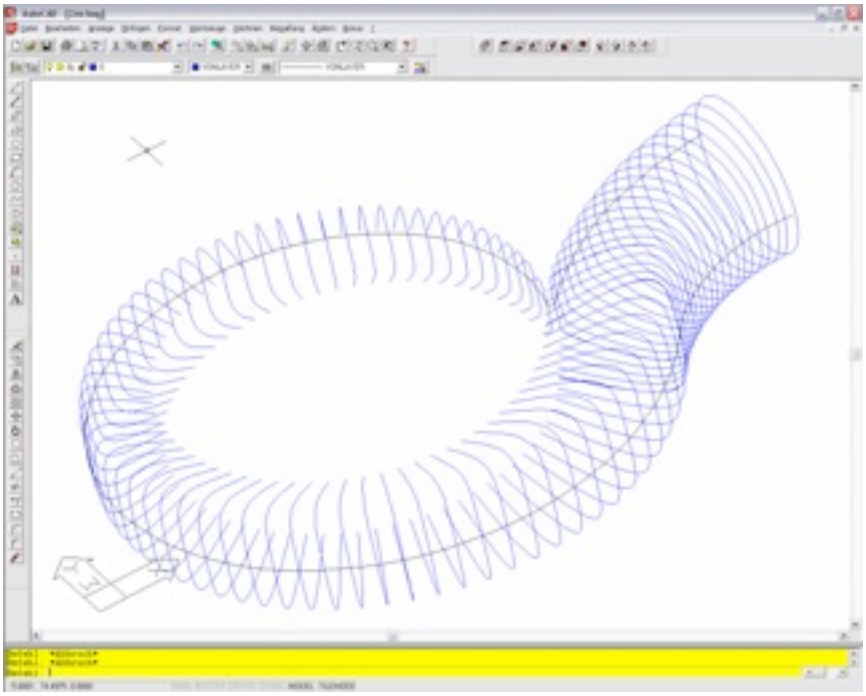


Fig. 1.19.Vista del programa Autocad

Con frecuencia, se usan las animaciones en las aplicaciones CAD. Las animaciones de tiempo real que utilizan armazones son útiles para probar el comportamiento de un vehículo o un sistema.

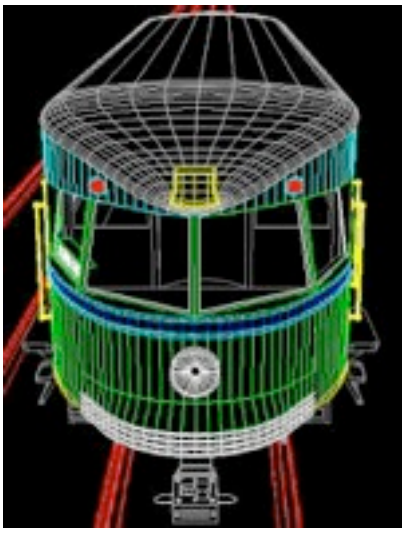


Fig. 1.20.Diseño frontal de un tren en Autocad

Existen paquetes para el diseño de circuitos electrónicos, los cuales permiten diseñar un sistema colocando sucesivamente los componentes en el esquema y conectando estos componentes. Esto permite que el diseñador experimente con esquemas de circuitos alternativos para reducir al mínimo el número de componentes o el espacio requerido para el sistema.

Cuando los diseños de objetos están completos, o casi completos, se aplican modelos de iluminación realista y presentaciones de superficie para mostrar la apariencia del producto final. También se crean vistas realistas para la publicidad de automóviles y otros vehículos mediante efectos especiales de iluminación y escenas de fondo.

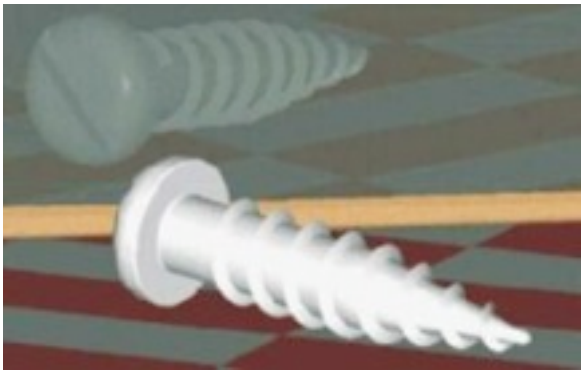


Fig. 1.21.Modelo de un tornillo

Fig. 1.22.Vista modelo de un edificio

Los arquitectos utilizan métodos gráficos interactivos para proyectar plantas arquitectónicas como la de la figura anterior, que muestran la disposición de habitaciones, ventanas, escaleras, anaqueles, barras de cocina y otras características de la construcción. Esto hace posible que los arquitectos y sus clientes estudien la apariencia de una construcción. En algunas ocasiones es posible simular un recorrido virtual por las habitaciones o alrededor de las construcciones para apreciar mejor el efecto general de un diseño.

Arte digital

Los métodos de gráficas por computadora se utilizan en forma forma generalizada tanto en aplicaciones de bellas artes como en aplicaciones de arte comercial. Los artistas o autores utilizan una variedad de métodos computacionales, incluyendo hardware de propósito especial como tabletas digitalizadoras, software desarrollado para este propósito, tales como Adobe Photoshop o Macromedia FreeHand y paquetes CAD.

Fig. 1.23.Tableta Digitalizadora

El artista puede dibujar utilizando la tableta digitalizadora o inclusive el Mouse o ratón para crear un dibujo simple o muy detallado. Herramientas como los “paintbrush” o brochas, incluidos en la mayoría de los programas de dibujo, permiten que los artistas “pinten” imágenes con distintas brochas, las cuales tienen propiedades como color, textura, tamaño y presión de trazo.

Con el propósito de crear pinturas tridimensionales, los artistas hacen uso de paquetes de modelado tridimensional, diagramación de la textura, programas de dibujo y software CAD, algunas veces, un solo paquete contiene todas estas herramientas.

Fig. 1.24.Pintura digital tridimensional

Existe también el arte digital matemático en el que los artistas utilizan una combinación de funciones matemáticas, procedimientos fractales con el fin de crear una variedad de formas tridimensionales y bidimensionales , al igual que pares de imágenes estereoscópicas.

De igual manera, estas técnicas y programas se utilizan para crear diseños con fines comerciales o personales. Entre los usos comerciales se encuentra la creación de logotipos, páginas de revistas, anuncios publicitarios, periódicos, etc. En lo que respecta al uso personal, los usuarios suelen modificar las propiedades de las imágenes (modificar su brillo, contraste, saturación), modificarlas (rotarlas, escalarlas, recortarlas) y hacer fotomontajes.

Un método común que se utiliza en los comerciales es el morphing o transformación, donde un objeto se transforma en otro; podemos ver una lata de aceite convertirse en un automóvil o el rostro de una persona transformase a un rostro diferente.

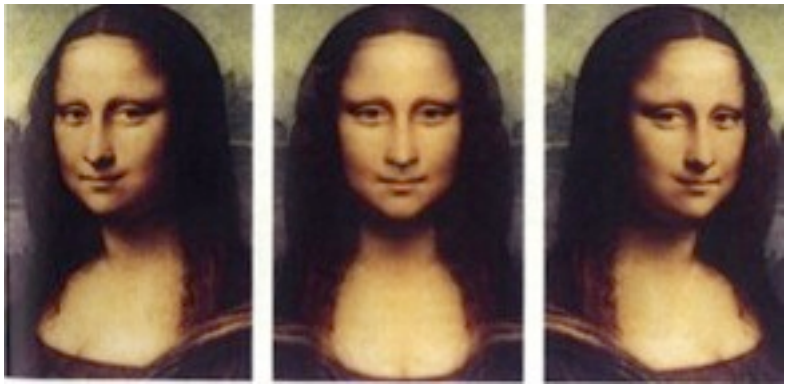


Fig. 1.25.Técnica de morphing aplicada a la MonaLisa

Fig. 1.26.Montaje de un perro con la cabeza de ave

Una ventaja que ofrece el arte digital es que a diferencia del convencional, el primero permite el uso de “layers” o capas, las cuales permiten añadir profundidad a las imágenes. De igual manera, el arte digital no se desgasta con el tiempo ni las condiciones normales a las que se somete el arte gráfico común. Algunos artistas y sociedades puristas no

consideran al arte digital como un arte debido a la ausencia de contacto con las herramientas tradicionales tales como brochas y lienzos.

Entretenimiento

En la actualidad, se utilizan comúnmente métodos de gráficas por computadora para producir películas, videos musicales y programas de televisión. En ocasiones se despliegan solo imágenes gráficas y otras veces se combinan objetos (creados en la computadora) con actores u objetos reales.

Animación por computadora

La animación pertenece al ámbito del cine y la televisión aunque está en relación directa con las artes visuales clásicas, dibujo, pintura y escultura, así como con la fotografía. Para realizar animaciones existen numerosas técnicas que van más allá de los familiares dibujos animados. Una técnica muy utilizada en la actualidad es la animación por computadora, esta permite reducir los costos de producción y edición.

La animación por computadora es para la mayoría de la gente un sinónimo de grandes eventos de la pantalla grande tales como Star Wars, Toy Story y Titanic. Pero no todas las animaciones, o al menos la mayoría, son hechas en Hollywood. No es inusual que las caricaturas sean hechas enteramente en computadora. Los juegos de computadora son un adelanto al estado del arte de las técnicas de gráficas de computadora.

La animación de escritorio es posible ahora a un precio razonable. La animación en el web, hoy, es cosa de rutina. Los simuladores digitales para entrenar pilotos, equipos SWAT y operadores de reactores nucleares son cosa de todos los días.

Fig. 1.27.Escena de la película Toy Story

La animación de escritorio ha tenido auge en los últimos cinco años, aún cuando el GIF animado surgió hace mucho tiempo, no fue sino hasta la popularización de Internet, que fue posible poner al alcance de todos la animación.

Paquetes de software como Macromedia Flash, Moho y algunos otros, han permitido que casi cualquier persona pueda hacer una animación fácilmente. Otras aplicaciones como Maya, permiten hacer animaciones más sofisticadas en 2D y 3D y con una gran calidad.

Fig. 1.28.Escena de la caricatura South Park generada en Flash

Fig. 1.29.Imagen de una animación en Maya

Las secuencias de animación son por lo general creadas en plataformas Macintosh. Los estudios de animación tales como Pixar, han creado enormes laboratorios de computadoras Mac para crear sus películas. La inversión en los proyectos de animación alcanza las cifras de millones de dólares y va en crecimiento.

Videojuegos

Un videojuego (llamado también juego de vídeo) es un programa informático, creado expresamente para divertir, formando parte del sector audiovisual. Los videojuegos están basados en la interacción entre una persona y una computadora (ordenador). Los videojuegos recrean entornos virtuales en los cuales el jugador puede controlar a un personaje o cualquier otro elemento de dicho entorno, y así conseguir uno o varios objetivos por medio de unas reglas determinadas.

Fig. 1.30.Lara Croft, heroína de el video juego Tomb Raider

Los videojuegos son programados como un software, siendo grabados en algún medio de almacenamiento (como un cartucho, una tarjeta, un disquette, un CD, etc.) El hardware que ejecuta los videojuegos puede ser una computadora, o un artefacto especialmente creado para ello, las videoconsolas, divididas a su vez en arcade (de uso público), caseras (que se acoplan en un televisor), portátiles o de bolsillo (de pequeño tamaño y que poseen pantalla propia) y más recientemente los teléfonos móviles (celulares).

Fig. 1.31.Consola de video juegos Xbox

La programación de videojuegos está involucrada con la animación y se complementan. La industria de los videojuegos ha crecido tanto, que algunas universidades ofrecen programas académicos de programación de videojuegos, es decir, que existen ingenieros especialistas en videojuegos. Entre las consolas mas populares se encuentran GameBoy, XBox, Nintendo, y PlayStation.

Películas

Las gráficas por computadora se utilizan en diversas etapas de la creación de películas. Se puede utilizar la animación, edición y efectos especiales, siendo los efectos especiales lo que mas llama la atención entre los consumidores.

Como se había mencionado anteriormente, en las películas o series de televisión es común que se combinen objetos animados y objetos o actores reales, estas técnicas son incluso utilizadas en los noticieros cuando el anunciador del clima es filmado sobre una pantalla azul y digitalmente se agregan los mapas e información del clima.

Fig. 1.32.Escena de la película creada por computadora Los Increíbles

Utilizando programas de edición, un productor puede mezclar, dividir, cortar o modificar escenas de una película con relativa facilidad. Los efectos especiales, son por lo general la última fase de edición de una película, estos pueden hacer volar a un actor, hacer explotar un submarino o crear una tormenta de arena solo por mencionar algunos ejemplos.

Fig. 1.33.Escena de la película Matrix, famosa por sus efectos especiales

La digitalización ha logrado reducir el costo en la filmación de películas; muchos productores, por lo general independientes, pueden llevar a cabo sus proyectos utilizando el formato digital en lugar del convencional.

Educación y capacitación

A menudo, se utilizan como instrumentos de ayuda educativa modelos de sistemas físicos, financieros y económicos, los cuales se generan por computadora. Modelos de sistemas físicos, fisiológicos, tendencias de población, pueden ayudar a los estudiantes a comprender la operación del sistema.

En el caso de algunas aplicaciones de capacitación, se diseñan sistemas especiales. Como ejemplos de tales sistemas especializados, podemos mencionar los simuladores para sesiones de práctica o capacitación de capitanes de barco, pilotos de avión, operadores de quipo pesado y el personal de control de tráfico aéreo.

Fig. 1.34.Simulador de vuelo para capacitación de la NASA

Algunos simuladores no tienen pantallas de video, por ejemplo, un simulador de vuelo solo tiene un panel de control como instrumento de vuelo. Sin embargo, la mayor parte de los simuladores cuenta con pantallas gráficas para la operación visual. Otro tipo de simulador es en el que se monta una pantalla con paneles múltiples en frente del simulador y proyectores a color despliegan la escena del vuelo en la pantalla.

Visualización

Científicos, ingenieros, personal médico, analistas comerciales y otros necesitan con frecuencia analizar grandes cantidades de información o estudiar el comportamiento de ciertos procesos. Las simulaciones numéricas que se efectúan en supercomputadoras a menudo producen archivos de dartos que contienenen miles e incluso millones de valores de datos.

De modo similar, cámaras vía satélite y otas fuentes acumulan grandes archivos de datos más rápido de lo que se puede interpretar. El rastreo de estos grandes conjuntos de número para determinar tendencias y relaciones es un proceso tedioso e ineficaz. Pero si se convierten a una forma visual es frecuente que se perciban de inmediato las tendencias y los patrones.

Fig. 1.35.Colonoscopía asistida por medios y gráficas electrónica

Existen muchas y clases de conjuntos de datos y los esquemas de visualización efectivos dependen de las características de los datos. Una compilación de datos contiene valores escalares, vectores, tensores de orden superior o cualquier combinación de estos tipos de datos. Y los conjuntos de datos pueden ser bidimensionales o tridimensionales.

La codificación de colores es solo una manera de visualizar un conjunto de datos. Las técnicas adicionales incluyen trazos, gráficas y diagramas de contorno, presentaciones de superficie y visualizaciones de interiores de volumen.

Procesamiento de imágenes

A pesar de que los métodos empleados en las gráficas por computadora y en el procesamiento de imágenes se traslapan, las dos áreas realizan, en forma fundamental, operaciones distintas.

En las gráficas por computadora, se utiliza una computadora para crear una imagen. Por otro lado, en el procesamiento de imágenes se aplican técnicas para modificar o interpretar imágenes existentes, como fotografías y rastreos de televisión. Las dos aplicaciones principales del procesamiento de imágenes son:

- El mejoramiento de la calidad de la imagen y
- la percepción de la máquina de información visual, como se utiliza en la robótica.

Para aplicar los métodos de procesamiento de imágenes, primero digitalizamos una fotografía u otra imagen en un archivo de imagen. Entonces, se pueden aplicar métodos digitales para reordenar partes de imágenes, para mejorar separaciones de colores o para aumentar la calidad del sombreado.

Estas técnicas se utilizan en gran medida en aplicaciones de arte comercial que implican el retoque y el reorden de secciones de fotografías y otras obras de arte. Se emplean métodos similares para analizar fotografías de la Tierra por satélite y fotografías de galaxias.

Fig. 1.36.Imagen vía satélite usada para predicciones del clima

Las aplicaciones médicas también hacen uso importante de técnicas de procesamiento de imágenes para mejorar fotografías, en tomografías y simulacros de operaciones. La tomografía es una técnica de fotografía por rayos X la cual permite el despliegue de vistas transversales de sistemas fisiológicos. Tanto la tomografía computarizada (CT; computed tomography) por rayos X, como la tomografía de emisión de posición (PET; position emission tomography) utilizan métodos de proyección para reconstruir secciones transversa-les a partir de datos digitales. Estas técnicas son empleadas para supervisar funciones internas y mostrar secciones transversales durante una cirugía.

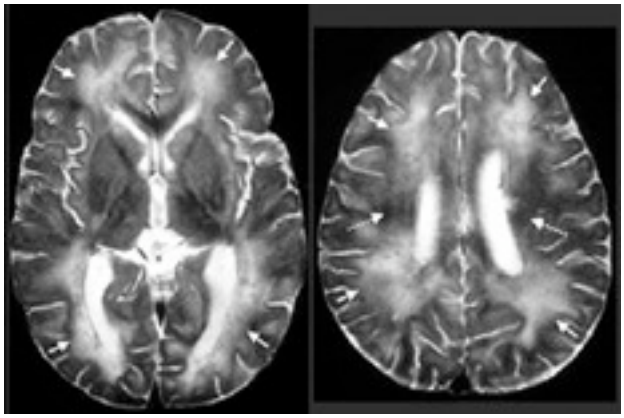


Fig. 1.37.Tomografía computarizada

Otras técnicas de proyección de imágenes médicas incluyen rastreadores ultrasónicos

y nucleares. Con el ultrasonido, se utilizan ondas sonoras de alta frecuencia, en vez de rayos X, para generar datos digitales. Los rastreadores para medicina nuclear recopilan datos digitales de la radiación que emiten radionúclidos ingeridos y trazan imágenes con codificación de colores.

Por lo general, el procesamiento de imágenes y las gráficas por computadora se combinan en muchas aplicaciones. Por ejemplo, en medicina se utilizan estas técnicas para modelar y estudiar funciones físicas, para diseñar miembros artificiales, así como planear y practicar cirugías.

Esta última aplicación se conoce, por lo general, cirugía asistida por computadora. Se obtienen secciones transversales bidimensionales del cuerpo a través de la utilización de técnicas de proyección de imágenes. Luego se ven y manipulan los cortes utilizando métodos gráficos para simular procedimientos quirúrgicos reales y experimentar con diversas incisiones quirúrgicas.

Fig. 1.38.Cirugía asistida por computadora

Interfaces gráficas de usuario

Es común que los paquetes de software ofrezcan una interfaz gráfica. Un componente importante de una interfaz gráfica es un administrador de ventanas que hace posible que un usuario despliegue áreas con ventanas múltiples. Cada ventana puede contener un proceso distinto que a su vez puede contener despliegues gráficos y no gráficos.

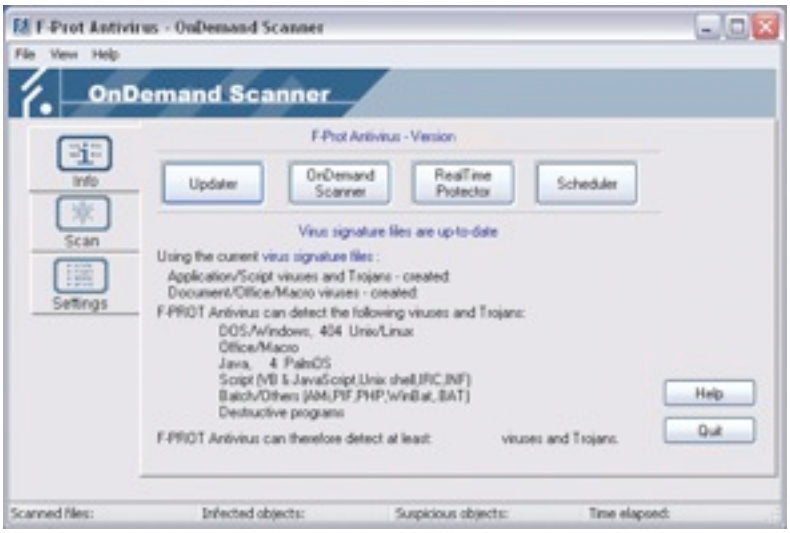


Fig. 1.39.Interfaz tipo ventana de un programa

Para activar una ventana en particular, sólo hacemos clic en esa ventana utilizando un dispositivo de pulsar interactivo. Las interfaces también despliegan menús e iconos para permitir una selección rápida de las opciones de procesamiento o de valores de parámetros.

Un icono es un símbolo gráfico diseñado para semejarse a la opción de procesamiento que representa. La ventaja de los iconos es que ocupan menos espacio en la pantalla que las descripciones textuales correspondientes y que se pueden entender con mayor rapidez si están bien diseñados. Los menús contienen listas de descripciones textuales e iconos.

Existen varios estilos de ventanas: los de Microsoft Windows, el estilo aqua, característico del Mac y los personalizables estilos de Linux.

Fig. 1.40.El estilo aqua de Mac OS

Otras interfaces que se están popularizando cada vez mas, son las interfaces web, estas tienen la singular característica de modificarse en minutos, lo cual es una gran ventaja contra las aplicaciones de escritorio. Muchos desarrolladores de software prefieren desarrollar aplicaciones web no solo por su facilidad en la programación sino porque son mas intuitivas que las interfaces de escritorio.

Fig. 1.41.Interface web

1.3.Formatos gráficos de almacenamiento

El almacenamiento de los datos que componen una imagen digital en un archivo binario puede realizarse utilizando diferentes formatos gráficos, cada uno de los cuales ofrece diferentes posibilidades con respecto a la resolución de la imagen, la gama de colores, la compatibilidad, la rapidez de carga, etc.

La finalidad última de un formato gráfico es almacenar una imagen buscando un equilibrio adecuado entre calidad, peso final del fichero y compatibilidad entre plataformas. Para ello, cada formato se basa en una o más técnicas diferentes, que pueden incluir codificación especial, métodos de compresión, etc.

Generalmente, todo fichero gráfico comienza con una cabecera (header) de estructura variable, que indica al programa que lo solicite las características de la imagen

que almacena (tipo, tamaño, resolución, modo de color, profundidad de color, número de colores de la paleta si la hay, etc).

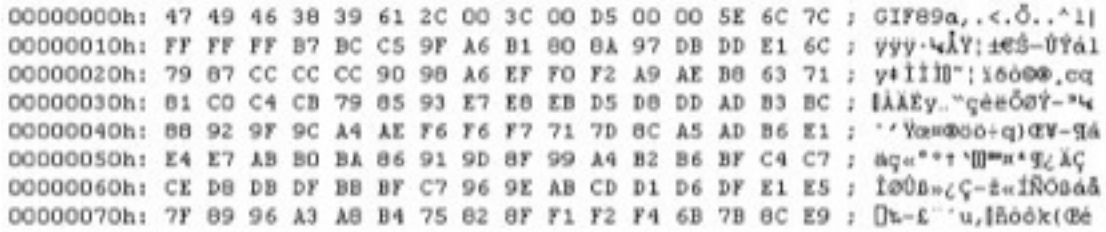


Fig. 1.42.Código de un archivo gráfico

A continuación se encuentran los datos propios de la imagen, generalmente comprimidos con un algoritmo específico de ese formato, que contienen información sobre el color de cada píxel de la imagen (mapas de bits) o una tabla con las características propias de cada objeto (gráficos vectoriales). En caso de usarse una paleta de colores, la información sobre dicha paleta también deberá estar contenida en el fichero.

La imagen puede estar formada por un número diferente de píxeles, dependiendo de su tamaño y resolución, y tener más o menos colores. En función del número de píxeles y del número de colores la imagen tendrá más o menos calidad, pero cuanto más calidad tenga, más ocupará el fichero necesario para almacenarla. En el caso de los gráficos vectoriales no se definen píxeles individuales, dependiendo la calidad y el peso final del formato concreto en que se almacenen.

Los ficheros gráficos de mapas de bits contienen pues una cabecera, los datos de los píxeles (generalmente comprimidos) y la paleta de colores (salvo si se usan 24 bits por píxel, caso en el que no es necesaria ninguna paleta). Los ficheros vectoriales, una cabecera y una tabla con las características de cada vector componente del gráfico.

Cada formato es independiente. Las posibilidades que ofrece cada formato con respecto a la gama de colores, a la compatibilidad, a la rapidez de carga, etc., merece ser explicada para determinar cuál de ellos es el más adecuado para la tarea que estamos realizando.

Existen dos tipos de formatos: los vectoriales y los de mapa de bits también conocidos como rasterizados.

Una imagen rasterizada es una estructura o fichero de datos que representan generalmente una rejilla rectangular de pixeles o puntos de color en un monitor de ordenador, papel u otro dispositivo de representación.

El color de cada pixel está definido individualmente; Por ejemplo, una imagen en un espacio de color RGB, almacenaría el valor de color de cada pixel en tres bytes: un para el verde, para el azul, y para el rojo.

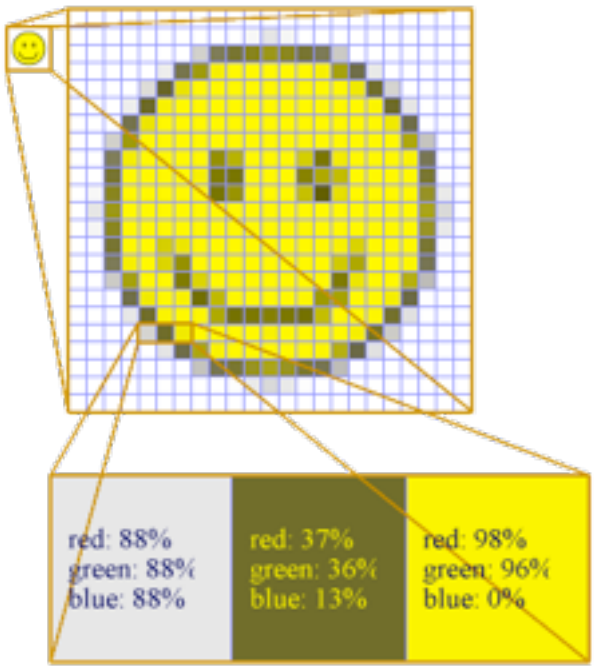


Fig. 1.43.Detalle de una imagen de mapa de bits

Los gráficos rasterizados se distinguen de los gráficos vectoriales en que estos últimos representan una imagen a través del uso de objetos geométricos como curvas y polígonos, no del simple almacenamiento del color de cada pixel.

Los formatos de mapa de bits más utilizados son los siguientes:

- ART
- BMP, Windows Bitmap
- CIN, Cineon
- CPT, Corel Photo Paint
- DPX, Digital Picture eXchange
- DRW, Draw

- EXR, Extended Dynamic Range Image File Format
- FPX, Flashpix
- GIF, Graphics Interchange Format
- JPG / JPEG, Joint Photographic Experts Group
- MNG, Multiple-image Network Graphics
- PBM, Portable Bitmap Format
- PCX, Picture eXchange
- PGM, Portable Graymap Format
- PIC, Pictue
- PNG, Portable Network Graphics
- PPM, Portable Pixmap Format
- PSD, PhotoShop Document
- PSP, PaintShop Pro Document
- TGA, Truevision TGA
- TIFF, Tagged Image File Format
- WBMP, Wireless Application Protocol Bitmap Format
- XBM, X BitMap
- XCF, eXperimental Computing Facility
- XPM, X-Pixmap

Los gráficos vectoriales son los que se representan en los gráficos por ordenador por medio de "trazos", es decir, por primitivas geométricas como puntos, líneas, curvas o polígonos. En contraste, se encuentran los gráficos formados por una retícula de pixeles como los bitmap.

En los gráficos vectoriales la imagen se genera como descripción de trazos. Por ejemplo, para crear una línea recta se indica: su posición inicial (x1,y1), su posición final (x2,y2), su grosor, color, etc. En cambio, en una imagen bitmap, esa misma línea estaría formada por un número determinado de puntos (pixeles) de color contiguos.

Al contrario que un bitmap, una imagen vectorial puede ser escalada, rotada o deformada, sin que ello perjudique en su calidad. Normalmente un conjunto de trazos se puede agrupar, formando objetos, y crear formas más complejas que permiten el uso de

curvas bezier, degradados de color, etc. En algunos formatos, como el SWF, las imágenes vectoriales pueden animarse muy facilmente sin que ello suponga un aumento excesivo en el tamaño del fichero, al contrario de los bitmaps

Fig. 1.44.Imagen de un tren vectorizada

Fig. 1.45.Imagen de un tren en mapa de bits.

Los formatos vectoriales más utilizados son los siguientes:

- AI, Adobe Illustrator Document
- CDR, CorelDRAW
- CGM, Computer Graphics Metafile
- DXF, ASCII Drawing Interchange
- DWG, AutoCAD Drawing Database
- EMF, Enhanced MetaFileEPS, Encapsulated PostScript
- FHX, Macromedia Freehand Document

- FLA, Fichero fuente de Macromedia Flash
- PDF, Portable Document Format
- PS, PostScript
- SVG / SVGZ, Scalable Vector Graphics
- SWF, ShockWave Flash
- WMF, Windows MetaFile

Existen también los metaficheros o metaformatos. Un metaformato es un fichero o archivo de intercambio que sirve para transportar datos entre varios sistemas o programas. Es común que las compañías de software creen sus propios archivos para facilitar el manejo de los datos de las imágenes creadas en sus aplicaciones. Un metaformato puede ser vectorial o de mapa de bits.

AI, Adobe Ilustrator

El metaformato AI es el utilizado por el programa Adobe Ilustrator para guardar sus ficheros gráficos nativos.

Fig. 1.46.Logotipo en formato AI

Los ficheros AI admiten cabecera de previsualización (thumbnail) y pueden trabajar con vectores y mapas de bits. Permiten texturas, degradados, fotos integradas o vinculadas a ficheros externos, textos trazados o con fuentes incluidas y manejo de capas y máscaras.

Suele producir ficheros de peso medio, dependiendo del contenido, pero se puede rebajar ya que admite algoritmos de compresión sin pérdidas.

Es un formato muy popular, válido para PC y MAC, apto para intercambiar gráficos entre diferentes aplicaciones, pero teniendo siempre en cuenta la versión de Ilustrator que creó el archivo original, ya que deben de ser versiones compatibles.

BMP, BitMaP

Los archivos con extesión .BMP, en los sistemas operativos Windows, representan la sigla BitMaP, o sea mapa de bits. Los archivos de mapas de bits se componen de direcciones asociadas a códigos de color, uno para cada cuadro en una matríz de pixeles tal como se esquematizaría un dibujo de "colorea los cuadros" para niños pequeños.

Normalmente, se caracterizan por ser muy poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales, al ser reescalados a un tamaño mayor, pierden calidad. Otra desventaja de los archivos BMP es que no son utilizables en páginas web debido a su gran tamaño en relación a su resolución.

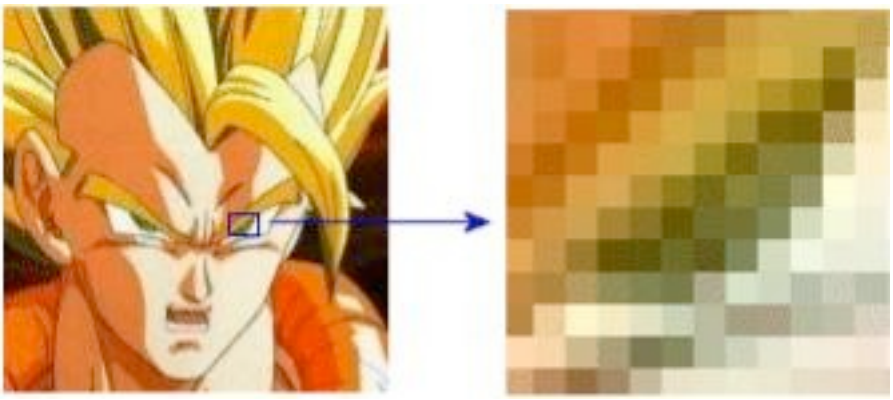


Fig. 1.47.Detalle de imagen en formato BMP

Dependiendo de la profundidad de color que tenga la imagen cada pixel puede ocupar 1 o varios bytes. Generalmente se suelen transformar en otros formatos, como JPEG (fotografías), GIF o PNG (dibujos y esquemas), los cuales utilizan otros algoritmos para conseguir una mayor comprensión (menor tamaño del archivo).

Los archivos comienzan (cabecera o header) con las letras 'BM' (0x42 0x4D), que lo identifica con el programa de visualización o edición. En la cabecera también se indica el tamaño de la imagen y con cuántos bytes se representa el color de cada pixel. Después se

guarda la información píxel por píxel. Tiene una curiosa forma de almacenar esta información: comienza desde la última línea inferior. Es por eso que los programas encargados de mostrar los BMP en pantalla trazan la imagen de abajo hacia arriba.

En imágenes de 4 y 8 bits, también puede escoger compresión Run-LengthEncoding (RLE); este esquema de compresión no produce pérdidas, es decir, no elimina ningún detalle de la imagen.

CDR, Corel Draw

CDR es el formato nativo del programa de gráficos vectoriales Corel Draw, siendo válido para PC y MAC.

Fig. 1.48.Diseño en formato CDR

Es un formato vectorial, pero admite la inclusión de elementos de mapa de bits (integrados o vinculados a ficheros externos), pudiendo llevar además cabecera de previsualización (thumbnail). Junto a AI es uno de los formatos con más posibilidades con respecto al color, a la calidad de los diseños y al manejo de fuentes, pudiendo contener los textos trazados o con fuentes incluidas.

Una de las principales desventajas de este formato es su falta de compatibilidad con el resto de aplicaciones gráficas, al ser éstas incapaces de almacenar imágenes bajo este formato.

CIN, Cineon

El formato Cineon fue diseñado específicamente para representar imágenes escaneadas de películas. Tiene algunas diferencias interesantes con otros formatos tales como el tiff y jpeg:

- Los datos son almacenados en un formato grande que corresponden directamente a la densidad del negativo. densidad = log(exposición)
- Cada canal (RGB) se almacena en 10 bits empacado en palabras de 32 bits con 2 bits libres

El formato tiene una noción del “humbral negro” y el “humbral blanco”, convencionalmente 95 y 685 en la escala de 0 - 1024. Los pixeles cerca de 685 son definidos como “más brillantes que el blanco” tales como el soles, luces cromadas, etc. Este es el concepto de HDRI, de hecho este concepto se uso en la industria de las películas muchos años antes de que se denominara HDRI en los 1990s debido al trabajo de Debevec.

El formato Cineon fue definido en un documento de Kodak por Glenn Kennel. El formato se reemplazó tiempo después por el formato DPX que es muy similar y está basado en el Cineon.

CPT, Corel PhotoPaint

Formato propietario usado por defecto en los documentos de Corel PhotoPaint. Dispone de importantes características extra, como la composición por capas. Compatible con muy pocos programas aparte de los de la misma casa. Su tamaño suele ser menor que el de los documentos creados por Adobe Photoshop.

DPX, Digital Picture eXchange

DPX es un formato común para películas digitales y es un estándar ANSI/SMPTE (268M - 2003). El formato representa la densidad de cada canal de color de un negativo escaneado en un formato de 10 bits de longitud donde la gama del negativo original es preservado tal como se tomo del escaner. DPX ofrece gran flexibilidad en el almacenamiento del color y otra información de intercambio para detalles de producción. Posee múltiples formas de empaquetado y alineación.

El formato DPX es un derivado de de el formato de salida del escáner Kodak Cineon y fue publicado por SMPTE como un estándar.

DRW, Draw

Formato gráfico vectorial usado por diferentes programas que funcionan bajo DOS y Windows, como Micrografx Designer o Windows Draw. Los gráficos .drw pueden ser incluidos en presentaciones creadas con PowerPoint, en diagramas de Microsoft Visio 2000 o en documentos de Microsoft Word.

DXF, Drawing Interchange Format

El formato DXF es un formato vectorial que la empresa Autodesk lanzó para permitir el intercambio de archivos de dibujo entre los diferentes programas de CAD. Soporta hasta 256 colores (8 bits).

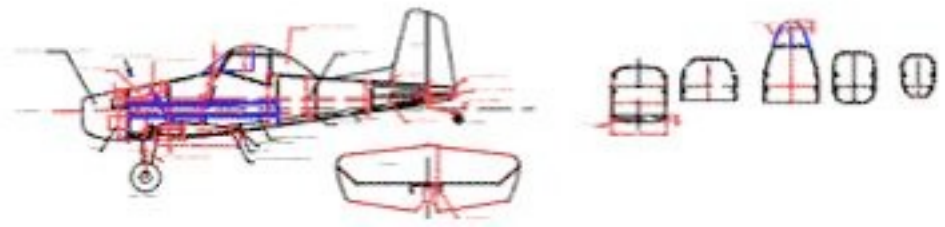


Fig. 1.49.Plano almacenado en formato DXF

Existen dos versiones de DXF (ASCII y binario), que no utilizan ningún algoritmo de compresión. Los ficheros de la versión ASCII contienen números y órdenes a realizar escritos en codificación ASCII, por lo que pueden ser abiertos y leídos con cualquier editor de texto, como el Notepad de Windows. Esta información indica la ubicación de puntos flotantes matemáticos o floating points, utilizados para exhibir la imagen en pantalla. Este sistema, más lento que el de otros formatos, requiere hardware avanzado para poder funcionar correctamente.

Los ficheros en formato DXF son reconocidos por la mayoría de sistemas CAD y por algunos programas de diseño gráfico vectorial, como Corel Draw y Adobe Ilustrator, que pueden manejarlo sin mayores dificultades.

EMF, Enhanced MetaFile

EMF es un metaformato gráfico vectorial de 32 bits, reconocido por casi todas las aplicaciones de diseño gráfico y compatible con los sistemas operativos Windows, pudiendo ser usado en las aplicaciones del paquete Office.

Junto a las características propias de los formatos vectoriales presenta la ventaja adicional de que sus ficheros pueden ser creados rápidamente, ya que lo que se encola en la impresora son comandos de dibujo, con lo que se puede evitar la sobrecarga en el caso de impresión remota de ficheros gráficos.

Además, este formato es más eficiente porque genera un archivo relativamente pequeño y genérico, que es compatible con todas las impresoras.

Como desventaja, los archivos de formato EMF no contienen la misma cantidad de detalles que los de otros tipos de formatos gráficos vectoriales, como los archivos DWF.

Por lo que respecta a la web, es soportado por Internet Explorer, aunque hay algunos gráficos que no son interpretados correctamente.

EPS, Encapsulated PostScript

PostScript encapsulado, o EPS, es un formato de archivo gráfico. Un archivo EPS es un archivo PostScript que satisface algunas restricciones adicionales. Estas restricciones intentan hacer más fácil a programas de software el incluir un archivo EPS dentro de otro documento PostScript.

Como mínimo, un archivo EPS contiene un comentario BoundingBox (bordes de la caja), describiendo el rectángulo que contiene a la imagen. Las aplicaciones pueden utilizar esta información para distribuir elementos en una página, incluso si son incapaces de interpretar el PostScript contenido en el archivo.

¿Es ps y eps lo mismo?

PostScript/ (PS) es un lenguaje de programación para describir páginas. Como lenguaje estructurado permite la programación (tiene estructuras de control y bucles), y recuerda el lenguaje de programación FORTH. Originalmente fue desarrollado por Adobe.

Existen varios intérpretes de PostScript que permiten la visualización de este formato. El más extendido es Ghostscript/ (GS), de Aladdin.

El formato PS se basa en describir cada página desde un origen de coordenadas que se sitúa en la esquina inferior izquierda de la página. PS permite, sin embargo, redefinir el origen, de forma que se puede recomenzar la descripción de un bloque de una página desde un origen arbitrario.

PS encapsulado (EPS) es el formato estándar para importar y exportar archivos PS en cualquier tipo de entornos. Usualmente es un archivo que contiene una sola página que describe una figura. El archivo EPS está especialmente pensado para incluirlo en otros archivos PS, y es como cualquier otro archivo PS con algunas restricciones.

EXR, Extended Dynamic Range Image File Format

OpenEXR es el formato de código libre para imágenes de alto rango dinámico (High dynamic-range o HDR) desarrollado por la industria Light & Magic para la generación de imágenes en las producciones de cine. La principal ventaja del formato es que soporta píxels en coma flotante de más de 32 bits y múltiples algoritmos de compresión sin pérdidas, con un ratio superior al 2:1 en imágenes con grano.

FLA, Macromedia Flash Document

Fla es el formato utilizado por Macromedia Flash para la creación y edición de sus populares animaciones. Un archivo fla guarda todos los datos de una película: los gráficos y textos en forma vectorial, las imágenes incluidas en la película, los sonidos o algún otro elemento tal como un vídeo, así como la información de la animación en sí y los actionscripts.

El formato fla al ser compilado da lugar a un archivo en formato swf, el cual puede ejecutarse en cualquier ordenador que cuente con el plug-in necesario. Cabe mencionar que un archivo en formato fla no puede reproducir la animación que contiene a menos que se esté trabajando con el archivo en Macromedia Flash.

FHx, Macromedia FreeHand File

Formato nativo del programa de gráficos vectoriales FreeHand x, x representa la versión de la aplicación mencionada. Puede llevar cabecera de previsualización (thumbnail) y se puede comprimir, dependiendo el tamaño final del contenido. Puede llevar las fotos integradas o vinculadas a ficheros externos y textos trazados o con fuentes incluidas. Es posible importarlo a diferentes programas gráficos, como Macromedia Flash o Adobe Ilustrator, pero no es soportado por ningún navegador web.

GIF, Graphics Interchange Format

GIF es un formato gráfico utilizado ampliamente en la World Wide Web, tanto para imágenes como para animaciones.

El formato fue creado por CompuServe en 1987 para dotar de un formato de imagen a color para sus áreas de descarga de ficheros, sustituyendo su temprano formato RLE en blanco y negro. GIF llegó a ser muy popular porque podía usar el algoritmo de compresión LZW (Lempel Ziv Welch) para realizar la compresión de la imagen, que era más eficiente que el algoritmo Run-Lenght Encoding (RLE) que usaban formatos como PCX y MacPaint. Por lo tanto, imágenes de gran tamaño podían ser descargadas en un razonable periodo de tiempo, incluso con modems muy lentos.

GIF es un formato sin pérdida de calidad, siempre que partamos de imágenes de 256 colores o menos. Una imagen de alta calidad, como una imagen de color verdadero (profundidad de color de 24 bits o superior) debería reducir literalmente el número de colores mostrados para adaptarla a este formato, y por lo tanto existiría una pérdida de calidad.

Sus principales características son:

- Uso de color indexado, a través de una paleta de colores que puede ser de distintos tamaños, dependiendo del valor del Size of Local Color Table, que tiene un tamaño de 3 bits. El número de colores se puede calcular mediante la fórmula: 2(Size of Local Color Table+1) Esto permite a GIF usar una paleta de 2,4,8,16,32,64,128 ó 256 colores.
- Aunque con mediante el uso de varias capas transparentes (con un máximo de 256 colores en cada una) separadas por 0 milisegundos (simultáneas) entre ellas, si pueden mostrarse imágenes con más de 24 colores diferentes, permitiendo mostrar un color real. Pese a esto, esta última técnica es poco eficiente, y rara vez se usa, sobretodo para demostrar esta posibilidad, a veces, estas imágenes no aparcenen simultáneamente sino que va apareciendo cada una de las capas sucesivamente. En este caso, cada capa sería un cuadrado de 16 por 16, en el que como mucho se podrían mostrar 256 colores, la imágen se divide en dichos recuadros, y se van superponiendo uno sobre otro.
- Permite transparencia de 1 bit, de tal forma que cada pixel de la imagen puede ser o no transparente. Esto lo diferencia de formatos como el PNG, que también dispone transparencia variable.
- Sus últimas versiones permiten hacer animaciones simples, aunque la compresión es muy deficiente.
- Permite utilizar entrelazado en imágenes, de tal forma que las imágenes se visualicen al completo nada más empezar su descarga, pero con una baja definición que va progresando hasta cargarse por completo en los navegadores.
- Profundidad de color: 8 bits máximo (256 colores simultáneos)

Ventajas e inconvenientes

Las principales ventajas del formato GIF son las siguientes:

- Es uno de los dos formatos históricos de Internet, junto con el JPEG, por lo que es compatible con la práctica totalidad de los navegadores.
- Permite la creación de animaciones, por lo que ha sido utilizado ampliamente en banners publicitarios.

- Se ha empleado mucho para logotipos y gráficas, por su transparencia binaria y el menor tamaño de archivo que se alcanza en imágenes con pocos colores frente al JPEG.

Sus principales inconvenientes son:

- Las paletas de un máximo de 256 colores lo hacen inapropiado para fotografías y otros tipos de imágenes con gran variedad cromática para los que se prefiere el JPEG (con pérdidas de cálidad) o PNG (sin perdidas).
- No soporta canal alfa, por lo que no permite transparencias suaves de 8 bits.
- El propietario de la patente del algoritmo LZW que se utiliza en el formato GIF reclama el pago de royalties por su uso. Así, cualquier programa capaz de abrir o guardar archivos GIF comprimidos con LZW debe cumplir con sus exigencias. Esto hace que su uso sea desaconsejado por el W3C, y perjudicial para el software libre y proyectos libres como Wikipedia. Es necesario recalcar que el formato GIF puede utilizar otros métodos de compresión no cubiertos por patentes, como el método Run-length encoding.
- Actualmente se tiende a sustituirlo por los formatos libres PNG (imágenes estáticas) y MNG (imágenes animadas). PNG soporta múltiples formatos: imágenes RGB de color verdadero con canal alfa e incluso imágenes de 8 bits con transparencia como el GIF. Para estas últimas, consigue un tamaño de fichero sensiblemente menor al GIF. Además ha sido elegido como estándar gráfico para la Web por el W3C.

JPEG, Joint Photographic Experts Group

JPEG es un algoritmo diseñado para comprimir imágenes con 24 bits de profundidad o en escala de grises. JPEG es también el formato de fichero que utiliza este algoritmo para comprimir imágenes. JPEG sólo trata imágenes fijas, pero existe un estándar relacionado llamado MPEG para videos. El formato de archivos JPEG se abrevia frecuentemente JPG debido a que algunos sistemas operativos sólo aceptan tres letras de extensión.

JPEG es un algoritmo de compresión con pérdida. Esto significa que al descomprimir la imagen no obtenemos exactamente la misma imagen que teníamos antes de la compresión.

Una de las características que hacen muy flexible el JPEG es el poder ajustar el grado de compresión. Si especificamos una compresión muy alta se perderá una cantidad significativa de calidad, pero obtendremos ficheros pequeños. Con una cantidad de compresión baja obtenemos una calidad muy parecida a la del original, y un fichero extremadamente grande .

Esta pérdida de calidad se acumula. Esto significa que si comprime una imagen y la descomprime obtendrá una calidad de imagen, pero si vuelve a comprimirla y descomprimirla otra vez obtendrá una perdida mayor. Cada vez que comprima y descomprima la imagen esta perderá algo de calidad.

El formato de ficheros JPEG o JPG fue creado por un grupo independiente, llamado JFIF (JPEG File Interchange Format), quienes se encargan solo de la utilización del algoritmo JPEG para almacenar imágenes. Existen otros formatos de fichero que también utilizan el algoritmo JPEG, el más conocido de ellos es JNG.

JPEG/JFIF es el formato más utilizado para almacenar y transmitir archivos de fotos en la Web. Pero la compresión con pérdida del formato no conviene a diagramas que incluyen textos y líneas.

Codificación

Muchas de las opciones del estándar JPEG se usan poco. Esto es una descripción breve de uno de los muchos métodos comúnmente usados para comprimir imágenes cuando se aplican a una imagen de entrada con 24 bits por pixel (ocho por cada rojo, verde, y azul). Esta opción particular es un método de compresión con pérdida.

Transformación del espacio de color

Lo primero que se realiza es convertir la imagen desde su modelo de color RGB a otro llamado YUV. Este espacio de color es similar al que usan los sistemas de color para televisión PAL y NTSC, pero es mucho más parecido al sistema de televisión MAC.

Este espacio de color (YUV) tiene tres componentes:

- La componente Y representa el brillo de cada pixel, es decir blanco y negro.
- Las componentes U y V representan juntas el color (saturación y hue)

El resultado es una imagen en la que el brillo está separado de la crominancia.

Submuestreo

Una opción que se puede aplicar al guardar la imagen, es reducir la información del color respecto a la de brillo. Hay varios métodos: si este paso no se aplica, la imagen sigue en su espacio de color YUV, (este submuestreo se entiende como 4:4:4), con lo que la imagen no sufre pérdidas.

Puede reducirse la información cromática a la mitad, 4:2:2 (reducir en un factor de 2 en dirección horizontal), con lo que el color tiene la mitad de resolución (en horizontal), y el brillo sigue intacto. Otro método, muy usado, es reducir el color a la cuarta parte, 4:2:0, en el que el color se reduce en un factor de 2 en ambas direcciones, horizontal y vertical. Si la imagen de partida estaba en escala de grises (blanco y negro), puede eliminarse por completo la información de color, quedando como 4:0:0.

Algunos programas que permiten el guardado de imágenes en JPEG (como el que usa GIMP) se refieren a estos métodos con 1×1,1×1,1×1 para YUV 4:4:4 (no perder color), 2×1,1×2,1×1 para YUV 4:2:2 y 2×2,1×1,1×1 para el último método, YUV 4:2:0.

Transformación discreta de coseno o DCT

Entonces, cada componente de la imagen se divide en pequeños bloques de 8×8 píxeles, que se procesan de forma casi independiente, de esto resulta la formación de los bloques, que se hace notable en imágenes guardadas con altas compresiones. Si la imagen sufrió un submuestreo del color, los colores quedarían en la imagen final en bloques de 8×16 y 16×16 pixeles, según fuese 4:2:2 o 4:2:0.

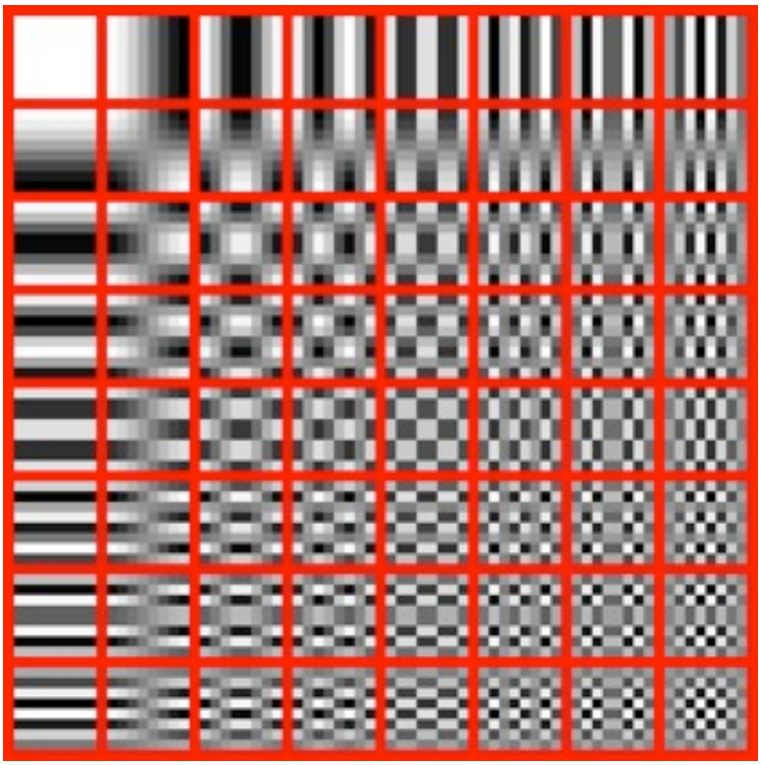


Fig. 1.50.Uso del algoritmo JPEG en una imagen

El algoritmo JPEG, transforma la imagen en cuadrados de 8×8 y luego almacena estos como una combinación lineal de estos 64 recuadros, permitiendo eliminar detalles de forma selectiva

Después cada pequeño bloque se convierte al dominio de la frecuencia a través de la transformación discreta de coseno bidimensional, abreviadamente llamada DCT.

Un ejemplo de uno de esos pequeños bloques de 8×8 inicial es este:

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

El siguiente proceso es restarles 128 para que queden números entorno al 0, entre -128 y 127.

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -63 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

Se procede a la transformación por DCT de la matriz, y el redondeo de cada elemento al número entero más cercano.

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

La transformación DCT utiliza las siguientes fórmulas:

$$f_j = \frac{1}{2}(x_0 + (-1)^j x_{n-1}) + \sum_{k=1}^{n-2} x_k \cos\left[\frac{\pi}{n-1} jk\right]$$

$$f_j = \sum_{k=0}^{n-1} x_k \cos\left[\frac{\pi}{n} j\left(k + \frac{1}{2}\right)\right]$$

Cuantización

El ojo humano es muy bueno detectando pequeños cambios de brillo en áreas relativamente grandes, pero no cuando el brillo cambia rápidamente en pequeñas áreas (variación de alta frecuencia), esto permite eliminar las altas frecuencias, sin perder excesiva calidad visual. Esto se realiza dividiendo cada componente en el dominio de la frecuencia por una constante para ese componente, y redondeándolo a su número entero más cercano. Este es el proceso en el que se pierde la mayor parte de la información (y calidad) cuando una imagen es procesada por este algoritmo. El resultado de esto es que los componentes de las altas frecuencias, tienden a igualarse a cero, mientras que muchos de los demás, se convierten en números positivos y negativos pequeños.

Una matriz de cuantización típica es esta:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Dividiendo cada coecifiente de la matriz de la imagen transformada entre cada coecifiente de la matriz de cuantización, se obtiene esta matriz, ya cuantizada:

$$\begin{bmatrix} -2 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Por ejemplo, cuantizando el primer elemento, el coecifiente DC, sería así:

$$\text{round}\left(\frac{-415}{16}\right) = \text{round}\left(-25.9375\right) = -26$$

Codificación entrópica

La codificación entrópica es una forma especial de la compresión sin pérdida de datos. Para ello se cogen los elementos de la matriz siguiendo una forma de zig-zag, poniendo grupos con frecuencias similares juntos, e insertando ceros de codificación, y usando la Codificación Huffman para lo que queda.

También se puede usar la codificación aritmética, superior a la de Huffman, pero que rara vez se usa, ya que está cubierta por patentes, esta compresión produce archivos un 5% menores, pero a costa de un mayor tiempo de codificación y decodificación, esta pequeña ganancia, puede emplearse también en aplicar un menor grado de compresión a la imagen, y obtener más calidad para un tamaño parecido.

En la matriz anterior, la secuencia en zig-zag, es esta:

−26, −3, 0, −3, −3, −6, 2, −4, 1 −4, 1, 1, 5, 1, 2, −1, 1, −1, 2, 0, 0, 0, 0, 0, −1, −1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

JPEG tiene un código Huffman para cortar la cadena anterior en el punto en el que el resto de coecifientes sean ceros, y así, ahorrar espacio:

–26, –3, 0, –3, –3, –6, 2, –4, 1 –4, 1, 1, 5, 1, 2, –1, 1, –1, 2, 0, 0, 0, 0, 0, –1, –1, EOB

Ruido producido por la compresión

El resultado tras la compresión, puede variar, en función de la agresividad de los divisores de la matriz de cuantización, a mayor valor de esos divisores, más coecifientes se convierten en ceros, y más se comprime la imagen. Pero mayores compresiones producen mayor ruido en la imagen, empeorando su calidad. Una imagen con una fuerte compresión (1%-15%) puede tener un tamaño de archivo mucho menor, pero tendrá tantas imperfecciones que no será interesante, una compresión muy baja (98%-100%) producirá una imagen de muy alta calidad, pero, tendrá un tamaño tan grande que quizás interese más un formato sin pérdida como PNG.

La mayoría de personas que naveguen por Internet estarán familiarizadas con estas imperfecciones, son el resultado de lograr una buena compresión; para evitarlos, se tendrá que reducir el nivel de compresión o aplicar compresión sin pérdida, produciendo mayores ficheros después.

Decodificación

El proceso es similar al seguido hasta ahora, sólo que de forma inversa. En este caso, al haber perdido información, los valores no coincidirán.

Se toma la información de la matriz, se descodifica, y se pone cada valor en su casilla correspondiente. Después se multiplica cada uno de estos valores por el valor correspondiente de la matriz de cuantización usada, como muchos valores son ceros, sólo se recuperan (y de forma aproximada) los valores de la esquina superior izquierda.

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Después se deshace la transformación DCT:

$$\begin{bmatrix} -68 & -65 & -73 & -70 & -58 & -67 & -70 & -48 \\ -70 & -72 & -72 & -45 & -20 & -40 & -65 & -57 \\ -68 & -76 & -66 & -15 & 22 & -12 & -58 & -61 \\ -62 & -72 & -60 & -6 & 28 & -12 & -59 & -56 \\ -59 & -66 & -63 & -28 & -8 & -42 & -69 & -52 \\ -60 & -60 & -67 & -60 & -50 & -68 & -75 & -50 \\ -54 & -46 & -61 & -74 & -65 & -64 & -63 & -45 \\ -45 & -32 & -51 & -72 & -58 & -45 & -45 & -39 \end{bmatrix}$$

Y finalmente se suma 128 a cada entrada:

$$\begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 80 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$

Para comparar las diferencias entre el bloque original y el comprimido, se halla la diferencia entre ambas matrices, la media de sus valores absolutos, da una ligera idea de la calidad perdida:

$$\begin{bmatrix} -8 & -8 & 5 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix}$$

Se puede observar que las mayores diferencias están cerca de la mancha, y por la parte inferior, entre la esquina izquierda y el centro, notándose más esta última, ya que corre una mancha clara que antes estaba más hacia la esquina. La media de los valores absolutos de las restas es 4.8125, aunque en algunas zonas es mayor.

JP2, Joint Photographic Experts Group 2000

JPEG 2000 es una norma de compresión de imágenes basada en la transformada discreta de wavelet. Fue creada por el comité Joint Photographic Experts Group que anteriormente había creado el algoritmo JPEG. Su objetivo fue el de mejorar el algoritmo JPEG, basado en la transformación discreta del coseno. Usualmente los archivos con este formato utilizan la extensión .jp2.

JPEG 2000 puede trabajar con niveles de compresión mayores a los de JPEG sin incurrir en los principales defectos del formato anterior con altas tasas de compresión: Generación de bloques uniformes y aspecto borroso. También se adapta mejor a la carga progresiva de las imágenes. Sus principales desventajas están en que tiende más a emborronar más la imagen que JPEG incluso para un mismo tamaño de archivo (pero sin

formar bloques), y que elimina algunos detalles pequeños y texturas, que el formato JPEG normal, si llega a representar.

Parte de JPEG 2000 ha sido publicada como una norma ISO, ISO/IEC 15444-1:2000. Actualmente JPEG 2000 no está ampliamente admitido por los programas de visualización de páginas web. En algunos navegadores, los diseñadores no tienen intención de incluirlo debido a su escaso uso y gran número de patentes que tiene. De todas formas, existen muchas extensiones que dan soporte, que opcionalmente pueden ser instaladas por el usuario. Un navegador con soporte para este formato es Konqueror.

Comparación con PNG

Si bien JPEG 2000 admite compresión sin pérdida, no está diseñado para reemplazar el formato PNG, que es uno de los más utilizados en la actualidad para este fin. PNG soporta algunas características, como la transparencia, que no están disponibles en JPEG 2000. Por las cuestiones inherentes a la compresión sin pérdida, de la cual PNG tiene mejor soporte y funcionalidad, este resulta como una mejor opción si lo deseado es almacenar fielmente y sin pérdidas, la imagen original.

Problemas legales por el uso de JPEG2000

JPEG 2000 no está ámpliamente admitido por los navegadores actuales por el peligro que desempeñan las Patentes de software en el método de compresión matemático, este area de matemáticas está empezando a ser muy patentado en general. JPEG 2000 no es por sí mismo una licencia libre, pero las compañías y organizaciones contribuyentes acordaron que las licencias para la primera parte — el corazón del sistema de codificación — pueden ser obtenidas libres de cargo desde todos los contribuidores.

El comité JPEG ha establecido:

> Es siempre una meta del comité JPEG que sus estándares puedan ser implementados en su forma basica sin pagar regalías (royalties) y licencias... El próximo estándar JPEG2000 ha sido preparado de acuerdo a estos lineamientos, y el acuerdo con otras 20 grandes organizaciones que mantienen muchas patentes en

> este área para permitir el uso de su propiedad intelectual en conexión con el estándar sin pagar licencias o regalías.

Aunque, el comité JPEG ha anotado además ha declarado que las patentes obscuras y no declaradas pueden ser un riesgo:

> Es posible por supuesto que otras compañías o particulares puedan reclamar derechos de propiedad intelectual que afecten la implementación del estándar, y muchos desarrolladores tienen que realizar sus propias búsquedas e investigaciones en este área.

MNG, Multiple-image Network Graphics

El MNG (pronunciado ming) es un formato de fichero, libre de derechos, para imágenes animadas. Las iniciales significan Multiple-image Network Graphics. El formato MNG está estrechamente vinculado al formato de imagen PNG.

Cuando comenzó el desarrollo de PNG a principios del año 1995, se decidió no incorporar la gestión de la animación, porque se empleaba poco esta capacidad del formato GIF en ese tiempo. Sin embargo, se desarrolló rápidamente un formato que soportaba la animación, el formato MNG, una extensión del formato PNG.

La versión 1.0 de las especificaciones de MNG salió el 31 de enero 2001. MNG es pues muy reciente, y actualmente no se actualiza tanto como el PNG. Sin embargo varios navegadores como Netscape Navigator y Konqueror ya soportan MNG; y los plugins de MNG están disponibles para Mozilla, Opera e Internet Explorer. Los desarrolladores de MNG esperan que MNG comience a sustituir a largo plazo al GIF para imágenes animadas en la Red, de la misma forma que el formato PNG ya comenzó a hacerlo para imágenes fijas.

La estructura de los ficheros de formato MNG es básicamente idéntica a la de los ficheros PNG, difiriendo solamente en la firma (8A 4D 4E 47 0D 1A 0A en hexadecimal) y en la utilización de unidades de información discretas que proporcionan una gran variedad de dispositivos de animación. Las imágenes utilizadas en la animación se almacenan en el fichero MNG como una encapsulación de imágenes con formato PNG o JNG.

Se crearon también dos versiones de MNG de complejidad reducida: MNG-LC (baja complejidad) y MNG-VLC (complejidad muy baja). Éstas permiten a las aplicaciones incluir el soporte de MNG en cierta medida, sin tener que poner todas las especificaciones de MNG.

MNG no dispone aún de un tipo registrado de soporte de vídeo MIME, pero se puede utilizar video/x-mng.

Son muy pocos los navegadores que soportan este formato. Safari no lo soporta. Mozilla retiró el soporte de MNG en la versión 1.5a y todas las futuras versiones, y no parece solucionarse al reinstalarlo por ahora, a pesar de las objeciones de la comunidad . El plugin oficioso de Firefox puede utilizarse para integrar el soporte en Mozilla. Hay que indicar que los programadores intentan crear una alternativa a Mozilla, llamada MNGzilla, integrando MNG.

PBM, Portable Bitmap Format

Formato simple para gráficos en blanco y negro. Utiliza 1 bit por pixel. A diferencia del resto de formatos gráficos, un fichero PBM contiene texto plano y puede ser modificado con un simple procesador de texto. Está relacionado con los formatos PGM (escala de grises) y PPM (color).

PCX, PiCture eXchange

PCX es un formato de imagen digital que usa la forma simple de la codificación run-length (un tipo de compresión sin perdidas).

PCX fue desarrollado por Zsoft Corporation de Marietta, Georgia (Estados Unidos). Fue el formato nativo para el programa Paintbrush de PC, el cual fue uno de los primeros programas de gráficos populares que funcionaban bajo DOS en los primeros PCs. Su popularidad también se debe a que era uno de los formatos utilizados por el Deluxe Paint, junto con el ILBM.

La mayoría de los archivos PCX usan una paleta de color indexada, pero el formato fue ampliado para permitir imágenes de 24 bits. PCX fue bastante popular en sistemas bajo

DOS o Windows, pero actualmente es poco común, siendo en buena parte reemplazado por formatos con mejor compresión y prestaciones, tal como el PNG o el JPEG.

Debido a que los colores en el índice 0x00..0xC0 se comprimen mejor que los colores 0xC1..0xFF, una buena ordenación de la paleta es importante. Normalmente es suficiente (aunque no siempre) con mover los colores más comunes a las posiciones de la paleta 0x00..0xC0, y los menos usados a las últimas posiciones de la paleta.

El algoritmo completo para ordenar la paleta es contar cuantas 63N+1 (N para enteros no negativos) veces aparece un color en una fila, y sólo entonces es posible utilizar valores de color sin prefijo para mejorar la compresión, y mover los colores con mayor cómputo en los índices 0x00..0xC0, y el resto a 0xC1..0xFF. Esto se justifica para producir resultados óptimos.

Este algoritmo de compresión es muy rápido y utiliza muy poca memoria, pero no es muy eficiente especialmente en fotografías.

PDF, Portable Document Format

PDF es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.

En esencia no es un formato gráfico propiamente dicho, sino un formato de almacenamiento de documentos, que permite almacenar texto con formato, imágenes de diferentes tipos, etc. Es una versión simplificada de PostScript; permite contener múltiples páginas y enlaces.

Está especialmente ideado para documentos susceptibles de ser impresos, ya que especifica toda la información necesaria para la presentación final del documento, determinando todos los detalles de cómo va a quedar, no requiriéndose procesos ulteriores de ajuste o maquetación.

Cada vez se utiliza también más como especificación de visualización, gracias a la gran calidad de las fuentes utilizadas y a las facilidades que ofrece para el manejo del documento, como búsquedas, hiperenlaces, etc.

Las características más importantes del formato PDF son:

- Es multiplataforma, es decir, puede ser presentado por los principales sistemas operativos (Windows, Linux o Mac), sin que se modifiquen ni el aspecto ni la estructura del documento original.
- Puede integrar cualquier combinación de texto, gráficos, imágenes e incluso música.
- Es uno de los formatos más extendidos en Internet para el intercambio de documentos. Por ello es muy utilizado por empresas, gobiernos e instituciones educativas.
- Es una especificación abierta, para la que se han generado herramientas de Software Libre que permiten crear, visualizar o modificar documentos en formato PDF. Un ejemplo es la suite ofimática OpenOffice.org.
- Puede cifrarse para proteger su contenido e incluso firmarlo digitalmente.

PGM, Portable Graymap Format

PGM es un formato de gráficos simple en escala de grises. Utiliza 8 bits por píxel. A diferencia del resto de formatos gráficos, un fichero PGM contiene texto plano y puede ser modificado con un simple procesador de texto. Está relacionado con los formatos PBM (blanco y negro) y PPM (color).

PIC, Picture

PIC es un formato utilizado en muchas aplicaciones gráficas que funcionan bajo MS-DOS y Windows, como PC Paint y Pictor.

Este formato puede almacenar una imagen de mapa de bits con dos posibilidades: 256 colores a una resolución máxima de 320 x 200 píxeles y 16 colores a una resolución de 640 x 480 píxeles. También puede almacenar una secuencia de imágenes en cada fichero, siendo en este caso sólo posible imágenes en tonos de gris.

Los datos de la imagen en el fichero vienen expresados como una matriz de bytes sin comprimir, en la que se almacena por filas el valor de los píxeles de la imagen o imágenes que lo forman. Hay que tener mucho cuidado al manejar ficheros con extensión .pic, ya que es utilizada por diferentes programas gráficos que producen ficheros incompatibles entre sí, como Lotus 1-2-3, Dr-Halo y Micrografx.

Las principales desventajas de este formato son la escasez de colores posibles en altas resoluciones y la incompatibilidad entre formatos PIC, que hacen que sea uno de los menos utilizados en la actualidad.

PNG , Portable Network Graphic

PNG es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujeto a patentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de color y otros importantes datos.

Historia y desarrollo

Las motivaciones para crear el formato PNG se generaron en 1995, después de que Unisys anunciara que haría cumplir la patente de software del algoritmo de compresión de datos LZW utilizado por el GIF (patente de EE.UU. 4.558.302 y otras alrededor del globo). Había otros problemas con el formato GIF que hacían deseable un cambio, por ejemplo su limitación a paletas de 8 bits de 256 colores como máximo, cuando los ordenadores ya soportaban miles o millones de colores.

Aunque el GIF soporta animación, el PNG se desarrolló como un formato de imagen estático y se creó el formato MNG como su variante animada.

El PNG ganó mayor popularidad en agosto de 1999 cuando Unisys puso fin a su política de licencias de patente libre de royalties para los desarrolladores de software libre o no comercial.

- La especificación de la versión 1.0 de PNG fue liberada el 1 de julio de 1996 y después apareció como RFC 2083. Rápidamente se convirtió en una recomendación W3C el 1 de octubre de 1996
- La versión 1.1 con algunos pequeños cambios y con 3 nuevas extensiones o "chunks" fue liberada el 31 de diciembre de 1998.
- Versión 1.2. Nueva extensión. Liberada el 11 de agosto de 1999
- Nueva versión, ligeramente diferente de la anterior y con una nueva extensión. Actualmente PNG es un estándar internacional (ISO/IEC 15948:2003), también recomendado por la W3C el 10 de noviembre de 2003.

Detalles técnicos

Las imágenes en formato PNG pueden ser imágenes de paleta indexada o estar formadas por uno o varios canales. Si existe más de un canal, todos los canales tienen el mismo número de bits por pixel (también llamado profundidad de bits por canal). Aunque en la especificación oficial del PNG se nombre la profundidad de bits por canal, normalmente los programas de edición nombran sólo la cantidad total de bits por pixel, es decir, la profundidad de color.

El número de canales depende de si la imagen es en escala de grises o en color y si dispone de canal alfa (también llamado canal de transparencia). La combinaciones permitidas por PNG son:

- Escala de grises (1 canal)
- Escala de grises y canal alfa (2 canales)
- Canales rojo, verde y azul (RGB, 3 canales. También llamado color verdadero o Truecolor)
- Canales rojo, verde, azul y alfa (RGB + alfa, 4 canales)

Por otra parte, las imágenes indexadas disponen de un tope de 256 colores como máximo. Esta paleta de colores está almacenada con una profundidad de canal de 8 bits. La paleta no puede tener más colores que los marcados por la profundidad de bits, es decir 28=256 colores, aunque sí puede tener menos (por ejemplo, una imagen de 50 colores sólo almacenará 50 entradas, evitando almacenar datos que no son utilizados).

Por otra parte, las imágenes indexadas disponen de un tope de 256 colores como máximo. Esta paleta de colores está almacenada con una profundidad de canal de 8 bits. La paleta no puede tener más colores que los marcados por la profundidad de bits, es decir 28=256 colores, aunque sí puede tener menos (por ejemplo, una imagen de 50 colores sólo almacenará 50 entradas, evitando almacenar datos que no son utilizados).

El método de compresión utilizado por el PNG es conocido como deflación (en inglés "Deflate algorithm"). También existen métodos de filtrado. En la especificación 1.2 se define un único tipo de filtro, que incluye 5 modos de predicción del valor del pixel, que resulta muy útil para mejorar la compresión, donde se elige para cada línea de la imagen (scanline) un método de filtrado que predice el color de cada pixel basándose en los colores de los pixeles previos y resta al color del pixel actual, el color pronosticado. Los cinco métodos son: None, Sub, Up, Average y Paeth.

Estos filtros pueden reducir notablemente el tamaño final del archivo, aunque depende en gran medida de la imagen de entrada. El algoritmo de compresión puede encargarse de la adecuada elección del método que mayor reducción ofrezca.

Profundidad de color y otros atributos

La tabla expuesta indica la profundidad de color para cada formato de imagen que soporta PNG. Ésta se extrae de la profundidad de bits por canal y se multiplica por el número de canales. Las casillas en rojo representan combinaciones no soportadas. El estándar requiere que los decodificadores puedan leer todos los formatos disponibles, pero muchos editores de imagen sólo pueden generar un pequeño subconjunto de ellos.

Otros atributos que pueden ser almacenados en un PNG incluyen valores de corrección gamma, color de fondo y metadatos. PNG además también utiliza la corrección de color que utilizan los sistemas de administración de color como el sRGB. Algunos programas como Adobe Photoshop disponen de este sistema.

Profundidad de bits por canal	1	2	4	8	16
Imagen indexada(1 canal)	1	2	4	8	
Escala de grises (1 canal)	1	2	4	8	16
Escala de grises con alfa (2 canales)				16	32
Color verdadero (RGB) (3 canales)				24	48
Color verdadero con alfa (RGBA) (4 canales)				32	64

Fig. 1.51.Rango total de opciones de color soportados por PNG

El tipo de media MIME para PNG es "image/png" (aprobado el 14 de octubre de 1996).

Fig. 1.52.Una imagen PNG con un canal alfa de 8 bits.

Fig. 1.53.Una imagen PNG con fondo transparente

Comparación técnica con otros formatos

Comparación con GIF

En la mayoría de los casos, PNG comprime mejor que el formato GIF, aunque algunas implementaciones (véase Photoshop) realizan una mala selección de los métodos de filtrado y se generan ficheros de mayor tamaño.

El PNG admite, al igual que el GIF, imágenes indexadas con transparencia de 1 bit o "binaria". Este tipo de transparencia no requiere de un canal adicional y únicamente admite que un color de la paleta aparezca transparente al 100%.

El PNG admite formatos con una profundidad de color de millones de colores (color verdadero) y canal alfa, lo que proporciona unos rangos de color mucho más ricos y precisos que el GIF y disponer de valores de transparencia intermedios. Desafortunadamente, esto permite que se compare erróneamente PNGs de color verdadero con un GIF de color indexado (256 colores)

Por contra, el PNG no soporta animación. MNG y APNG fueron diseñados para solventar esto, aunque no están muy extendidos.

Comparación con JPEG

PNG y JPEG son formatos que están diseñados para funciones diferentes, por lo que únicamente se puede realizar una comparación generalista.

- JPEG tiene un ratio de compresión enorme en perjuicio de la calidad de la imagen, ideal para imágenes de gran tamaño y fotografías. No admite trasparencia.
- PNG es un formato sin perdida de calidad con una excelente compresión, ideal para imágenes formadas por grandes áreas de color plano o con pocas variaciones de color. Admite canal alfa y algunos atributos extra como la corrección gamma.

PNG en la web

Pese a que las características técnicas y de compresión hacen del PNG un formato ideal para sustituir al GIF, su adopción ha sido muy lenta debido en parte a comparaciones erróneas y algunas desventajas técnicas:

- No está soportado por algunos navegadores muy viejos (sin embargo estos navegadores son muy raros hoy en día)
- No soporta animación
- La administración de color fallaba en algunos navegadores (actualmente no es muy importante y se puede evitar)

Falsas creencias:

- Internet Explorer no soporta PNGs transparentes. Habría que matizar un punto. Internet Explorer 6 e inferiores admiten transparencias binarias como en el GIF, pero fallan al mostrar imágenes con canal alfa. Eso se debe a que el paquete que especifica el canal alfa es opcional (tRNS) según la especificación PNGLas imágenes en PNG pesan más que los GIF. De nuevo, no es cierto. Esta falsa creencia es debido a que se compara con PNGs mal codificados o de 32 bits con GIFs de 256 colores.

Problemas de color

Algunas versiones de algunos navegadores presentan los valores de corrección gamma incluso cuando no están especificados en el PNG. Navegadores conocidos con problemas de visualización de PNG:

- Internet Explorer 5.5 y 6
- Netscape (Mozilla) 7.0 a 7.2
- Opera (versiones anteriores a la 7.50)

El efecto final es que el color mostrado en el PNG no coincide con el esquema de color del resto de la página web. Un forma sencilla de evitar esto es volviendo a codificar el PNG truncando ciertos atributos. Algunas utilidades para tal fin:

- PNGOUT es un utilidad gratuita de DOS que utiliza un algoritmo exclusivo para recomprimir un PNG y reducir el tamaño del fichero al máximo.
- Superpng, es un plugin gratuito para Photoshop que permite la optimización de ficheros PNG.

De manera informal PNG también se conoce como "PNG's Not GIF".

PPM, Portable Pixmap Format

PPM es un formato gráfico simple en color. Utiliza 24 bits por píxel: 8 para el rojo, 8 para el verde y 8 para el azul. A diferencia del resto de formatos gráficos, un fichero PPM contiene texto plano y puede ser modificado con un simple procesador de texto. Está relacionado con los formatos PGM (escala de grises) y PBM (blanco y negro).

PSD (Photoshop Digital Format)

PSD es el formato de mapa de bits (aunque con funcionalidades avanzadas) nativo del programa de tratamiento de imágenes Adobe Photoshop, válido para MAC y PC.

Es un formato sin compresión, por lo que no produce pérdidas de calidad, y admite todos los Modos de Color, canales alfa, tintas Planas, guías, trazados, selecciones, textos, capas simples y de ajuste y máscaras.

Soporta hasta 32 bits de profundidad de color en cualquier modo de color, produciendo imágenes de alta calidad que se pueden exportar, sin pérdida de calidad, a programas de auto edición y diseño como PageMaker, QuarkXpress, Ilustrator, etc. Incluso existen programas como CorelDraw que pueden abrir ficheros PSD manteniendo la estructura de capas original.

PSP, Documento de Paint Shop Pro

Formato estándar de los documentos de Paint Shop Pro, similares a los documentos .psd de Photoshop. Compatible con muy pocos programas.

SVG, Scalable Vector Graphics

SVG es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en XML. SVG se convirtió en una recomendación del W3C en Septiembre de 2001, por lo que ya ha sido incluido de forma nativa en el navegador web del W3C Amaya.

La versión 1.5 de Mozilla Firefox soporta gráficos hechos con SVG y desde su versión 8, también el navegador Opera ha implementado SVG 1.1 Tiny en su núcleo. Otros navegadores web, necesitan un conector o plug-in, para lo que se puede utilizar el Visualizador SVG de Adobe.

SVG rivaliza con Macromedia Flash en términos de potencial y poder, con la diferencia de que el primero es un estándar abierto.

Características

SVG permite tres tipos de objetos gráficos: Figuras de gráficos vectoriales (e.g., paths, que consisten en líneas rectas y curvas), imágenes y Texto.

Los objetos gráficos se pueden agrupar, estilizar, transformar y componer en objetos previamente renderizados. El texto puede estar en cualquier espacio para nombres XML adecuado para la aplicación, lo que extiende la facilidad de búsqueda y accesibilidad de los gráficos SVG.

El conjunto de características incluye transformaciones anidadas, truncamiento de rutas, máscaras alfa, efectos de filtro, objetos de plantillas y extensibilidad. Los dibujos SVG pueden ser dinámicos e interactivos. El Modelo de Objetos Documentales (DOM) para SVG, que incluye el DOM de XML completo, permite la animación directa y eficiente de los

gráficos vectoriales vía scripting. Se puede asignar un rico conjunto de manipuladores de eventos como onmouseover y onclick a cualquier objeto gráfico SVG.

Debido a su compatibilidad y potenciamiento de otros estándares web, se pueden hacer en elementos de SVG y otros elementos de XML de diversos espacios de nombres, características como el scripting simultáneo dentro de la misma página web.

SWF, ShockWave Flash

SWF es la extensión de los archivos creados con Macromedia Flash, y significa ShockWave Flash. Los archivos SWF pueden protegerse para que no sean editables, y son una compilación y compresión del archivo de autor (FLA) editable desde Flash.

Los fichero SWF están construidos principalmente por dos elementos: objetos basados en vectores e imágenes. Las versiones más modernas también incorporan audio, vídeo (en formato Flash Video-FLV) y multitud de formas diferentes de interacción con el usuario.

Una vez creados, los ficheros SWF pueden ser ejecutados por el reproductor Macromedia Flash Player, tanto en formato plugin de un navegador o como aplicación autónoma. En muchas ocasiones es posible encapsular los ficheros SWF junto con el reproductor, creando un proyector autónomo que reproduce la animación que contiene cuando se ejecuta.

El formato SWF fue desarrollado por Macromedia y tiene un objetivo principal: crear ficheros de reducido tamaño pero de gran calidad que permita interactividad. La idea fue disponer de un formato que pudiese funcionar sobre cualquier sistema y sobre un reducido ancho de banda (cómo un navegador de internet conectado a través de un módem).

El plugin que permite reproducir ficheros SWF está disponible en Macromedia para diferentes navegadores y diferentes sistemas operativos, incluido Microsoft Windows, Apple Macintosh y Linux. Este plugins está instalado en un 98% de los ordenadores de los internautas. El formato es bastante simple, si bien es cierto que está en formato binario y por lo tanto no es de lectura accesible como su rival SVG basado en XML.

SWF ha utilizado la compresión Zlib desde el 2002, y en general el objetivo del formato es almacenar todos los datos usando el menor número de bits, minimizando la redundancia. SWF es un formato abierto, es decir, es posible crear programas que generen este formato sin tener que pagar royalties, aunque no es posible crear reproductores.

TGA, Truevision TGA

El formato TGA es un formato gráfico de mapa de bits desarrollado por la empresa Truevision para las tarjetas Targa y Vista, válido para PC y MAC, que permite guardar imágenes monocromáticas (2 bits) y con diferentes niveles de profundidad de color (8, 16, 24 y 32 bits), utilizando o no una paleta gráfica. Puede trabajar en Escala Grises, Color Indexado, RGB (16 y 24 bits sin canales alfa) y RGB de 32 bits (un solo canal alfa).

Permite almacenar los archivos comprimidos o sin comprimir, aunque la mayoría de programas que lo soportan solo pueden abrir archivos TGA sin compresión, siendo entonces el que el peso de los ficheros es muy elevado.

Este formato está especialmente indicado para retocar diseños profesionales que se vayan a reproducir en pantalla, debido a que la amplia gama de colores produce un efecto muy realista y sumamente elaborado. También es muy útil cuando se trabaja con escáneres de alta calidad y para la exportación de imágenes a edición profesional vídeo. Sin embargo, en impresión es poco usado, ya que con profundidades de color de 16 bits o menos las imágenes pierden detalles.

Las principales desventajas de este formato son el tamaño de los archivos, que ocupan bastante más espacio que otros formatos de igual calidad, y que no guarda muchos detalles a veces necesarios, como la resolución que soporta.

TIFF, Tagged Image File Format

La denominación en inglés Tagged Image File Format (formato de archivo de imágenes con etiquetas) se debe a que los ficheros TIFF contienen, además de los datos de la imagen propiamente dicha, "etiquetas" en las que se archiva información sobre las características de la imagen, que sirve para su tratamiento posterior.

Es capaz de almacenar imágenes en blanco y negro (1 bit), escala de grises (9 bits), RGB (24 bits), CMYK (32 bits) con más de diez técnicas de compresión disponibles (sin compresión, LZX, JPEG, MAC Packbit, etc.) y Color Lab.

Soporta el algoritmo de compresión sin pérdidas LZW, almacenando las imágenes en una serie de bloques que pueden contener información sobre la imagen en sí, su tamaño, su manejo del color, información a las aplicaciones que utilicen ese archivo, texto e incluso una miniatura (thumbnail), pequeña representación de la imagen, a la cual el programa accede rápidamente y no pierde tiempo descomprimiendo toda la imagen.

Es válido para todas las plataformas, siendo soportado por multitud de programas gráficos de pintura e ilustración, por lo que es muy utilizado para intercambiar archivos entre diferentes aplicaciones y plataformas.

Es uno de los formatos más utilizados en artes gráficas, así cómo en fotografías en las que queremos que no haya ninguna perdida, bien para imprimirla o bien para interpolarla para aumentar su resolución. También es el formato más usado cuando se trabaja con escáneres, debido a su útil manejo del color.

Como desventajas, no tiene soporte para vectores ni texto, por lo que todos los tipos deben ser convertidos a mapas de bits antes de aplicarse al archivo. También, que debido a la flexibilidad que presenta respecto a los sistemas de compresión y a la compatibilidad entre aplicaciones, los programas diseñados para leer archivos TIFF deben disponer de la misma flexibilidad para entender los datos contenidos en ellos, lo que desafortunadamente no siempre ocurre.

Etiquetas

Las etiquetas que utiliza describen el formato de las imágenes almacenadas, que pueden ser de distinta naturaleza:

- Binarias (blanco y negro), adecuadas para textos, por ejemplo.
- Niveles de gris, adecuadas para imágenes de tonos continuos como fotos en blanco y negro.

- Paleta de colores, adecuadas para almacenar diseños gráficos con un número limitado de colores.
- Color real, adecuadas para almacenar imágenes de tono continuo, como fotos en color.

Las etiquetas también describen el tipo de compresión aplicado a cada imagen, que puede ser:

- Sin compresión
- PackBitsHuffman modificado, el mismo que las imágenes de fax (UIT grupo III y IV anteriormente CCITT).
- LZW, el mismo que usa el formato GIF.
- JPEG

Hay también etiquetas que especifican el formato interno de almacenamiento de la imagen: completas, por bandas o por secciones rectangulares, lo cual permite a muchas aplicaciones optimizar los tiempos de carga o leer únicamente la zona de interés de una imagen grande.

Un aspecto muy práctico del formato TIFF es que permite almacenar más de una imagen en el mismo archivo.

Un mito que ha de desterrarse es la idea de que el formato TIFF no permite comprimir las imágenes. No obstante, algunas cámaras fotográficas digitales ofrecen la opción de grabar fotos en el formato TIFF, lo cual suele entenderse como “sin compresión”.

Creadores y Dueños

El formato TIFF fue desarrollado por Aldus y Microsoft, y es actualmente propiedad de Adobe Systems. La última revisión del formato es la número 6, del año 1992. Hay algunas extensiones, como las anotaciones que utiliza el Imaging de Microsoft, pero ninguna puede considerarse estándar.

WMF, Windows MetaFile Format

WFM es un metaformato de 16 bits de los sistemas operativos Windows, siendo un estándar de intercambio de gráficos entre las diferentes aplicaciones Microsoft (Word, Excel, Access, etc.).

WMF es un formato vectorial (aunque no basado en curvas de Bézier) y escalable, que funciona copiando en un archivo los comandos para realizar la imagen en cuestión, ahorrando con ello una cantidad considerable de espacio. Teóricamente puede almacenar cualquier elemento gráfico, ya sean imágenes bitmap, textos o gráficos vectoriales complejos.

XBM, X BitMap

Formato nativo en blanco y negro del sistema X Window, compatible con la mayoría de navegadores web. Se trata de un formato ASCII sin compresión diseñado de tal forma que los ficheros tienen sintaxis de C/C++, pudiendo ser incluidos en el código fuente.

XCF, eXperimental Computing Facility

Formato nativo para el programa The GIMP (OpenSource), con múltiples características extra, como la composición por capas. Usado, sobre todo, en The GIMP, pero también leíble por ImageMagick.

X-Pixmap (.xpm)

Inspirado en el formato XBM, es usado casi exclusivamente en plataformas UNIX, Linux, BSD con el sistema X Windows.

Resumen

Los gráficos por computadora son un herramienta versátil que representa una ventaja que puede aplicarse a campos diversos. Los descubrimientos de autores como Euclides, Descartes y Schoenberg representan la base de los gráficos.

La historia de la graficación por computadora comienza con el Proyecto Whirlwind y el sistema computacional SAGE; el lápiz luminoso de la SAGE fue uno de los primeros dispositivos de hardware utilizados para la graficación.

El Sketchpad de Ivan Sutherland en 1963 permitía el diseño interactivo con el uso de lápiz luminoso, este hecho es considerado por muchos como el nacimiento de los gráficos por computadora. En 1966 se comenzó el desarrollo del primer vídeo juego doméstico denominado Fox Hounds.

Otros descubrimientos e invenciones importantes en los 1960s fueron las curvas paramétricas, la transformada de Furier, el mouse en los laboratorios Xerox PARC y desarrollo de algoritmos como los de sombreado, iluminación, z-buffer y mapeo de textura.

En los 1971 se comenzó a comercializar el primer video juego como arcade: Computer Space. Pong también marcó un hito en la historia dando pie a la creación de consolas. La microcomputadora Altair de MITS condujo a la revolución de el computador personal.

En 1976 la Apple I fue el primer éxito de la computación personal, en ese mismo año la película “Futureworld” incluyó la animación de un rostro y mano humanos siendo la primera en utilizar los gráficos por computadora.

En 1981 la IBM PC comenzó a venderse popularizando el término “computadora personal”. En 1982 se lanzó al mercado la primera tarjeta de vídeo denominada Hércules la cual solo servía para gráficos de un solo color.

Apple Lisa la primera computadora comercial con interfaz gráfica y ratón, se lanzó en 1983. En 1987 se creó en IBM la primeta tarjeta gráfica VGA. En 1989 Adobe Photoshop comenzó a comercializarse siendo hoy una aplicación popular.

En 1992 OpenGL se convirtió en un estándar de APIs gráficas. El primer navegador gráfico de internet se creó en 1993 con el nombre de Moisaic. En esta época comenzacón a utilizarse los estándares de MPEG.

El Super Mario 64 salió a la venta y es considerado el mejor juego de plataformas; este juego creó el primer sistema de control de cámaras en un juego en 3 dimensiones.

En 1995, se anunció el formato DVD utilizando la especificación desarrollada por 10 compañías líderes. En 1996 el puerto AGP fue inventado por Intel solucionando los cuellos de botella producidos en las tarjetas con bus PCI.

En la actualidad todas las computadoras utilizan tarjetas de vídeo con capacidad de mostrar colores reales. Se hace uso de dispositivos como cámaras digitales, tabletas digitalizadoras, mouse y monitores de alta resolución.

En los procesos de diseño se hace un uso importante de las gráficas por computadora, en particular para sistemas de ingeniería y arquitectura a través de los métodos CAD. El diseño asistido por computadora (CAD, Computer Aided Design) se trata básicamente de una base de datos de entidades geométricas con la que se puede operar a través de una interfaz gráfica.

El arte digital utiliza las gráficas por computadora incluyendo hardware de propósito especial, software y paquetes CAD. Un artista puede dibujar utilizando la tableta digitalizadora para crear ilustraciones tan complejas como se desee. Las aplicaciones contienen herramientas como paintbrush o brochas que tienen propiedades como color, textura, tamaño y presión de trazo.

Existe también el arte digital matemático que hace uso de funciones matemáticas y procedimientos fractales. El ate digital a diferencia del convencional ofrece ventajas como el uso de capas y no se desgasta con el tiempo.

La animación por computadora permite reducir costos de producción y edición en el ámbito del cine y la televisión. El uso de internet ha permitio poner al alcance de todos la animación. Aplicaciones como Flash y Moho, permiten que casi cualquier persona pueda hacer una animación. Aplicaciones más sofisticadas como Mzaya permiten hacer animaciones en 2D y 3D.

La programación de videojuegos está involucrada con la animación y se complementan. La industria de los juegos a crecido en tal medida que algunas universidades ofrecen programas académicos de programación en videojuegos.

Las gráficas por computadora se utilizan en diversas etapas de la creación de películas. Se puede utilizar la animación, edición y efectos especiales, siendo los efectos especiales lo que mas llama la atención entre los consumidores.

Las gráficas también se utilizan en aplicaciones de educación y capacitación. Los simuladores para sesiones de práctica o capacitación de capitanes de barco, pilotos de avión, operadores de equipo pesado y personal de tráfico aéreo permiten a los estudiantes a comprender mejor la operación del sistema y a practicar en un entorno virtual muy apegado a la realidad.

El procesamiento de imágenes se utiliza en áreas muy diversas. Las principales aplicaciones del procesamiento de imágenes son: el mejoramiento de la calidad de la imagen y la percepción de la máquina de información visual.

Los formatos gráficos de archivos son la forma en que la información de una imagen se guarda en un archivo. Cada formato utiliza técnicas de codificación especial, métodos de compresión y otras técnicas de para buscar el equilibro entre la calidad, el pes del fichero y la compatibilidad entre plataformas.

En general, todos los ficheros gráficos comienzan con una cabecera que indica las características de la imagen, después se encuentran los datos de la imagen los cuales suelen estar comprimidos.

Existen dos tipos de formatos: los vectoriales y los de mapa de bits también conocidos como rasterizados. Los rasterizados se distinguen de los vectoriales en que estos últimos representan una imagen a través del uso de objetos geométricos como curvas y polígonos mientras que los mapas de bits se almacenan como un conjunto de pixeles.

2. Transformaciones geométricas

Objetivo. *El estudiante conocerá y aplicará las principales transformaciones geométricas sobre objetos en 2D y 3D.*

Habitualmente, un paquete gráfico permite al usuario especificar que parte de una imagen definida se debe visualizar y dónde esta parte se debe colocar en el dispositivo de visualización. Cualquier sistema de coordenadas que sea conveniente, referido al sistema de referencia de coordenadas del mundo, se puede usar para definir la imagen.

En el caso de las imágenes bidimensionales, una vista se selecciona especificando una región del plano *xy* que contiene la imagen total o cualquier parte de ella. Un usuario puede seleccionar una única zona para visualización, o varias zonas para visualización simultánea o para una secuencia animada panorámica a través de una escena.

Las partes dentro de las zonas seleccionadas se mapean entonces sobre zonas especificadas de las coordenadas del dispositivo. Cuando se seleccionan múltiples zonas de vista, estas zonas se pueden colocar en ubicaciones de visualización independientes, o algunas zonas se podrían insertar en zonas de visualización más grandes. Las transformaciones de visualización bidimensional desde las coordenadas universales a las coordenadas del dispositivo implican operaciones de traslación, rotación y cambio de escala, así como procedimientos de borrado de aquellas partes de la imagen que se encuentran fuera de los límites de una zona seleccionada de la escena.

2.1.Transformaciones bidimensionales

Traslación

Se aplica una traslación en un objeto para cambiar su posición a lo largo de la trayectoria de una línea recta de una dirección de coordenadas a otra. Convertimos un punto bidimensional al agregar las distancias de traslación, t_x y t_y a la posición de coordenadas original (x, y) para mover el punto a una nueva posición (x', y').

2.1. $$x' = x + t_{x'} \qquad y' = y + t_y$$

El par de distancia de traslación ($t_{x'}$, t_y) se llama vector de traslación o vector de cambio.

Se pueden expresar las ecuaciones anteriores en una sola ecuación matricial al utilizar vectores de columna para representar las posiciones de coordenadas y el vector de traslación:

2.2. $$P = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \qquad P' = \begin{bmatrix} X'_1 \\ X'_2 \end{bmatrix} \qquad T = \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Esto nos permite expresar las dos ecuaciones de traslación bidimensional en la forma de matriz:

2.3. $$P' = P + T$$

en ocasiones, las ecuaciones de transformación matricial se expresan en términos de vectores de renglón de coordenadas en vez de vectores de columna. En este caso, expresaríamos las representaciones matriciales como $P[x\ y]$ y $T[t_x\ t_y]$.

La traslación es una transformación de cuerpo rígido que mueve objetos sin deformarlos, es decir, se traslada cada punto del objeto la misma distancia. Se traslada un segmento de línea recta al aplicar la ecuación de transformación 2.3 en cada uno de los

extremos de la línea y se vuelve a trazar la línea entre las nuevas posiciones de los extremos.

Los polígonos se trasladan al sumar el vector de traslación a la posición de coordenadas de cada vértice y se vuelve a generar el polígono utilizando un nuevo conjunto de coordenadas y vértices y las especificaciones actuales de los atributos.

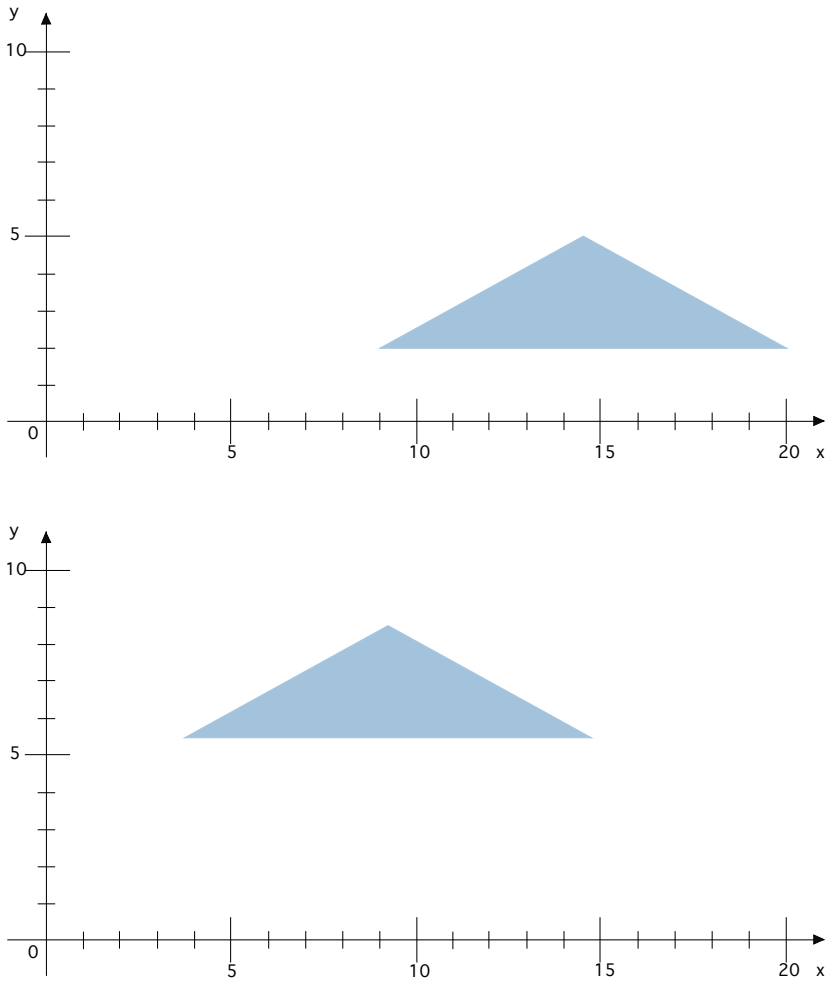


Fig. 2.1.Movimiento de un polígono de la posición a) a la posición b) con el vector de traslación (-5.50, 3.75)

Rotación

Se aplica una rotación bidimensional en un objeto al cambiar su posición a lo largo de la trayectoria de una circunferencia en el plano de *xy*. para generar una rotación, especificamos un ángulo de rotación θ y la posición (x_r, y_r) del punto de rotación (o punto pivote) en torno al cual se gira el objeto.

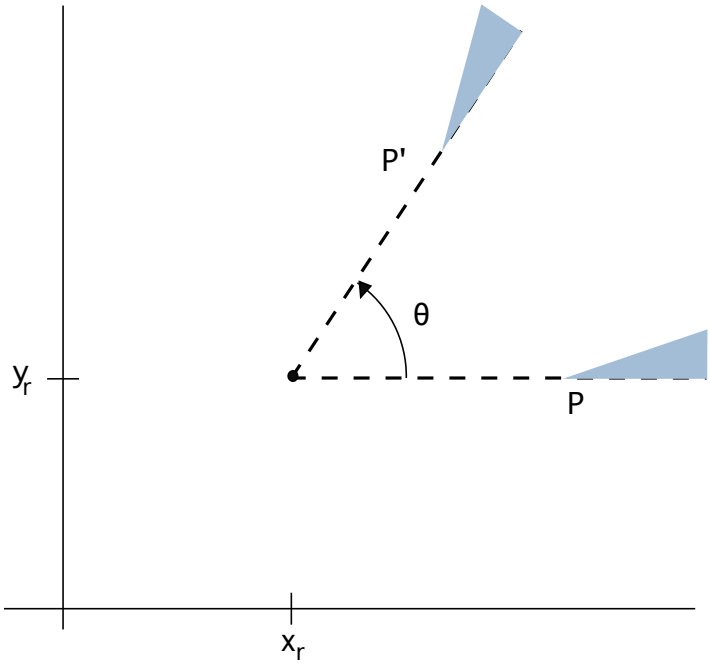


Fig. 2.2.Rotación de un objeto a través del ángulo θ alrededor del punto pivote

Los valores positivos para el ángulo de rotación definen rotaciones en sentido opuesto a las manecillas del reloj alrededor del punto pivote y los valores negativos giran los objetos en la dirección del reloj . También es posible describir esta transformación como una rotación sobre el eje de rotación que es perpendicular al plano de *xy* y pasa a través del punto pivote.

El primer paso es determinar las ecuaciones de transformación para la rotación de la posición de un punto P cuando el punto pivote está en el origen de las coordenadas.

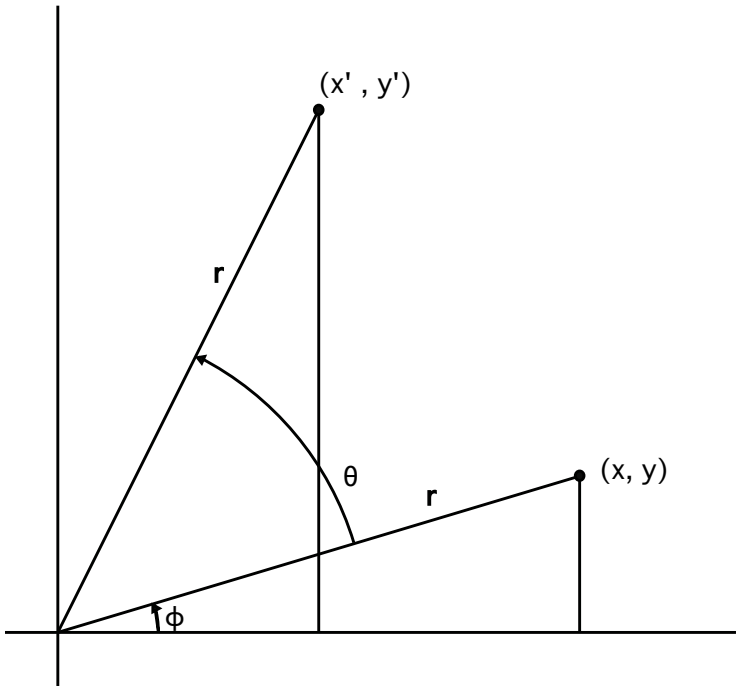


Fig. 2.3.Rotación de un punto desde la posición (x, y) a la posición (x', y') a través de un ángulo θ con respecto del origen de las coordenadas. El desplazamiento angular original del punto desde el eje de las x es ϕ.

En la figura 2.3, *r* es la distancia entre constante del punto desde el origen, el ángulo ϕ es la posición angular original del punto desde el plano horizontal y θ es el ángulo de rotación. Utilizando identidades trigonométricas podemos expresar las coordenadas transformadas en término de los ángulos θ y ϕ como:

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta - r\sin\phi\cos\theta \quad \text{2.4.}$$

Las coordenadas originales del punto en las coordenadas polares son

$$x = r\cos\phi,\ y = r\sin\theta \quad \text{2.5.}$$

Al sustituir las expresiones 2.5 en 2.6, obtenemos las ecuaciones de transformación para girar un punto en la posición (x, y) s través de un ángulo θ al rededor de un origen:

$$x' = x\cos\theta - y\sin\theta$$
$$y = x\sin\theta + y\cos\theta \quad \text{2.6.}$$

Con las representaciones del vector de columna 2.2 para las posiciones de coordenadas podemos expresar las ecuaciones de rotación en la forma de matriz:

2.7. $$P' = R \cdot P$$

donde la matriz de rotación es:

2.8. $$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Cuando las posiciones de coordenadas se representan como vectores de renglón en vez de vectores de columna, el producto de la matriz en la ecuación de rotación 2.7 se transpone, de modo que el vector de coordenadas de renglón transformado [*x' y'*] se calcula como

$$P'^{T} = (R \cdot P)^{T} = P^{T} \cdot R^{T}$$

donde $P^T = [x\ y]$ y se obtiene la transposición R^T de la matriz R con sólo cambiar el signo de los términos del seno.

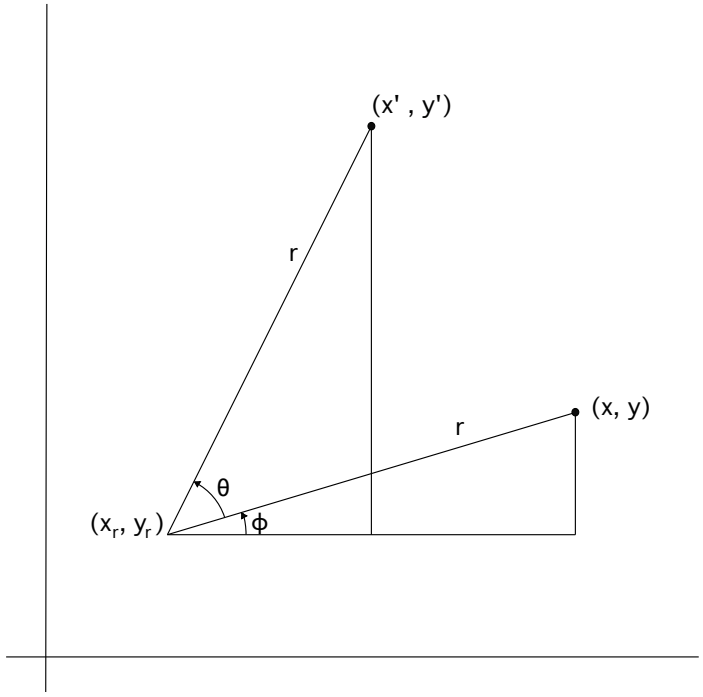


Fig. 2.4.Rotación de un punto desde la posición (x,y) a la posición (x', y') a través de un ángulo θ con respecto del punto de rotación (xr, yr)

En la figura 2.4 se ilustra la rotación de un punto alrededor de una posición pivote arbitraria. Al utilizar las relaciones trigonométricas en esta figura, podemos generalizar las ecuaciones 2.6 para obtener las ecuaciones de transformación para la rotación de un punto con respecto de cualquier posición de rotación específica (x_r, y_r):

$$x' = x_r + (x + x_r)\cos\theta - (y - y_r)\sin\theta$$
$$y' = y_r + (x + x_r)\sin\theta - (y - y_r)\cos\theta \qquad 2.9.$$

Estas ecuaciones difieren de las ecuaciones de rotación 2.6 por la inclusión de términos aditivos, así como los factores de multiplicación en los valores de las coordenadas.

Sl igual que con las traslaciones, las rotaciones son transformaciones de cuerpos rígidos que mueven los objetos sin deformarlos. Se giran a través del mismo ángulo todos los puntos del objeto.

Se gira un segmento de linea recta al aplicar las ecuaciones de rotación en cada extremo de la línea y se vuelve a trazar la línea entre las nuevas posiciones de los extremos . Los polígonos se giran al desplegar cada vértice a través del ángulo de rotación específico y se vuelve a generar el polígono utilizando los nuevos vértices.

Las líneas curvas se giran al cambiar la posición de los puntos de definición y se vuelven a trazar las curvas. Por ejemplo, se puede girar una circunferencia o una elipse alrededor de un eje no central al mover la posición del centro a través del arco que subtiende el ángulo de rotación específico. Es posible girar una elipse sobre sus coordenadas de centro al girar los ejes mayor y menor.

Escalación

Una transformación de escalación altera el tamaño de un objeto. Se puede realizar esta operación para polígonos al multiplicar los valores de coordenadas *(x, y)* de cada vértice por los factores de escalación s_x y s_y para producir las coordenadas transformadas (*x', y'*):

$$x' = x \cdot s_x,\ y' = y \cdot s_y \qquad 2.10.$$

El factor de escalación s_x escala objetos en la dirección de *x* mientras que el factor de escalación s_y lo hace en dirección de *y*. También se pueden expresar las ecuaciones de transformación 2.10 en forma matricial:

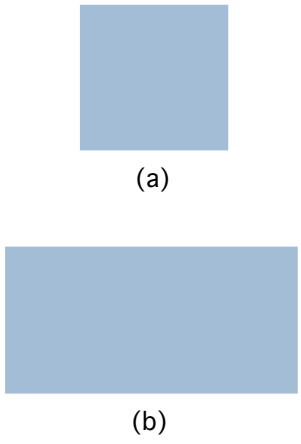
2.11. $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

o

2.12. $$P' = S \cdot P$$

donde *S* es la matriz de escalación 2 por 2 en la ecuación 2.11.

Se pueden asignar valores numéricos positivos cualesquiera a los factores de escalación s_x y s_y.Los valores menos que 1 reducen el tamaño de los objetos y los valores mayores que 1 producen una ampliación. Al especificar un valor de 1 tanto para s_x como para s_y no se altera el tamaño de los objetos. Cuando se asigna el mismo valor a s_x y s_y se genera una escalación uniforme que mantiene las proporciones relativas de los objetos. Cuando s_x y s_y tienen valores distintos se obtiene una escalación diferencia como la de la figura siguiente.



(a)

(b)

Fig. 2.5.Conversión de un cuadrado (a) en un rectángulo (b) con los factores de escalación sx = 2 y sy = 1.

Los objetos que se transforman con las ecuaciones 2.11 se escalan y cambian de posición. Los factores de escalación con valores menores que 1 acercan los objetos al origen de las coordenadas en tanto que los valores mayores que 1 alejan las posiciones de coordenadas del origen.

Podemos controlar la localización de un objetos escalado al seleccionar una posición llamada punto fijo, que debe permanecer sin cambios después de la transformación de escalación.

Se pueden seleccionar las coordenadas para el punto fijo (x_f, y_f) como uno de los vértices, el centroide del objeto o cualquier otra posición.

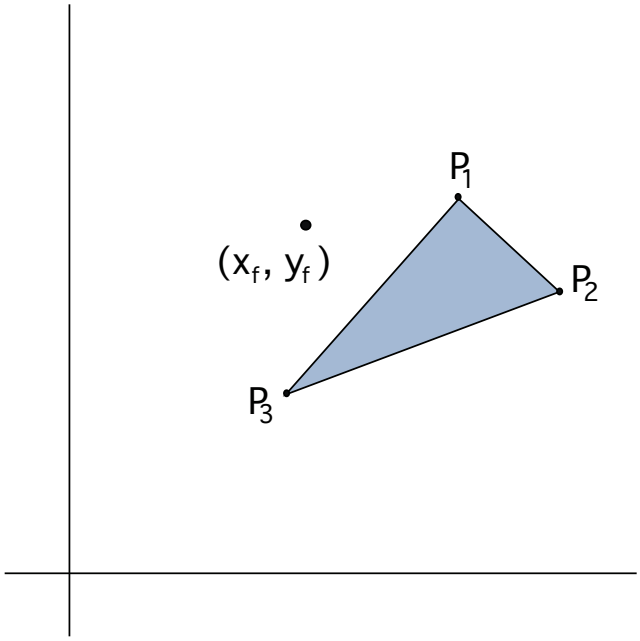


Fig. 2.6.Escalación con respecto de un punto fijo seleccionado (xf, yf). Las distancias desde el polígono a el punto fijo se escalan mediante las ecuaciones de transformación.

Así, se escala un polígono con respecto del punto fijo al escalar la distancia de cada vértice al punto fijo. Para un vértice con coordenadas (x, y), se calculan las coordenadas escaladas (x', y') como

2.13. $$x' = x_f + (x - x_f)s_x, \qquad y' = y_f + (y - y_f)s_y$$

Podemos volver a expresar estas transformaciones de escalación para separar los términos de multiplicación y de adición donde los términos aditivos $x_f(1 - s_x)$ y $y_f(1 - s_y)$ son constantes para todos los puntos en el objeto

$$x' = x \cdot s_x + x_f(1 - s_x)$$
$$y' = y \cdot s_y + y_f(1 - s_y) \qquad 2.14.$$

Incluir coordenadas para un punto fijo en las ecuaciones de escalación es similar a incluir coordenadas para un punto pivote en las ecuaciones de rotación.

Se escalan polígonos al aplicar las transformaciones 2.14 en cada vértice y luego volver a generar el polígono utilizando los vértices transformados . Se escalan otros objetos al aplicar las ecuaciones de transformación de escalación a los parámetros que definen los objetos.

2.2.Coordenadas homogéneas y representación matricial

En las aplicaciones de diseño y de creación de imágenes, realizamos traslaciones, rotaciones y escalaciones para ajustar los componentes de la imagen en sus posiciones apropiadas. En este tema consideramos cómo se pueden volver a formular las representaciones de la matriz de modo que se pueden procesar de manera eficiente esas secuencias de transformación.

Es posible expresar cada una de las transformaciones básicas en la forma de matriz general con las posiciones de coordenadas P y P' representadas como columnas de vector.

$$P' = M_1 \cdot P + M_2 \qquad 2.15.$$

La matriz M_1 es una matriz de 2 por 2 que contiene factores de multiplicación y M_2 es una matriz de columnas de dos elementos que contiene términos de traslación. Para la traslación, M_1 es la matriz de identidad. Para la rotación o la escalación M_2 contiene los términos de traslación asociados con el punto pivote o el punto fijo de escalación.

Con el fin de producir una secuencia de transformaciones con estas ecuaciones, como escalación seguida por rotación y luego traslación, debemos calcular las coordenadas transformadas un paso a la vez. Primero, se escalan las posiciones de coordenadas, después se giran estas coordenadas escaladas y por último se trasladan las coordenadas giradas.

Un planteamiento mas eficiente combinaría las transformaciones de manera que se obtengan las coordenadas finales directamente a partir de las coordenadas iniciales para eliminar el cálculo de coordenadas intermedias. De esta manera, se debe de formular nuevamente la ecuación 2.15 para eliminar la adición de la matriz asociada con los términos de la traslación M_2.

Podemos combinar los términos de multiplicación y de adición para transformaciones geométricas dibimensionales en una sola representación de matriz al ampliar las representaciones de matriz de 2 por 2 a matrices de 3 por 3.

Esto nos permite expresar todas las ecuaciones de matriz como multiplicaciones de matriz , si también ampliamos las representaciones de matriz para las posiciones de coordenadas. Para expresar cualquier transformación bidimensional como una multiplicación de matriz, representamos cada posición de coordenadas cartesianas (x, y) con las tres coordenadas homogéneas (x_h, y_h, h), donde

2.16. $$x = \frac{x_h}{h}, \qquad y = \frac{y_h}{h}$$

Por tanto, una representación general de coordenadas homogéneas se puede expresar también como ($h \cdot x, h \cdot y, h$). Para transformaciones geométricas bidimensionales, seleccionamos el parámetro homogéneo h como cualquier valor no cero. Así, existe un número finito de representaciones homogéneas equivalentes para cada punto de coordenadas (x, y).

Una opción conveniente consiste en sólo establecer $h = 1$. Entonces, se representa cada posición bidimensional con las coordenadas homogéneas ($x, y, 1$). Se requieren otros valores para el parámetro h, por ejemplo, en las formulaciones de matriz de transformaciones de vista tridimensionales.

En matemáticas se utiliza el término coordenadas homogéneas para referirse al efecto de esta representaciones de ecuaciones cartesianas. Cuando se convierte un punto cartesiano (x, y) a una representación homogénea (x_h, y_h, h) las ecuaciones que contienen x y y como $f(x, y) = 0$, se convierten en ecuaciones homogéneas en los tres parámetros x_h, y_h, h.

Esto sólo significa que si se sustituye cada uno de los tres parámetros con cualquier valor v veces ese parámetro, el valor v se puede factorizar fuera de las ecuaciones.

Expresar posiciones en coordenadas homogéneas nos permite representar todas las ecuaciones de transformación geométrica como multiplicaciones de matriz. Se representan las coordenadas con vectores de columna de tres elementos y las operaciones de transformación se expresan como matrices de 3 por 3. Para la traslación tenemos:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_y \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad 2.17.$$

que podemos expresar en forma abreviada

$$P' = T(t_x, t_y) \cdot P \qquad 2.18.$$

con $T(t_x, t_y)$ como la matriz de traslación 3 por 3 en la ecuacion 2.17. Se obtiene el inverso de la matriz de traslación al reemplazar los parámetros de traslación t_x y t_y con sus valores negativos $-t_x$ y $-t_y$.

De modo similar, ahora se expresan las ecuaciones de transformación de rotación respecto al origen de las coordenadas como

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad 2.19.$$

o como

2.20. $$P' = R(\theta) \cdot P$$

El operador de transformación de rotación R(θ) es la matriz de 3 por 3 en la ecuación 2.19 con el parámetro de rotación θ. Obtenemos la matriz de rotación inversa cuando se sustituye θ con -θ.

Por último, ahora se expresa una transformación de escalación con respecto del origen de las coordenadas como la multiplicación de matriz

2.21. $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

o

2.22. $$P' = S(s_x, s_y) \cdot P$$

donde $S(s_x, s_y)$ es la matriz de 3 por 3 en la ecuación 2.21 con los parámetros s_x y s_y. Al sustituir sus inversos multiplicativos ($1/s_x$ y $1/s_y$) se obtiene la matriz de escalación inversa.

Las representaciones de matriz o representaciones matriciales son métodos estándar para implementar transformaciones en sistemas de gráficas. En muchos sistemas, las funciones de rotación y escalación producen transformaciones con respecto del origen de las coordenadas como en las ecuaciones 2.19 y 2.21.

2.3.Composición de transformaciones bidimensionales

Con las representaciones de matriz del tema anterior, podemos establecer una matriz para cualquier secuencia de transformaciones como una matriz de transformación compuesta al calcular el producto de la matriz de las transformaciones individuales. La creación de productos de matrices de transformación a menudo se conoce como concatenación o composición de matrices.

Traslaciones, rotaciones y escalaciones

Traslaciones

Se se aplican dos vectores de traslación sucesivos (t_{x1}, t_{y1}) y (t_{x2}, t_{y2}) en la posición de coordenadas P, la localización transformada final P, la localización transformada final P' se calcula como:

2.23. $$P' = T(t_{x2}, t_{y2}) \cdot \{T(t_{x1}, t_{y1}) \cdot P\} = \{T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1})\} \cdot P$$

donde se representan P y P' como vectores de columna de coordenadas homogéneas. Podemos verificar este resultado al calcular el producto de la matriz para las dos agrupaciones asociativas. Asimismo, la matriz de transformación compuesta para esta secuencia de transformaciones es

2.24. $$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{x1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

ó

2.25. $$T(t_{x2}, t_{y2}) \cdot T(t_{x1}, t_{y1}) = T(t_{x1} + t_{x2}, t_{y1} + t_{y2})$$

que demuestra que dos transformaciones sucesivas son aditivas.

Rotaciones

Dos rotaciones sucesivas que se aplican en el punto *P* producen la posición transformada

2.26. $$P' = R(\theta_2) \cdot \{R(\theta_1) \cdot P\} = \{R(\theta_2) \cdot R(\theta_1)\} \cdot P$$

Al multiplicar las dos matrices de rotación, podemos verificar que dos rotaciones sucesivas son aditivas:

2.27. $$R(\theta_2)\cdot R(\theta_1) = R(\theta_1 + \theta_2)$$

de modo que es posible calcular las coordenadas giradas finales con la matriz de rotación compuesta como

2.28. $$P' = R(\theta_1 + \theta_2)\cdot P$$

Escalaciones

Concatenar matrices de transformación para dos operaciones de escalación sucesivas produce la siguiente matriz de escalación compuesta:

2.29. $$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1}\cdot s_{x2} & 0 & 0 \\ 0 & s_{y1}\cdot s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ó

2.30. $$S(s_{x2}, s_{y2})\cdot S(s_{x1}, s_{y1}) = S(s_{x1}\cdot s_{x2}, s_{y1}\cdot s_{y2})$$

La matriz resultante en este caso indica que las operaciones de escalación sucesivas son multiplicativas. Es decir, si debiéramos triplicar el tamaño de un objeto dos veces en una sucesión, el tamaño final sería nueve veces el tamaño original.

Rotación del punto pivote general

Con un paquete gráfico que sólo ofrezca una función de rotación para girar objetos con respecto del origen de las coordenadas, podemos generar casi cualquier punto pivote seleccionado (x_r, y_r) al realizar la siguiente secuencia de operaciones de traslación-rotación-traslación:

1. Traslade el objeto de modo que se mueva la posición del punto pivote al origen de las coordenadas.
2. Gire el objeto con respecto del origen de las coordenadas.

3. Traslade el objeto de manera que se regrese el punto pivote a su posición original.

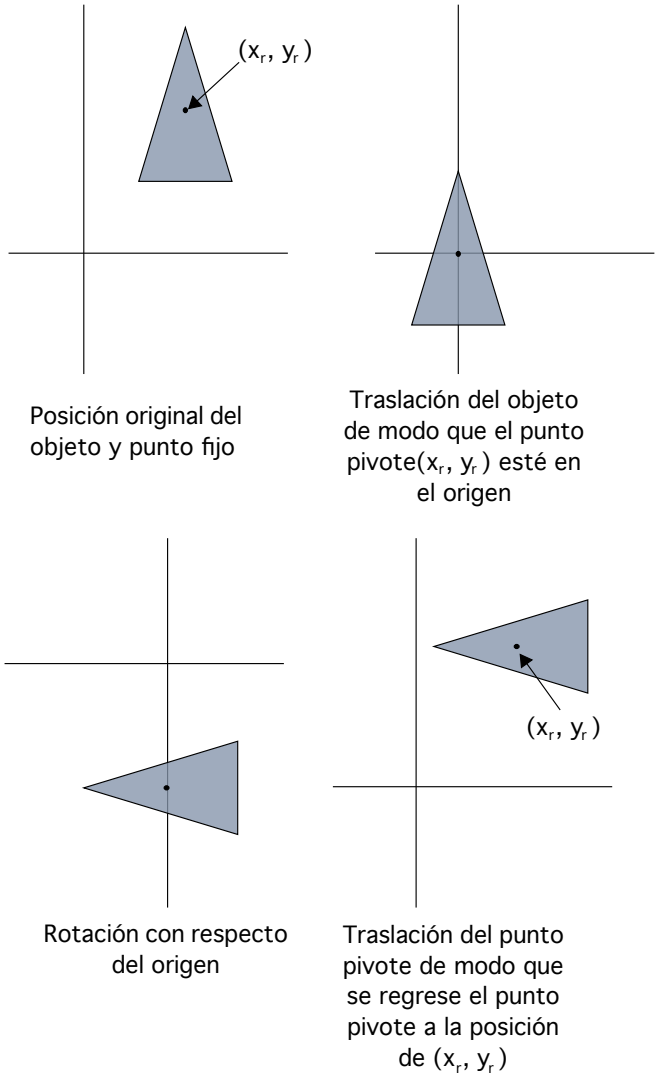


Fig. 2.7.Secuencia de transformación para girar un objeto con respecto de un punto pivote específico al utilizar la matriz R (θ) de la transformación 2.19

En la figura anterior se ilustra esta secuencia de transformación. La matriz de transformación compuesta para esta secuencia se obtiene de la concatenación

2.31. $$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

que se puede expresar en la siguiente forma

2.32. $$T(x_r, y_r)\cdot R(\theta)\cdot T(-x_r, -y_r) = R(x_r, y_r, \theta)$$

donde $T(-x_r, -y_r) = T^{-1}(x_r, y_r)$. En general, es posible determinar una función de rotación para aceptar parámetros para las coordenadas del punto pivote, así como el ángulo de rotación y generar en forma automática la matriz de rotación de la ecuación 2.31.

Escalación del punto fijo general

La siguiente figura ilustra una secuencia de transformación para producir escalación con respecto de una posición fija seleccionada (x_f, y_f) al utilizar una función de escalación que sólo puede escalar en relación con el origen de las coordenadas.

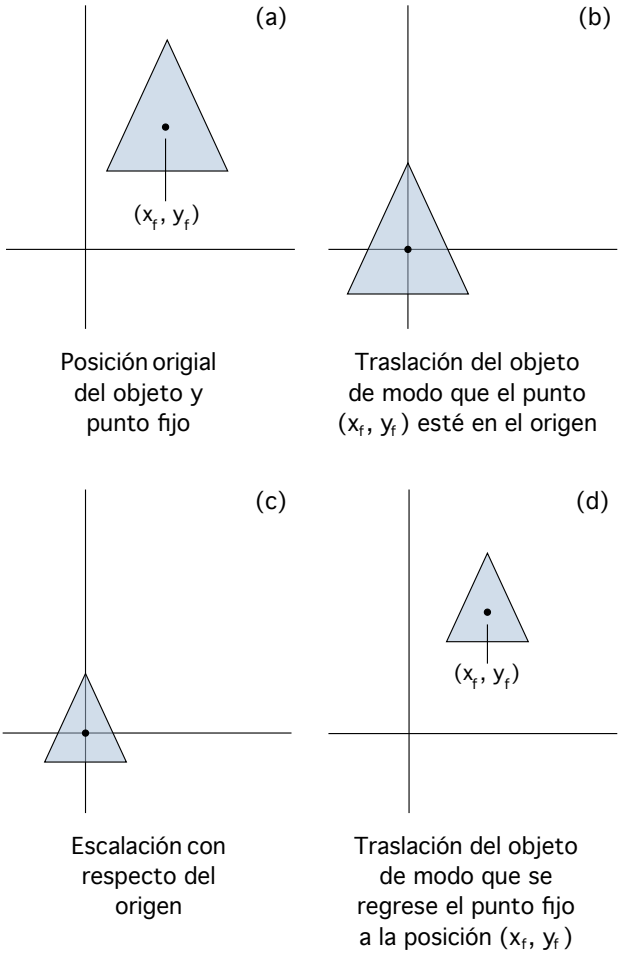


Fig. 2.8.Secuencia de transformación para escalar un objeto con respecto a la posición fija específica utilizando la matriz de escalación S(sx, sy) de la transformación 2.21

1. Traslade el objeto de modo que el punto fijo coincida con el origen de las coordenadas.
2. 2. Escale el objeto con respecto del origen de las coordenadas
3. Utilice la traslación inversa del paso 1 para regresar el objeto a su posición original.

La concatenación de las matrices para estas res operaciones produce la matriz de escalación requerida

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix} \quad 2.33.$$

o

$$T(x_f, y_f) \cdot S(s_x, s_y) \cdot T(-x_f, -y_f) = S(x_f, y_f, s_x, s_y) \quad 2.34.$$

Esta transformación se genera de manera automática en sistemas que ofrecen una función de escalación que acepta las coordenadas para el punto fijo.

Propiedades de concatenación

La multiplicación de matrices es asociativa. Para tres matrices cualesquiera A, B y C, el producto matricial A · B · C se puede lelvar a cabo al multiplicar primero a por B o multiplicar primero B por C:

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C) \quad 2.35.$$

Por tanto, podemos evaluar los productos matriciales al utilizar una agrupación asociativa ya sea de izquierda a derecha o de derecha a izquierda.

Por otro lado, los productos de la transformación tal vez no sean conmutativos. en general el producto matricial A · B no es igual que B · A. Esto significa queremos trasladar y girar un objeto, debemos tener cuidado sobre el sentido en que se evalúa la matriz compuesta

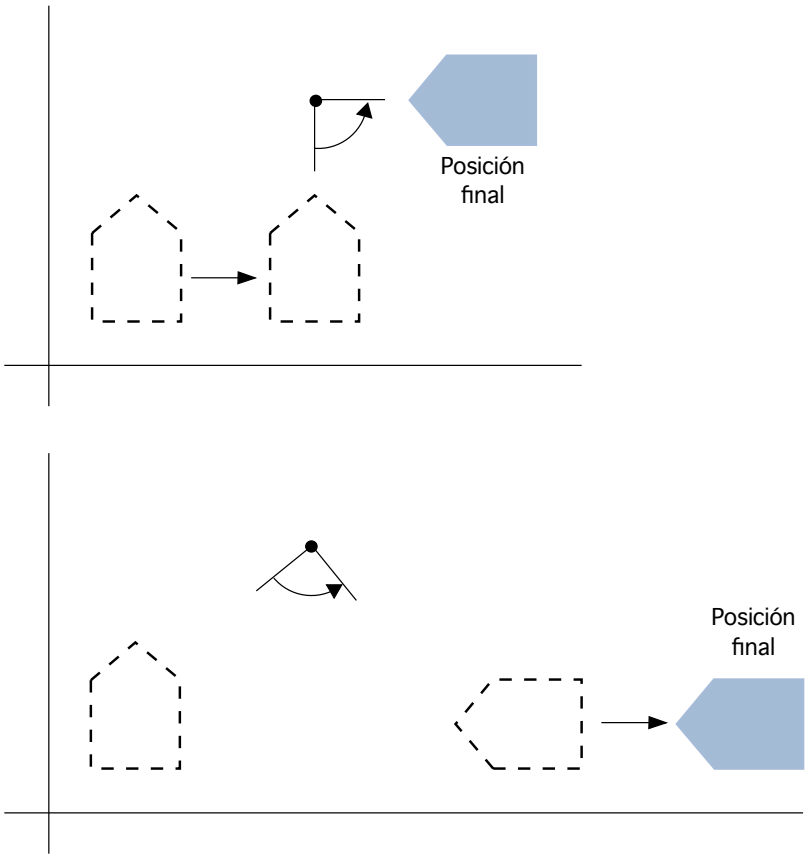


Fig 2.9.Invertir el orden en que se lleva a cabo una secuencia de transformaciones puede afectar la posición transformada de un objeto. En la parte (a), primero se traslada un objeto y luego se gira. En la parte (b), primero se gira el objeto y después se traslada.

Para algunos casos especiales, como una secuencia de transformaciones todas de la misma clase, la multiplicación de las matrices es conmutativa.

Como un ejemplo, se podrían realizar dos rotaciones sucesivas en cualquier sentido y la posición final sería la misma. Esta propiedad conmutativa también se aplica para dos traslaciones sucesivas o dos escalaciones sucesivas. Otro par conmutativo de operaciones es la rotación y la escalación uniforme. ($S_x = S_y$)

2.4.Transformación ventana-área de vista

Algunos paquetes gráficos permiten que el programador especifique coordenadas de primitivas de salida en un sistema de coordenadas de mundo de punto flotante, usando las

unidades que sean relevantes para el programa de aplicación: angstroms, micras, metros, millas, años luz, etcétera. Se emplea el término de mundo porque el programa de aplicación representa un mundo que se crea o presenta interactivamente para el usuario:

Como las primitivas de salida se expresan en coordenadas de mundo, hay que indicar al paquete de subrutinas gráficas cómo establecer la correspondencia entre las coordenadas de mundo y las coordenadas de pantalla (usaremos el término específico coordenadas de pantalla para relacionar este análisis específicamente con SRGP, pero podrían usarse dispositivos de impresión, en cuyo caso sería más apropiado el término coordenadas de dispositivo).

Esta correspondencia se puede efectuar si el programador de la aplicación proporciona al paquete gráfico una matriz de transformación para la correspondencia. Otra forma es que el programador de la aplicación especifique una región rectangular en coordenadas de mundo, llamada ventana de coordenadas mundiales y una región rectangular correspondiente en coordenadas de pantalla, llamada área de vista, con la cual se establece la correspondencia de la ventana de coordenadas mundiales. La transformación que establece la correspondencia entre la-ventana y el área de vista se aplica a todas las primitivas de salida en coordenadas de mundo para que correspondan a coordenadas de pantalla.

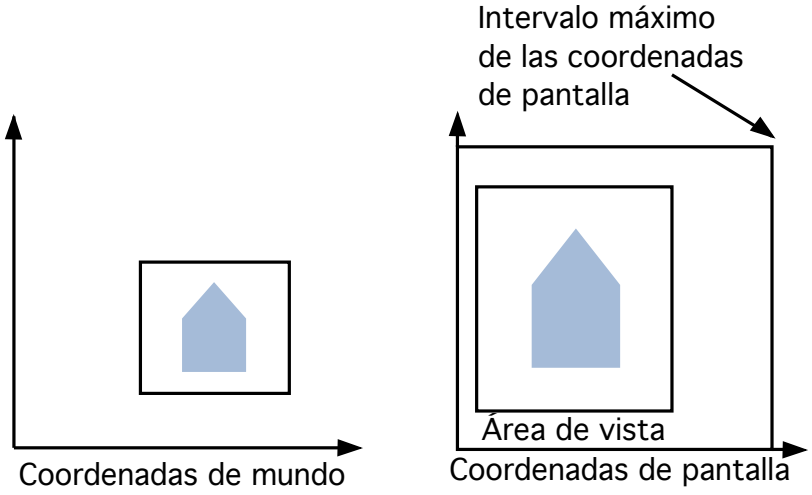


Fig. 2.10.La ventana en coordenadas de mundo y el área de vista en coordenadas de pantalla determinan la correspondencia que se aplica a todas las primitivas de salida en coordenadas de mundo.

Como se puede ver, si la ventana y el área de vista no tienen la misma razón altura-anchura, ocurre un escalamiento no uniforme. Si el programa de aplicación cambia la ventana o el área de vista, las nuevas primitivas de salida que se dibujen en la pantalla se verán afectadas por el cambio, no así las primitivas existentes.

El modificador coordenadas de mundo se emplea con ventana para subrayar que no se trata de una ventana de administrador de ventanas, un concepto distinto y más reciente que por desgracia tiene el mismo nombre. Este modificador se omitirá cuando no exista ninguna ambigüedad con respecto al tipo de ventana que se trate.

Si SRGP proporcionara primitivas de salida en coordenadas de mundo, el área de vista se hallaría en el lienzo actual, que por omisión es el lienzo 0, la pantalla. El programa de aplicación podría cambiar en cualquier instante la ventana o el área de vista, en cuyo caso las primitivas de salida que se especificaran subsecuentemente estarían sujetas a una nueva transformación. Si el cambio incluyera un área de vista distinta, las nuevas primitivas de salida se colocarían en el lienzo en posiciones distintas a las anteriores.

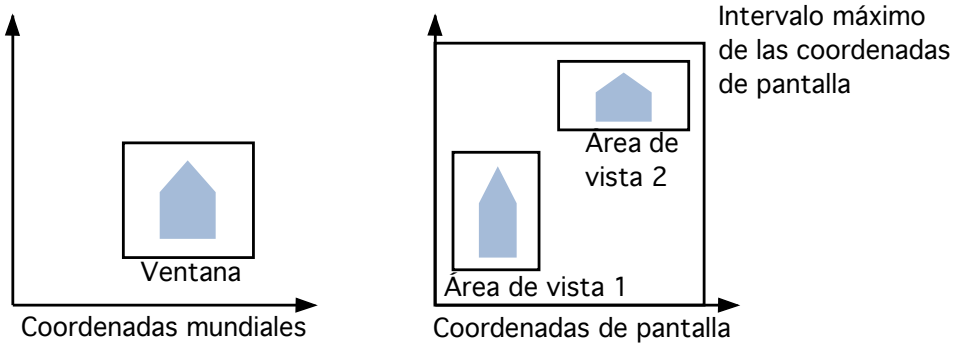


Fig. 2.11.Efecto de dibujar primitivas de salida con dos áreas de vista. Las primitivas de salida que especifican la casa se dibujaron primero con el área de vista 1, que luego se cambió al área de vista 2., para lo cual el programa de aplicación invocó nuevamente al paquete gráfico para dibujar las primitivas de salida.

Un administrador de ventanas podría establecer la correspondencia entre el lienzo 0 de SRGP y una ventana menor que la pantalla completa; en este caso no siempre estará visible todo el lienzo ni toda el área de vista.

Si tenemos una ventana y un área de vista, ¿cuál es la matriz de transformación que establece la correspondencia entre las coordenadas mundiales de la ventana y las

coordenadas de pantalla del área de vista? Esta matriz se puede desarrollar corno una composición de transformación de tres pasos, como se sugiere en la figura 2.12

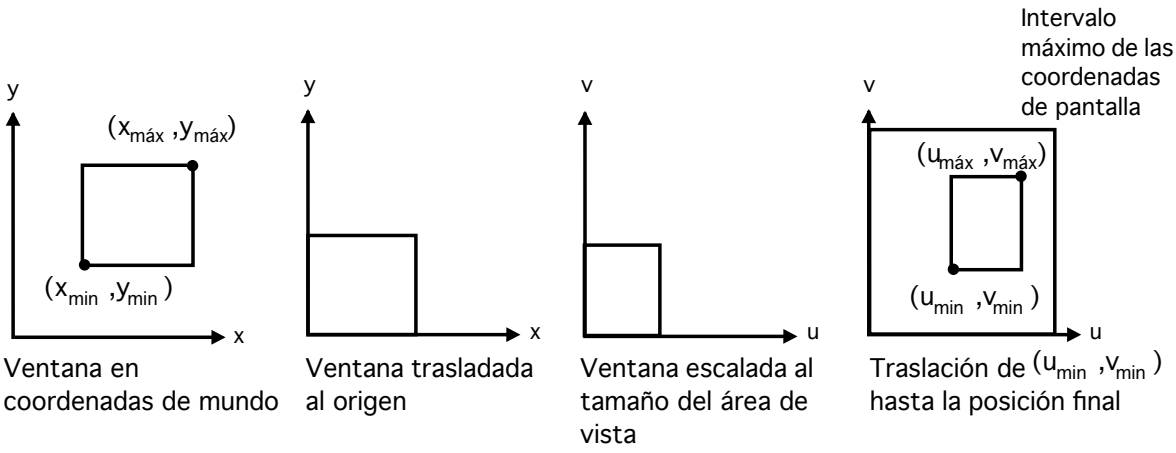


Fig. 2.12.Pasos utilizados en la transformación de una ventana de coordenadas mundiales a un área de vista.

La ventana, especificada por sus vértices inferior izquierdo y superior derecho, se traslada primero al origen de las coordenadas de mundo. Después se escala el tamaño de la ventana para que sea igual al tamaño del área de vista. Por último se usa una traslación para colocar un área de vista. La matriz global M_{wv}, que se obtiene por medio de la composición de dos matrices de traslación y la matriz de escalamiento, es

$$M_{wv} = T(u_{mín}, v_{mín}) \bullet S\left(\frac{u_{máx} - u_{mín}}{x_{máx} - x_{mín}}, \frac{v_{máx} - v_{mín}}{y_{máx} - y_{mín}}\right) \bullet T(-x_{mín} - y_{mín})$$

$$= \begin{bmatrix} 1 & 0 & u_{mín} \\ 0 & 1 & v_{mín} \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \frac{u_{máx} - u_{mín}}{x_{máx} - x_{mín}} & 0 & 0 \\ 0 & \frac{v_{máx} - v_{mín}}{y_{máx} - y_{mín}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -x_{mín} \\ 0 & 1 & -y_{mín} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{u_{máx} - u_{mín}}{x_{máx} - x_{mín}} & 0 & -x + \frac{u_{máx} - u_{mín}}{x_{máx} - x_{mín}} \\ 0 & \frac{v_{máx} - v_{mín}}{y_{máx} - y_{mín}} & -y + \frac{v_{máx} - v_{mín}}{y_{máx} - y_{mín}} \\ 0 & 0 & 1 \end{bmatrix}$$

2.36.

Al multiplicar $P = M_{wv}[x \text{ y } 1]^T$ se obtiene el resultado esperado:

$$P = \begin{bmatrix} (x - x_{mín}) \bullet \frac{u_{máx} - u_{mín}}{x_{máx} - x_{mín}} + u_{mín} & (y - y_{mín}) \bullet \frac{v_{máx} - v_{mín}}{y_{máx} - y_{mín}} + v_{mín} & 1 \end{bmatrix}$$

2.37.

Muchos paquetes gráficos combinan la transformación ventana-área de vista con el recorte de primitivas de salida con respecto a la ventana. La siguiente figura ilustra el recorte en el contesto de ventanas y áreas de vista.

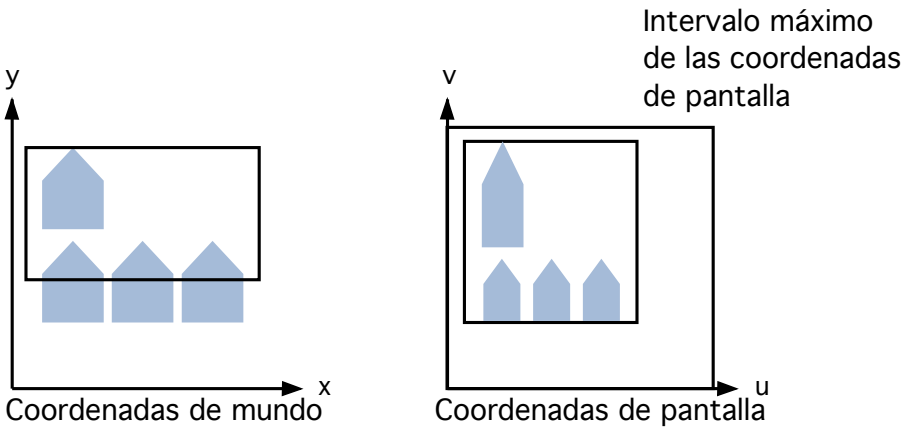


Fig. 2.13.Las primitivas de salida en coordenadas mundiales se recortan con respecto a la ventana. Las que permanecen se presentan en el área de vista.

2.5.Transformaciones de composición general y de eficiencia computacional

Una transformación bidimensional general, que representa una combinación de traslaciones, rotaciones y escalaciones se puede expresar como

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} rs_{xx} & rs_{xy} & trs_x \\ rs_{yx} & rs_{yy} & trs_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2.38.)$$

Los cuatro elementos rs_{ij} son los términos multiplicativos de rotación -escalación en la transformación que implican sólo ángulos de rotación y factores de escalación. Los elementos trs_x y trs_y son los términos de traslación que contienen combinaciones de distancias de traslación, coordenadas de punto pivote y de punto fijo, así como de ángulos de rotación y parámetros de escalación.

Por ejemplo, si debe escalar, girar un objeto con respecto de las coordenadas de su centroide (x_c, y_c) y después trasladarlo, los valores para los elementos de la matriz de transformación compuesta son

$$T(t_x,t_y)\cdot R(x_c,y_c,\theta)\cdot S(x_c,y_c,s_x,s_y)=\begin{pmatrix} s_x\cos\theta & -s_x\sin\theta & x_c(1-s_x\cos\theta)+y_c s_y\sin\theta)+t_x \\ s_x\sin\theta & -s_y\cos\theta & y_c(1-s_y\cos\theta)+x_c s_x\sin\theta)+t_y \\ 0 & 0 & 1 \end{pmatrix} \quad \text{2.39.}$$

Aunque la ecuación de matriz 2.35 requiere nueve multiplicaciones y seis adiciones, los cálculos explícitos para las coordenadas transformadas son

$$\begin{aligned} x' &= x\cdot rs_{xx} + y\cdot rs_{xy} + trs_x, \\ y' &= x\cdot rs_{yx} + y\cdot rs_{yy} + trs_y \end{aligned} \quad \text{2.40.}$$

Así, en realidad solo necesitamos efectuar cuatro multiplicaciones y cuatro adiciones para transformar las posiciones de las coordenadas. Este es el número máximo de cálculos que se requieren para cualquier secuencia de transformación, una vez que se han concatenado las matrices individuales y evaluado los elementos de la matriz compuesta.

Sin concatenación, se aplicarían las transformaciones individuales una a la vez y se podría reducir en forma considerable el número de cálculos. De esta manera, una implementación eficiente de las operaciones de transformación consiste en formular matrices de transformación, concatenar cualquier secuencia de transformación y calcular las coordenadas transformadas al utilizar la ecuación 2.40.

En sistemas paralelos las multiplicaciones matriciales directas con la matriz de transformación compuesta de la ecuación 2.38 pueden ser igual de eficientes.

Se puede expresar una matiz de transformación de cuerpo rígido general que sólo implica traslaciones y rotaciones en la forma

$$\begin{bmatrix} r_{xx} & r_{xy} & tr_x \\ r_{yx} & r_{yy} & tr_y \\ 0 & 0 & 1 \end{bmatrix} \quad \text{2.41.}$$

donde los cuatro elementos r_{ij} son los términos de traslación. En ocasiones. un cambio de cuerpo rígido en la posición de coordenadas se denomina transformación de movimiento rígido.

Todos los ángulos y distancias entre las posiciones de coordenadas permanecen sin cambio al realizar la transformación. Además, la matriz 2.41 tiene la propiedad de que su submatriz superior izquierda de 2 por 2 está en una matriz ortogonal. Esto significa que si consideramos cada renglón de la submatriz como un vector, entonces los dos vectores (r_{xx}, r_{xy}) y (r_{yx}, r_{yy}) forman un conjunto ortogonal de vectores unitarios: cada vector tiene longitud unitaria.

2.42. $$r_{xx}^2 + r_{xy}^2 = r_{yx}^2 + r_{yy}^2 = 1$$

y los vectores son perpendiculares (su producto de punto es 0):

2.43. $$r_{xx}r_{yx} + r_{xy}r_{yy} = 0$$

Por tanto, si estos vectores unitarios se transforman por la submatriz de rotación, (r_{xx}, r_{xy}) se convierte en un vector unitario a lo largo del eje *x*, y (r_{xx}, r_{xy}) se transforma en un vector unitario a lo largo del eje y del sistema de coordenadas:

2.44. $$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{xx} \\ r_{xy} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

2.45. $$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Como un ejemplo, la siguiente transformación de cuerpo rígido primero gira un objeto a través de un ángulo θ con respecto del punto pivote (x_r, y_r) y luego lo traslada:

2.46. $$T(t_x, t_y) \cdot R(x_r, y_r, \theta) = \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta) + t_x \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r \sin\theta) + t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Aquí, los vectores unitarios ortogonales en la submatriz superior izquierda de 2 por 2 son (*cos* θ, *–sin* θ) y (*sin* θ, *cos* θ), y

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta \\ -\sin\theta \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

2.47.

De modo similar, el vector unitario (*sin* θ, *cos* θ) se convierte por la matriz de transformación de la ecuación anterior en el vector unitario en la dirección de y.

La propiedad ortogonal de las matrices de rotación es útil para realizar una matriz de rotación cuando conocemos la orientación final de un objeto en lugar de la cantidad de rotación angular necesaria para colocar el objeto en esa posición. Las direcciones para la orientación deseada de un objeto se podrían determinar mediante la alineación de ciertos objetos en escena o mediante posiciones seleccionadas en la escena.

La figura 2.14 ilustra un objeto que se debe alinear con los vectores de unidad *u'* y *v'*. Al suponer que la orientación original del objeto como se muestra en el inciso a) de la misma figura, está alineada con los ejes de las coordenadas, realizamos la transformación deseada al asignar los elementos de *u'* al primer renglón de la matriz de rotación y los elementos de *v'* al segundo renglón.

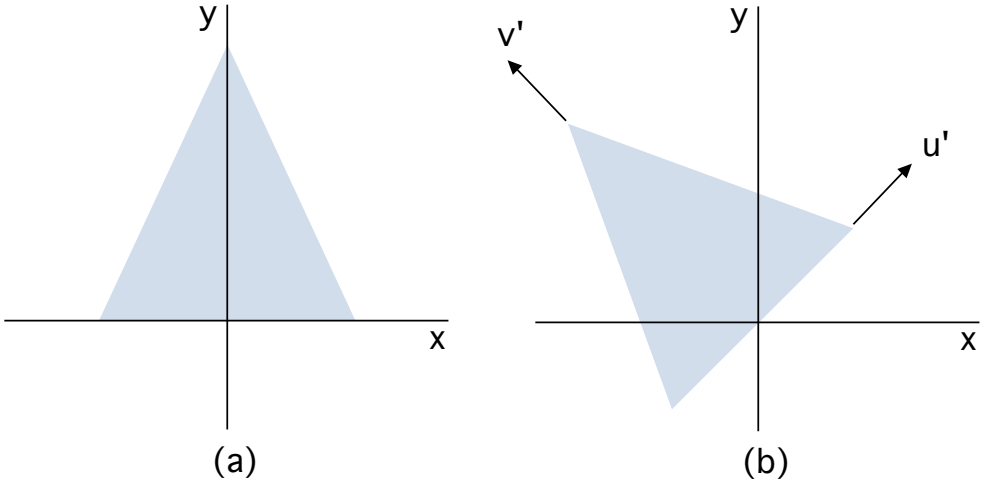


Fig. 2.14.La matriz de rotación para girar un objeto de la posición (a) a la posición (b)se puede crear con los valores de los vectores de orientación unitaria u' y v' con respecto de la orientación original.

Este puede ser un método conveniente para obtener la matriz de transformación para la rotación en un sistema de coordenadas local (u "objeto") cuando conocemos los vectores de orientación finales.

Dado que los cálculos de la rotación requieren evaluaciones trigonométricas y varias multiplicaciones para cada punto que se transforma, la eficiencia computacional puede tornarse una consideración importante en las transformaciones de rotación.

En animaciones y otras aplicaciones que implican muchas transformaciones repetidas y ángulos de rotación pequeños, podemos utilizar aproximaciones y cálculos repetitivos para reducir los cálculos en las ecuaciones de transformación compuesta.

Cuando el ángulo de rotación es pequeño, se pueden sustituir las funciones trigonométricas con valores de aproximación con base en los primeros términos de sus expansiones de series de potencia. Para ángulos bastante pequeños (menores a 10°) *cos* θ es de alrededor de *1 sen* θ tiene un valor muy cercano al valor de θ en radianes.

Por ejemplo, si giramos en pasos angulares pequeños alrededor del origen, podemos establecer *cos* θ como 1 y reducir los cálculos de la transformación en cada paso a dos multiplicaciones y dos adiciones para cada conjunto de coordenadas que se deben girar:

$$x' = x - y\sin\theta,$$
$$y' = x\sin\theta + y \qquad (2.48)$$

donde se evalúa *sin* θ una vez para todos los pasos, si se supone que el ángulo de rotación no cambia.

El error que implica esta aproximación en cada paso disminuye conforme el ángulo de rotación es menor. Pero incluso con ángulos de rotación pequeños, el error acumulado sobre muchos pasos puede tornarse muy grande.

Podemos controlar el error acumulado al estimar el error en *x'* y *y'* en cada paso y volver a especificar las posiciones del objeto cuando la acumulación del error es muy grande.

Las transformaciones compuestas a menudo implican cálculos de la matriz inversa. Por ejemplo, las secuencias de transformación para las direcciones de escalación general y para reflexiones y recortes se pueden describir con componentes de rotación inversa.

Se pueden generar con procedimientos simples las representaciones de matriz inversa para las transformaciones geométricas básicas. Se obtiene una matriz de traslación inversa

al realizar un transposición de matriz (o cambiar el signo de los términos del seno). Estas operaciones son mucho más sencillas que los cálculos de la matriz inversa.

En el siguiente ejemplo, se escala y gira un polígono con respecto de un punto de referencia determinado y luego se traslada el objeto. En la figura se ilustran las posiciones original y final del polígono que se transforma mediante esta secuencia.

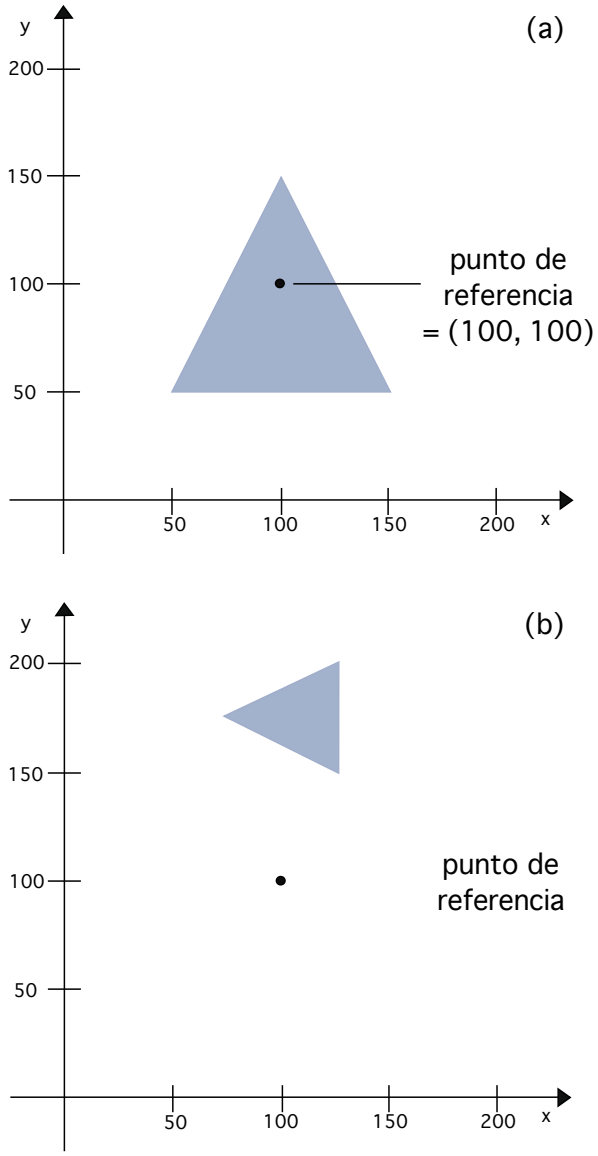


Fig. 2.15.Se transformó el polígono (a) en el (b) mediante operaciones compuestas

2.6.Representación matricial de transformaciones tridimensionales

Así como las transformaciones bidimensionales se pueden representar con matrices de 3 X 3 usando coordenadas homogéneas, las transformaciones tridimensionales se pueden representar con matrices de 4 X 4, siempre y cuando usemos representaciones de coordenadas homogéneas de los puntos en el espacio tridimensional. Así, en lugar de representar un punto como (*x, y, z*), lo hacemos como (*x, y, z, W*), donde dos de estos cuádruplos representan el mismo punto si uno es un multiplicador distinto de cero del otro: no se permite el cuádruplo (0, 0, 0, 0). Como sucede en el espacio bidimensional, la representación estándar de un punto (*x, y, z, W*) con $W \neq 0$ se indica (*x/W, y/W, z/W, 1*).

La transformación de un punto a esta forma se denomina homogeneización, igual que antes. Además los puntos cuya coordenada W es cero se llaman puntos en el infinito. También existe una interpretación geométrica. Cada punto en el espacio tridimensional se representa con una línea que pasa por el origen en el espacio de cuatro dimensiones, y las representaciones homogeneizadas de estos puntos forman un subespacio tridimensional de un espacio de cuatro dimensiones definido por la ecuación $W = 1$.

El sistema de coordenadas tridimensionales que se usará en los siguientes ejemplos es de mano derecha como se ilustra en la figura 2.16- Por convención las rotaciones positivas en el sistema de mano derecha son tales que, al ver hacia un eje positivo desde el origen, una rotación de 90˚ en sentido contrario al giro de las manecillas del reloj transformará un eje positivo en otro. La tabla siguiente se desprende de esta convención

EJE DE ROTACIÓN	DIRECCIÓN DE LA ROTACIÓN POSITIVA
x	y a z
y	z a x
z	x a y

Estas direcciones positivas se ilustran en la figura 2.16.

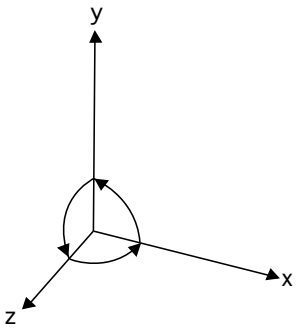


Fig. 2.16.Sistema de coordenadas de mano derecha

Se usa el sistema de mano derecha porque se trata de una convención matemática estándar, aunque en la graficación tridimensional es conveniente pensar en un sistema de mano izquierda sobrepuesto en la pantalla (fig. 2.17), ya que un sistema de mano izquierda da la interpretación natural de que los valores mayores de *z* se encuentran más lejos del observador. Observe que en un sistema de mano izquierda, las rotaciones positivas son en el sentido del giro de las manecillas del reloj cuando se observa desde un eje positivo hacia el origen.

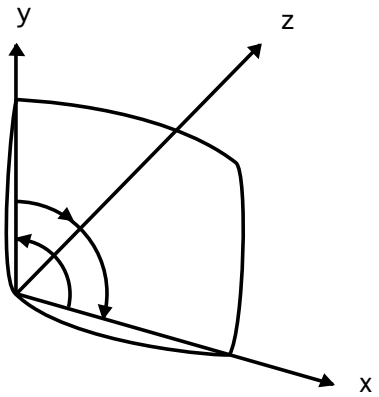


Fig. 2.17.Sistema de coordenadas de mano izquierda, con una pantalla superpuesta

La traslación en el espacio tridimensional es una simple extensión de la que se lleva a cabo en el espacio bidimensional:

$$T(d_z,d_y,d_z)=\begin{bmatrix}1 & 0 & 0 & d_z\\ 0 & 1 & 0 & d_y\\ 0 & 0 & 1 & d_z\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad 2.49.$$

Es decir $T(d_x, d_y, d_z)\cdot[x\ y\ z\ 1]^T = [x+ d_x y + d_y z + d\ 1]^T$

El escalamiento se extiende en forma similar:

$$S(s_x, s_y, d_z) = \begin{bmatrix} s_z & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.50.

Al revisar se observa que $S(s_x, s_y, s_z) \cdot [x\ y\ z\ 1]^T = [s_x \cdot x s_y \cdot y s_z \cdot z\ 1]^T$.

La rotación bidimensional de la ecuación 2.8 es simplemente una rotación tridimensional con respecto al eje z, que es

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

2.51.

Es fácil verificar esta observación: una rotación de 90° de [1 0 0 1] que es el vector unidad sobre el eje *x*, debe producir el vector unidad [0 1 0 1] sobre el eje *y*. Al evaluar el producto

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

2.52.

se obtiene el resultado previsto de $[0\ 1\ 0\ 1]^T$.

La matriz de rotación del eje *x* es

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

2.53.

La matriz de rotación del eje *y* es

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

2.54.

Las columnas (y las filas) de la submatriz superior izquierda de 3 X 3 de $R_z(\theta)$, $R_x(\theta)$ y $R_y(\theta)$ son vectores unidad mutuamente perpendiculares y el determinante de la submatriz es 1, lo que quiere decir que las matrices son ortogonales especiales. Además, la submatriz superior izquierda de 3 X 3 formada por una secuencia arbitraria de rotación es ortogonal especial. Recuerde que las transformaciones ortogonales conservan las distancias y los ángulos.

Todas estas matrices de transformación tienen inversas. La inversa de *T* se obtiene cambiando de signo dc d_x, d_y y d_z; la de *S* reemplazando s_x, s_y y s_z por sus recíprocos; y la inversa de cada una de las tres matrices de rotación, negando el ángulo de rotación.

La inversa de una matriz ortogonal *B* es la transpuesta de *B:* $B^{-1} = B^T$. De hecho, para tomar la transpuesta no es necesario intercambiar los elementos del arreglo que contiene la matriz: basta intercambiar los índices de las filas y las columnas al acceder a la matriz. Observe que este método para hallar la inversa es consistente con el resultado del cambio de signo de θ para hallar la inversa de R_x, R_y y R_z.

Es posible multiplicar juntas cualquier cantidad de matrices de rotación, escalación y traslación. El resultado siempre tiene la forma

$$M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.55.

Como en el caso bidimensional, la submatriz superior izquierda de 3 X 3R nos da la rotación y el escalamiento combinados, mientras que T nos da la traslación subsecuente. Podemos obtener mayor eficiencia computacional si efectuamos la transformación explícita, como

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \bullet \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T, \tag{2.56}$$

donde R y T son submatrices de la ecuación 2.55

Existen submatrices de sesgo tridimensional correspondientes a las matrices de sesgo bidimensional. El sesgo (x, y) es

$$SH_{xy}(sh_x, sh_y) = \begin{bmatrix} 1 & 0 & sh_z & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.57}$$

Al aplicar SH_{xy} al punto $[x\ y\ z\ 1]^T$ se obtiene $[x + sh_x \cdot z \quad y + sh_y \cdot z\ 1]^T$. Los sesgos sobre los ejes x y y tienen forma similar.

Hasta ahora se centró la atención en la transformación de puntos individuales. La transformación de líneas, definidas éstas por puntos, se obtiene transformando los puntos extremos. Los planos si están definidos por tres puntos se pueden manejar de la misma forma, pero por lo general se definen con una ecuación de plano y los coeficientes de esta ecuación deben transformarse de forma distinta. También puede ser necesario transformar la normal al plano.

Representemos un plano como el vector columna de los coeficientes de la ecuación del plano $N = [A\ B\ C\ D]^T$. Entonces, un plano está definido por los puntos P tales que $N \cdot P = 0$, donde el símbolo "$\cdot$" es el producto punto vectorial y $P = [x\ y\ z\ 1]^T$. Este producto punto da lugar a la conocida ecuación del plano $Ax + By + Cy + D = 0$, que también se puede expresar como el producto del vector fila de los coeficientes de la ecuación del plano multiplicado por la columna P: $N^T \cdot P = 0$. Suponga ahora que transformamos todos los puntos P en el plano con una matriz M. Para mantener $N^T \cdot P = 0$ para todos los puntos transformados, quisiéramos transformar N por una matriz Q (por determinarse) que dé lugar a la ecuación $(Q \cdot N)^T \cdot M \cdot P = 0$ usando la identidad $(Q \cdot N)^T = N^T \cdot \cdot Q^T$. La ecuación será verdadera si $Q^T \cdot M$ es múltiplo de la matriz identidad. Si el multiplicador es 1, esta situación nos lleva a $Q^T = M^{-1}$ o

$Q=(M^{-1})^T$. Por lo tanto el vector columna N de coeficientes de un plano transformado por M se expresa como

2.58. $$N' = (M^{-1})^T \bullet N$$

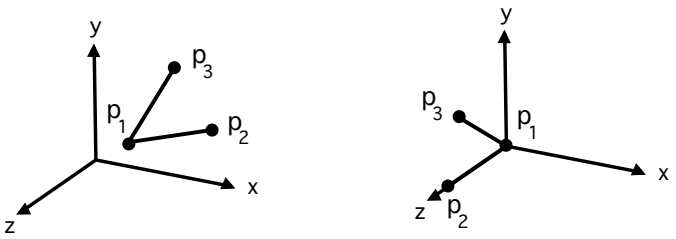
Por lo general no es necesario que exista la matriz $(M^{-1})^T$, ya que el determinante M puede ser cero. Esta situación ocurriría si M incluye una proyección (quizá querríamos investigar el efecto de una proyección de perspectiva sobre el plano).

Si sólo se transformará la normal al plano (por ejemplo, para efectuar algunos cálculos de sombreado) y si M sólo consiste en una composición de matrices de traslación, rotación y escalamiento uniforme, entonces se simplificarán las matemáticas. La N de la ecuación anterior se puede simplificar a $[A'\ B'\ C'\ 0]^T$. (Con un componente W igual a cero, un punto homogéneo representa un punto en el infinito, lo cual se puede considerar como una dirección).

2.7.Composición de transformaciones tridimensionales

En este apartado se analizará la forma de componer matrices de transformación tridimensionales usando un ejemplo. El objetivo es transformar los segmentos de línea dirigida P_1P_2 y P_1P_3 en la figura 2.18 de su posición inicial en la parte (a) a su posición final en la parte (b). De esta manera, el punto P_1 se trasladará al origen P_1P_2 quedará en el eje positivo y P_1P_3 quedará en la mitad del eje positivo del plano (x, y). Las longitudes de las líneas no se verán afectadas por la transformación.

Se presentan dos formas de lograr la transformación deseada. El primer método es componer las transformaciones primitivas T, R_x, R_y y R_z. Este método, aunque es algo tedioso, es fácil de ilustrar y su comprensión nos ayudará en nuestro conocimiento de las transformaciones. El segundo método, que utiliza las propiedades de las matrices ortogonales especiales que se analiza en la sección anterior, se explica de manera mas breve pero es más abstracto.



(a) Posición inicial (b) Posición final

Fig. 2.18.Transformación de P_1, P_2 y P_3 de su posición inicial (a) a su posición final (b)

Para trabajar con las transformaciones primitivas, de nuevo dividimos un problema difícil en varios más sencillos. En este caso, la transformación deseada se puede llevar a cabo en cuatro pasos:

1. Traslación de *P1* al origen.
2. Rotación sobre el eje y para que *P1P2* esté en el plano (*y*, *z*).
3. Rotación sobre el eje x para que *P1P2* esté en el eje *z*.
4. Rotación sobre el eje z para que *P1P3* esté en el plano (*y, z*).

Paso 1: Traslación *P1* al origen. La traslación es

$$T(-x_1,-y_1,-z_1)=\begin{bmatrix}1 & 0 & 0 & -x_1\\ 0 & 1 & 0 & -y_1\\ 0 & 0 & 1 & -z_1\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (2.59)$$

Al aplicar *T* a *P1*, *P2* y *P3* se obtiene

$$P'_1 = T(-x_1,-y_1,-z_1)\bullet P_1 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \quad (2.60)$$

$$P'_2 = T(-x_1,-y_1,-z_1)\bullet P_2 = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \\ 1 \end{bmatrix}$$

2.61.

$$P'_3 = T(-x_1,-y_1,-z_1)\bullet P_3 = \begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \\ 1 \end{bmatrix}$$

2.62.

Paso 2: Rotación sobre el eje y. En la figura 2.19 se muestra P_1P_2 después del paso 1, así como la proyección de P_1P_2 sobre el plano (x, z). El ángulo de rotación es –(90 - θ) = θ -90.

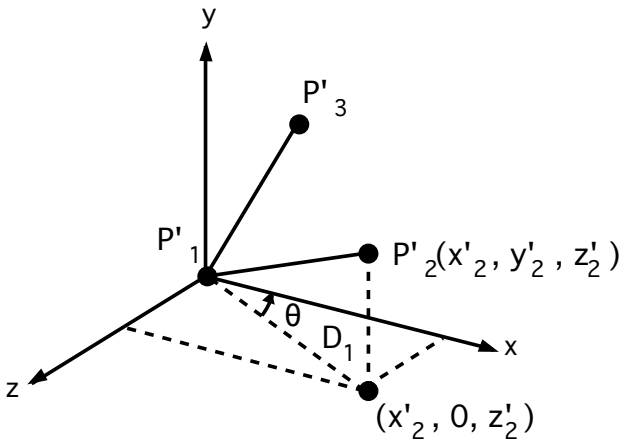


Fig. 2.19.Rotación sobre el eje y. La proyección P1'P2', cuya longitud es D1, se hace rotar al eje z. El ángulo muestra la dirección positiva de la rotación sobre el eje y. El ángulo que se usa en realidad es -(-90 - θ).

Entonces,

$$\cos(\theta - 90) = \sin\theta = \frac{z'_2}{D_1} = \frac{z_2 - z_1}{D_1},$$
$$\sin(\theta - 90) = -\cos\theta = \frac{x'_2}{D_1} = -\frac{x_2 - x_1}{D_1}$$

2.63.

Donde

2.64. $$D_1 = \sqrt{(z'_2)^2 + (x'_2)^2} = \sqrt{(z_2 - z_1)^2 + (x_2 - x_1)^2}$$

Al sustituir estos valores en la ecuación 2.63 se obtiene

2.65. $$P_2'' = R_y(\theta - 90) \bullet P_2' = \left[0\, y_2 - y_1\, D_1\, 1\right]^T$$

Como era de esperar, el componente *x* de P_2'' es cero y el componente *z* es la longitud D_1.

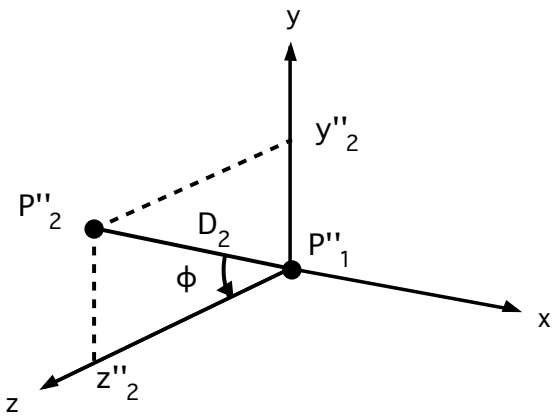


Fig. 2.20.Rotación sobre el eje x: P1''P2'' rota en el eje z con el ángulo positivo ϕ. D2 es la longitud del segmento de línea. No se presenta el segmento de la línea P1''P2'', ya que no se usa para determinar los ángulos de rotación. Las dos líneas se hacen rotar con Rx(ϕ)

Paso 3: Rotación sobre el eje *x*. En la figura 2.20 se muestra P_1P_2 después del paso 2. El ángulo de rotación es φ, para el cual

2.66. $$\cos\phi = \frac{z''_2}{D'_2}, \sin\phi = \frac{y''_2}{D'_2}$$

Donde $D_2 = |P_1''P_2''|$, la longitud de la línea $P_1''P_2''$. Sin embargo, la longitud de la línea $P_1''P_2''$ es igual a la longitud de la línea P_1P_2, ya que las transformaciones de rotación y traslación conservan la longitud; por lo tanto,

2.67. $$D_2 = \left|P_1'' P_2''\right| = \left|P_1 P_2\right| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

El resultado de la rotación en el paso 3 es

$$P_2''' = R_x(\phi)\bullet P_2'' = R_x(\phi)\bullet R_y(\theta-90)\bullet P_2'$$
$$= R_x(\phi)\bullet R_y(\theta-90)\bullet T\bullet P_2 = \begin{bmatrix} 0 & 0 & |P_1P_2| & 1 \end{bmatrix}^T \quad (2.68.)$$

Es decir, P_1P_2 coincide ahora con el eje z positivo.

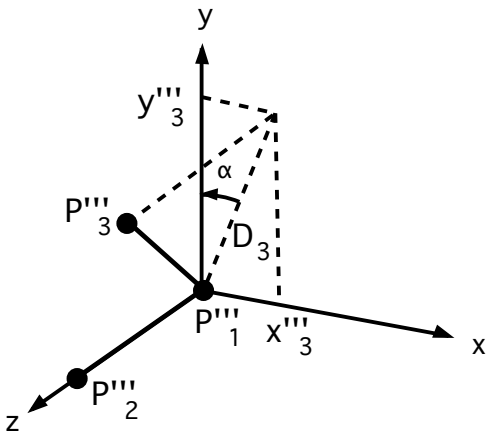


Fig. 2.21.Rotación sobre el eje z. El ángulo positivo hace rotar la proyección de P1'P3', cuya longitud es D3, en el eje y, con lo cual la línea llega al plano (y, z). D3 es la longitud de la proyección.

Paso 4. Rotación sobre el eje z. En la figura 2.21 se muestran P_1P_2 y P_1P_3 después del paso 3, con P_2''' en el eje z y P_3''' en la posición

$$P_3''' = \begin{bmatrix} x_3''' y_3''' z_3''' 1 \end{bmatrix}^T = R_x(\phi)\bullet R_y(\theta-90)\bullet T(-x_1,-y_1,-z_1)\bullet P_3 \quad (2.69.)$$

La rotación a través del ángulo positivo α, con

$$\cos\alpha = y_3'''/D_3,$$
$$\sin\alpha = x_3'''/D_3,$$
$$D_3 = \sqrt{x_3'''^2,+y_3'''^2} \quad (2.70.)$$

Con el paso 4 se obtiene el resultado que se presenta en la figura 2.18(b).

La matriz compuesta

2.71.
$$R_z(\alpha)\bullet R_x(\phi)\bullet R_y(\theta-90)\bullet T(-x_1,-y_1,-z_1)=R\bullet T$$

es la transformación requerida, con $R = Rz(\alpha)\cdot Rx(\varphi)\cdot Ry(\theta - 90)$. Dejaremos que usted aplique esta transformación a P_1, P_2 y P_3 para verificar que P_1 se transforme al origen, P_2 se transforme al eje z positivo y que P_3 se transforma a la mitad y positiva del plano (y, z).

La segunda manera de obtener la matriz R es usar las propiedades de las matrices ortogonales. Recuerde que los vectores fila unidad de R rotan hacia los ejes principales. Al reemplazar los segundos subíndices de la ecuación 2.55 con x, y y z para que la rotación sea más conveniente,

2.72.
$$R=\begin{bmatrix} r_{1_x} & r_{2_x} & r_{3_x} \\ r_{1_y} & r_{2_y} & r_{3_y} \\ r_{1_z} & r_{2_z} & r_{3_z} \end{bmatrix}$$

Como R_z es el vector unidad sobre P_1P_3 que rotará hacia el eje positivo,

2.73.
$$R_z=\begin{bmatrix} r_{1_x} & r_{2_x} & r_{3_x} \end{bmatrix}^T=\frac{P_1P_2}{|P_1P_2|}$$

Además, el vector unidad R, es perpendicular al plano de P_1, P_2 y P_3 y rotará hacia el eje x positivo, de manera que R, debe ser el producto cruz normalizado de dos vectores en el plano:

2.74.
$$R_x=\begin{bmatrix} r_{1_x} & r_{2_x} & r_{3_x} \end{bmatrix}^T=\frac{P_1P_3\times P_1P_2}{|P_1P_3\times P_1P_2|}$$

Finalmente,

2.75.
$$R_y=\begin{bmatrix} r_{1_x} & r_{2_x} & r_{3_x} \end{bmatrix}^T=R_z\times R_x$$

rotará hacia ele eje y positivo. La matriz compuesta se expresa como

$$\begin{bmatrix} r_{1_x} & r_{2_x} & r_{3_x} & 0 \\ r_{1_y} & r_{2_y} & r_{3_y} & 0 \\ r_{1_z} & r_{2_z} & r_{3_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet T(-x_1, -y_1, -z_1) = R \bullet T$$

2.76.

Donde R y T son como en la ecuación 2.71. En la figura siguiente se muestran los vectores individuales R_x, R_y y R_z.

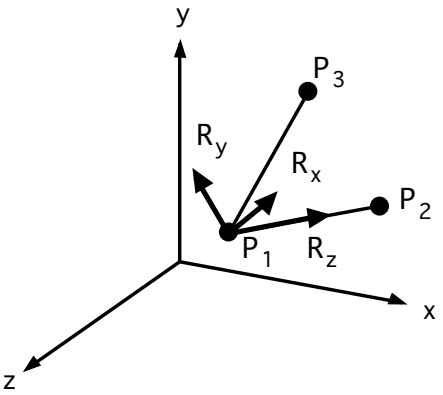


Fig. 2.22.Los vectores unidad Rx, Ry y Rz que se transforman hacia los ejes principales.

Considere otro ejemplo. En la figura 2.23 se muestra un aeroplano definido en el sistema de coordenadas x_p, y_p, z_p y centrado en el origen. Queremos transformar el aeroplano para que apunte en la dirección indicada por el vector DDV (dirección de vuelo), esté centrado en p y no inclinado, como se muestra en la figura 2.24

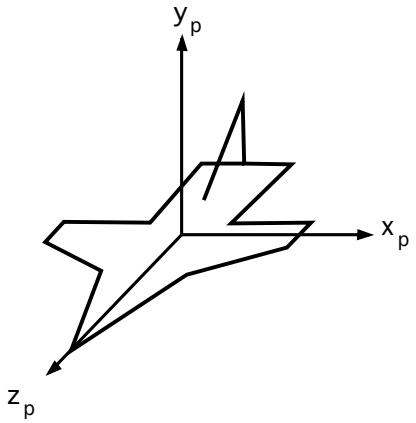


Fig. 2.23.Un aeroplano en el sistema de coordenadas x_p, y_p, z_p.

La transformación necesaria para llevar a cabo esta reorientación consiste en una rotación para apuntar el aeroplano en la dirección correcta, seguida por una traslación del

origen a P. Para hallar la matriz de rotación solo hay que determinar en qué dirección apuntan los ejes x_p, y_p y z_p en la figura 2.24, asegurarse de que las direcciones están normalizadas y luego usar estas direcciones como vectores columna en una matriz de rotación.

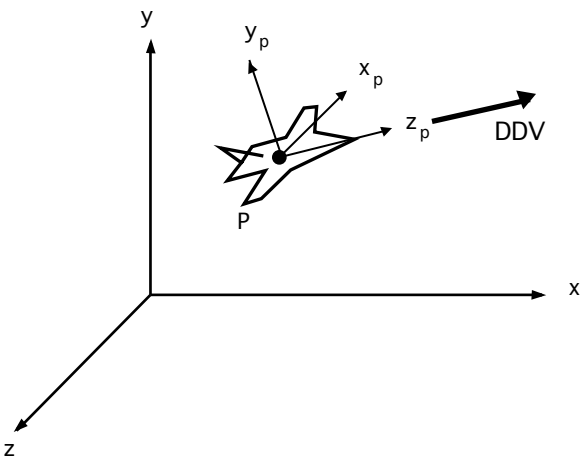


Fig. 2.24.El aeroplano de la figura anterior, ubicado en el punto P y apuntando en la dirección DDV.

El eje *z*, debe transformarse a la dirección DDV, mientras que el eje *x*, debe transformarse a un vector horizontal perpendicular a DDV, o sea, en la dirección $y \times DDV$, el producto cruz entre de *y* y *DDV*. La dirección y_p esta indicada por $z_p \times x_p = DDV\ (y \times DDV)$, el producto cruz de z_p y x_p; por lo tanto las tres columnas de matriz de rotación son los vectores normalizados $|y \times DDV|$, $|DDV \times (y \times DDV)|$ y $|DDV|$:

$$R = \begin{bmatrix} |y \times DOF| & |DOF \times (y \times DOF)| & |DOF| & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 2.77.$$

Resumen

Las transformaciones geométricas básicas son la traslación, la rotación y la escalación. La traslación mueve un objeto con una trayectoria en línea recta de una posición a otra. La rotación mueve un objeto de una posición a otra a lo largo de una trayectoria circular sobre un eje de rotación específico.

Para aplicaciones bidimensionales, la trayectoria de rotación se encuentra en el plano xy sobre un eje que es paralelo al eje z. Las transformaciones de cambio de escala cambian las dimensiones de un objeto con respecto a una posición fija.

Podemos expresar las transformaciones bidimensionales como operadores de matrices de 3 por 3 y las transformaciones tridimensionales como operadores de matrices de 4 por 4, de tal forma que esas secuencias de transformaciones pueden concatenarse dentro de una matriz compuesta.

En general, podemos representar tanto transformaciones bidimensionales como tridimensionales con matrices de 4 por 4. Representar operaciones de transformaciones geométricas con matrices de formulación eficiente, en tanto en cuanto nos permite reducir los cálculos aplicando una matriz compuesta a una descripción de un objeto para obtener su posición transformada.

Para encontrar la matriz de la posición transformada expresamos posiciones de coordenadas como matrices de columna. Elegimos la representación de matriz columna para puntos de coordenadas porque ese es el convenio matemático estándar y, muchos paquetes gráficos siguen dicha convención.

Nos referimos a una matriz de tres o cuatro elementos (vector) como una representación de coordenadas homogéneas. Para transformaciones geométricas, al coeficiente homogéneo se le asigna el valor 1.

Las transformaciones compuestas se forman como multiplicación de matrices de traslación, rotación, cambio de escala y otras transformaciones. Podemos usar combinaciones de traslación y rotación para aplicaciones de animación y podemos usar combinaciones de rotación y escalación para cambiar el tamaño de los objetos en cualquier dirección especificada.

En general, la multiplicación de matrices no es conmutativa. Obtenemos diferentes resultados, por ejemplo, si cambiamos el orden de la secuenca traslación-rotación.

Las transformaciones entre sistemas de coordenadas cartesianos se lleva a cabo con una secuencia de transformaciones traslación.rotación que hacen que los dos sistemas coincidan. Especificamos el origen de coordenadas y vectores de eje para un marco de referencia respecto al marco de referencia original.

En un sistema bidimensional, un vector define completamente las direcciones del eje de coordenadas; pero en un sistema tridimensional, hay que especificar dos de las tres direcciones de los ejes.

Las transformaciones geométricas son transformaciones afines. Esto es, pueden expresarse como una función lineal de posiciones de coordenadas. Traslación, rotación y escalación son transformaciones afines. Transforman líneas paralelas en líneas paralelas y posiciones de coordenadas finitas en posiciones finitas.

3. Modelado Geométrico

Objetivo. *El estudiante conocerá y aplicará las diferentes técnicas de modelado geométrico y su proyección en el área de vista.*

La cantidad total de datos que se deben almacenar en un modelo informático, depende del ámbito de las preguntas que algorítmicamente queramos responder a partir del modelo.

Muchos de los problemas a resolver mediante modelos tienen naturaleza geométrica. Por ejemplo, el problema de hallar la imagen coloreada de un objeto incluye cuestiones geométricas tales como:

1) ¿Qué partes del objeto son visibles para el observador?

2) ¿Qué color ha de ser asignado a cada punto de la imagen?

Si podemos representar en el ordenador la forma geométrica de un objeto, podremos responder a estas preguntas y a muchas otras. De hecho, la información geométrica sobre un objeto es la parte más útil del total de información sobre el objeto. Además, las técnicas para almacenar y procesar la información geométrica son relativamente independientes de un modelo particular. Así, procesos esencialmente iguales de modelado se utilizan en la construcción de modelos de barcos, casas, o zapatos.

Acorde con lo dicho anteriormente, en un modelo tiene sentido separar la información geométrica de los objetos, de la no geométrica. Bajo este planteamiento, al total de información del modelo informático se conoce como modelo del objeto, mientras que la información exclusivamente geométrica constituye el modelo geométrico .

Por lo tanto, el concepto de Modelado Geométrico se refiere al conjunto de métodos utilizados para definir la forma y otras características de los objetos. La construcción de los objetos es normalmente, en si misma, una operación asistida por ordenador. Éstos juegan un papel primordial, ya que sin su potencia de cálculo los procedimientos del Modelado Geométrico solamente podrían aplicarse en modelos de escasa importancia práctica.

Los métodos del Modelado Geométrico vienen a ser un compendio de las técnicas utilizadas en varias disciplinas, como la Geometría Analítica y Descriptiva, la Topología, la Teoría de Conjuntos, el Análisis Numérico, las Estructuras de Datos, el Cálculo Vectorial y los Métodos Matriciales.

Se pueden enumerar tres aplicaciones básicas del Modelado Geométrico:

- Representación de los objetos existentes.
- Diseño de los objetos inexistentes y
- Visualización (rendering) de los objetos.

El Diseño Asistido por Ordenador (CAD) y la Fabricación Asistida por Ordenador (CAM), han sido las principales fuerzas de desarrollo del campo del Modelado Geométrico, aunque otras áreas como la Robótica, Reconocimiento de Formas, Inteligencia Artificial, y el Cálculo Estructural (modelos de elementos finitos) han contribuido también ha su desarrollo.

3.1.Modelado de Superficie

Existen varias razones para querer representar un objeto mediante un modelo de superficie:

- Cuando el objeto mismo es una superficie que podemos suponer sin grosor (por ejemplo, la chapa metálica del capó de un vehículo). Este tipo de representación nos permite visualizar superficies abiertas, mientras que los sólidos se caracterizarán por tener su superficie necesariamente cerrada sobre sí misma.
- Cuando tan sólo nos interesa visualizar su aspecto visual externo, sin detalles sobre su estructura interna, aunque el objeto ocupe un cierto volumen.

- Cuando deseamos realizar una visualización en tiempo real, y para ello utilizamos hardware o software gráfico que está sólo preparado para visualizar polígonos.

En cualquiera de estos casos es conveniente utilizar una representación de la superficie del objeto.

En principio la información sobre una superficie debe dar cuenta de su geometría, de sus propiedades visuales (cómo se comporta frente a la luz) y quizás también de alguna propiedad física (como la elasticidad) si se va a efectuar una simulación física sobre el objeto.

Si la totalidad del objeto consta de diferentes partes (por ejemplo, varios polígonos o varios trozos definidos por diferentes ecuaciones), entonces podemos añadir también información topológica, es decir, sobre cómo estas diferentes partes se conectan entre sí para formar la superficie.

Con esta información adicional el modelo se conoce como una representación de frontera del objeto (b-rep: boundary representation). Esta representación de frontera se utiliza frecuentemente en combinación con un modelo sólido, ya que una de ellas se puede reconstruir a partir de la otra.

En general, las diferentes representaciones no se excluyen entre sí, y muchos programas suelen combinarlas, pasando de una a otra según convenga.

Una característica que suele exigirse a las superficies representadas es que sean variedades bidimensionales (en inglés: superficies 2-manifold), es decir, que no existan puntos singulares donde la superficie se intersecte consigo misma o se abra en varias hojas.

Además de la información geométrica y topológica, es frecuente añadir a la representación de los objetos datos adicionales requeridos para su visualización (como el color, las propiedades visuales del material, textura, etc.) o para efectuar procesos de simulación física (distribución de densidad, temperatura, composición).

Existen dos tipos de modelado de superficie: modelado plano y modelado curvo. En esta sección se hablará únicamente del modelado curvo. El modelado plano consiste en el uso de polígonos para crear superficies. Puede consultar la sección 3.3.1 para observar este tema.

Superficies de Bezier

Las superficies spline son simplemente una extensión bidimensional de las curvas spline. Las superficies spline, se pueden entender como una curva spline en la que el lugar de cada punto de control es sustituido por una curva spline del mismo tipo de la curva inicial. Es esta restricción la que hace que existan exactamente los mismos tipos de curvas que de superficies spline. Esto es, superficies de Bézier, superficies spline, y en ambos casos estas pueden ser racionales o no.

Forma de Bernstein de las superficies de Bezier

La fórmula de una curva de Bézier expresada en la forma de Bernstein, para una curva de grado n es (véase sección 3.3):

$$C_n(u) = \sum_{i=0}^{n} p_i B_i^n(u) \qquad 3.1.$$

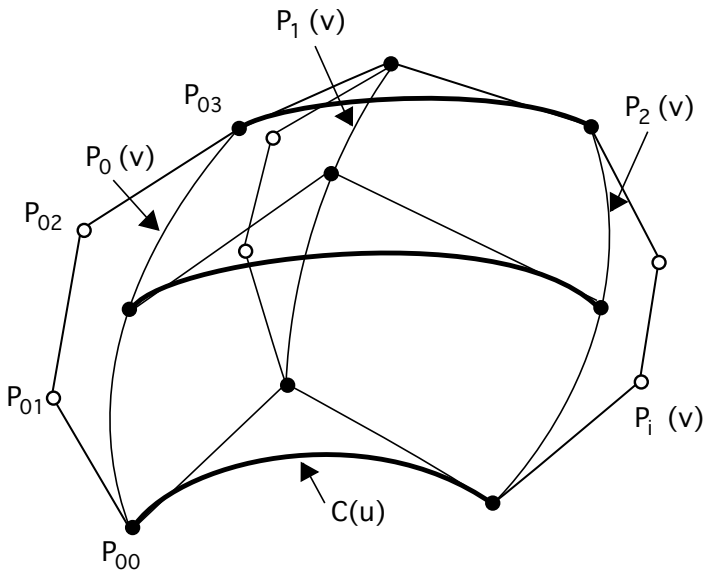
Haciendo que los p_i no sean unos valores constantes sino que sean a su vez el resultado de una función paramétrica de parámetro *v*, resulta que para cada valor concreto del parámetro *v*, la función *C(u)* describe una curva diferente. Haciendo además que las ecuaciones paramétricas que rigen las curvas que siguen los puntos pi de control sean todas curvas de Bézier y lo sean del mismo orden m, cada punto pi de control puede ser expresado como:

$$p_i = p_i(v) = \sum_{j=0}^{m} p_{i,j} B_j^m(v) \qquad 3.2.$$

Combinando las dos ecuaciones anteriores, se obtiene la ecuación paramétrica de una superficie.

$$S_{n,m}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} p_{i,j} B_j^m(v) B_i^n(u) \qquad 3.3.$$

Esta ecuación representa una superficie generada en la forma indicada en la figura siguiente.



P_i (v) Puntos de control generados por una curva Bézier de orden 4

C(u) Curva de Bézier de orden 3 con puntos de control pi

Fig. 3.1.Superficie generada desplazando los puntos de control de una curva de Bézier sobre otra curva de Bézier

Para llegar a la expresión paramétrica 3.3 de las superficies de Bézier, ha sido necesario hacer varias suposiciones; a saber

- La curva que se mueve es una curva de Bézier de grado n fijo en todo su desplazamiento.
- Las curvas que describen los puntos de control son también curvas de Bézier de grado m fijo para todos los puntos de control.

Estas restricciones suponen una seria limitación en cuanto a la forma de las curvas que se pueden representar con esta aproximación tensorial. Por ejemplo, para generar una superficie complicada en un extremo pero muy sencilla en el otro, podría pensarse en utilizar curvas con muchos puntos de control en un extremo y con pocos en el otro.

La expresión que se acaba de obtener, no permite la generación de una superficie como esta. Otro caso es el de una superficie de forma triangular. Mediante la expresión

vista, es necesario que en un extremo se colapsen todos los puntos de control sobre una misma posición, dando lugar a singularidades.

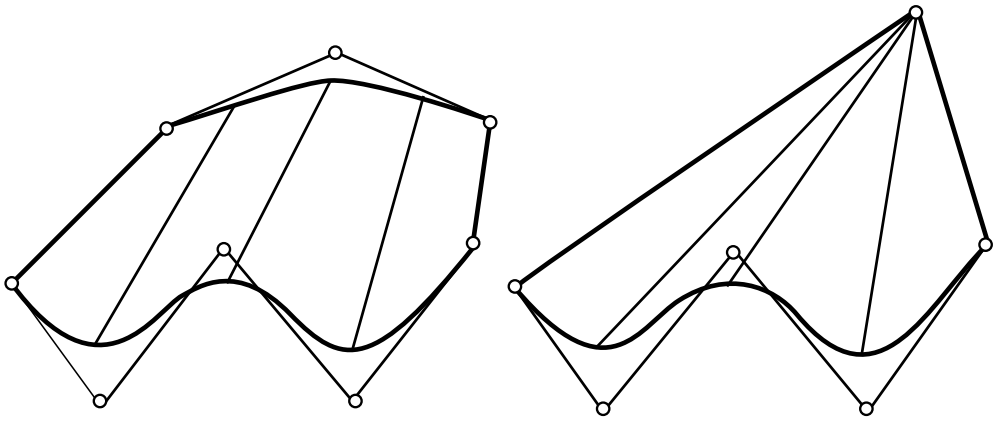


Fig. 3.2.Superficies no adaptadas a la forma tensorial de superficies de Bézier

La expresión de superficies de Bézier en forma de producto de matrices se demuestra de un modo totalmente análogo al empleado para las curvas de Bézier. Para una superficie de Bézier de orden *m* en la dirección *u* y orden *n* en la dirección del parámetro *v*, su expresión en función de los polinomios de Bernstein es la de la ecuación 3.3 y se puede expresar matricialmente como:

$$S_{n,m}(u,v) = (B_0^n(u)..B_n^n(u))\begin{pmatrix} p_{00} & .. & p_{0m} \\ .. & & .. \\ p_{n0} & .. & p_{nm} \end{pmatrix}\begin{pmatrix} B_0^m(v) \\ \\ B_m^m(v) \end{pmatrix}$$

3.4.

en donde el primer vector contiene los polinomios de Bernstein que representan la curva *C(u)*, la matriz de coeficientes p_{ij}, contiene los puntos de control y el último vector contiene los polinómios de Bernstein que definen el desplazamiento de los puntos de control, siguiendo con la interpretación dada inicialmente a la ecuación de las superficies de Bézier. Por supuesto esto es simplemente una interpretación, pues se podría ver igualmente como una curva *C'(v)* cuyos puntos de control son desplazados por *C(u)*.

Como se puede ver, los polinomios de Bernstein se pueden expresar asimismo como producto de matrices, lo que lleva a la expresión 3.5 en donde los coeficientes de las matrices *M* y *N* son los obtenidos para la expresión matricial de las curvas de Bézier .

$$S_{n,m}(u,v) = (u^0..u^n)N^T \begin{pmatrix} p_{00} & .. & p_{0m} \\ .. & & .. \\ p_{n0} & .. & p_{nm} \end{pmatrix} M \begin{pmatrix} v^0 \\ \\ v^m \end{pmatrix}$$

3.5.

Propiedades de las Superficies de Bézier

A continuación se citarán solo aquellas propiedades que conciernen al modelado de superficies:

Polígono envolvente. Cualquier superficie de Bézier siempre está totalmente contenida dentro de la figura geométrica formada por la unión de todos los puntos de control que la definen. Esta propiedad se deduce de la no negatividad de los polinomios de Bernstein para valores del parámetro entre cero y uno.

Interpolación en los extremos. Las curvas límite de una superficie de Bézier son curvas de Bézier. Los puntos de control de estas curvas son la primera y última fila y la primera y última columna de la matriz p_{ij}, en la expresión matricial de la superficie como puede verse en las ecuaciones 3.6. En particular, las cuatro esquinas de la superficie, son los puntos de control p_{00}, p_{0n}, p_{m0} y p_{mn}.

$$S_{n,m}(u=0,v) = (1,0..0)\begin{pmatrix} p_{00} & .. & p_{0m} \\ .. & .. & .. \\ p_{n0} & .. & p_{nm} \end{pmatrix}\begin{pmatrix} B_0^m(v) \\ \\ B_m^m(v) \end{pmatrix} = (p_{00}..p_{0m})\begin{pmatrix} B_0^m(v) \\ \\ B_m^m(v) \end{pmatrix}$$

$$S_{n,m}(u=1,v) = (0,0..1)\begin{pmatrix} p_{00} & .. & p_{0m} \\ .. & .. & .. \\ p_{n0} & .. & p_{nm} \end{pmatrix}\begin{pmatrix} B_0^m(v) \\ \\ B_m^m(v) \end{pmatrix} = (p_{00}..p_{0m})\begin{pmatrix} B_0^m(v) \\ \\ B_m^m(v) \end{pmatrix}$$

3.6.

Derivadas de las superficies de Bézier

En el caso de curvas, la derivada quedaba expresada como una curva de Bézier en la que los puntos de control eran sustituidos por vectores dados por las diferencias de puntos de control consecutivos. Lo mismo ocurre en superficies. Si se empieza considerando la

derivada parcial respecto a *u* (3.7) el término entre corchetes es una curva de Bézier, por lo que su derivada se puede expresar como 3.8

3.7. $$\frac{\partial}{\partial u} S_{n,m}(u,v) = \sum_{j=0}^{m} \frac{\partial}{\partial u}\left[\sum_{i=0}^{n} p_{i,j} B_i^n(u)\right] B_j^m(v)$$

3.8. $$\frac{\partial}{\partial u}\left[\sum_{i=0}^{n} p_{i,j} B_i^n(u)\right] = n\sum_{i=0}^{n-1}(p_{i+1,j} - p_{i,j}) B_i^{n-1}(u) = n\sum_{i=0}^{n-1}(\Delta_j^{1,0}) B_i^{n-1}(u)$$

en donde Δj1,0 representa los vectores obtenidos como diferencia de puntos de control “consecutivos en u” y *Δi0,1* representa los vectores obtenidos como diferencia de puntos de control “consecutivos en v”. De este modo las derivadas primeras resultan en las ecuaciones 3.9 y 3.10.

3.9. $$\frac{\partial}{\partial u} S_{n,m}(u,v) = n\sum_{j=0}^{m}\sum_{i=0}^{n-1}(\Delta_j^{1,0}) B_i^{n-1}(u) B_j^m(v)$$

3.10. $$\frac{\partial}{\partial v} S_{n,m}(u,v) = n\sum_{i=0}^{n}\sum_{j=0}^{m-1}(\Delta_i^{1,0}) B_j^{m-1}(v) B_i^n(u)$$

Es de notar el hecho de que las derivadas fueron obtenidas separadamente para las componentes *u* y *v*. Ello es debido a la forma de creación de las superficies de Bézier como combinación de curvas de Bézier en dos parámetros distintos. El buen comportamiento de esta separación, es también debido a las restricciones hechas en la definición de este tipo de superficies.

Como interpretación gráfica de estas derivadas, es de destacar el que la derivada en cada dirección *u* y *v* sigue la dirección del vector tangente que va de los puntos de control del extremo de la superficie a los puntos de control vecinos, como se indica en la figura 3.3. Esto es importante a la hora de procurar continuidad entre parches vecinos.

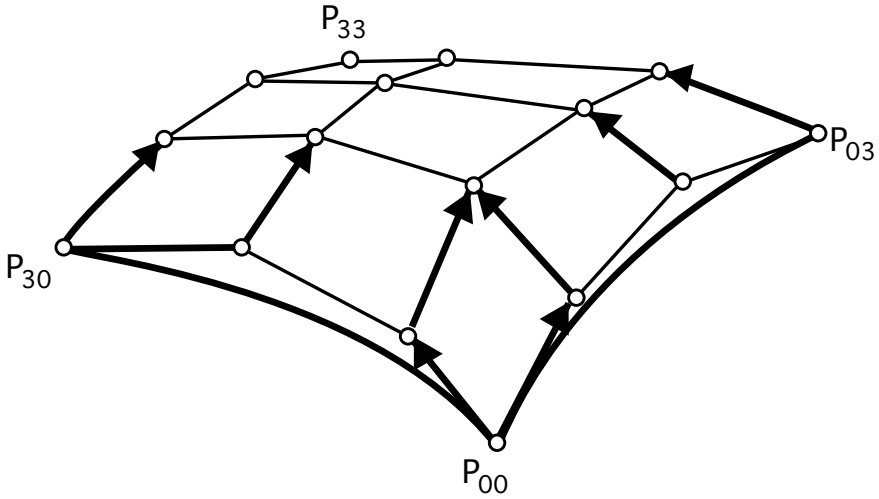


Fig. 3.3.Interpretación gráfica de las derivadas en los extremos de una superficie de Bézier

Vector normal a la superficie

Vector normal a una superficie en un punto es aquel perpendicular a la superficie en ese punto y de módulo 1. Este vector se puede obtener directamente mediante el producto vectorial de dos vectores tangentes. Esto es:

$$n(u,v) = \frac{\frac{\partial}{\partial u} S_{n,m}(u,v) \times \frac{\partial}{\partial v} S_{n,m}(u,v)}{\left| \frac{\partial}{\partial u} S_{n,m}(u,v) \times \frac{\partial}{\partial v} S_{n,m}(u,v) \right|} \qquad 3.11.$$

En particular, en las esquinas del parche, el vector normal se obtiene directamente de los vectores diferencia de puntos de control. Por ejemplo

$$n(0,0) = \frac{(p_{10} - p_{00}) \times (p_{01} - p_{00})}{\left| (p_{10} - p_{00}) \times (p_{01} - p_{00}) \right|} \qquad 3.12.$$

Esta forma de obtener la normal funciona perfectamente en parches bien configurados pero hay ciertos casos especiales que han de ser tenidos en cuenta. El caso mas común es el de los parches degenerados en triangulares pues todos los puntos de control de un extremo son el mismo punto. En este caso, solo se puede calcular una derivada en el extremo degenerado pues la otra derivada está indefinida.

Se podrá evaluar la normal si realmente existe un plano tangente en ese punto. El plano tangente existe si todas las derivadas en la dirección del parámetro que se comporta bien son coplanares. Esto se indica en la figura 3.4. De todos modos, siempre se ha intentado evitar el uso de parches degenerados, buscando formas alternativas de definir la superficie cuando esto ha sido posible.

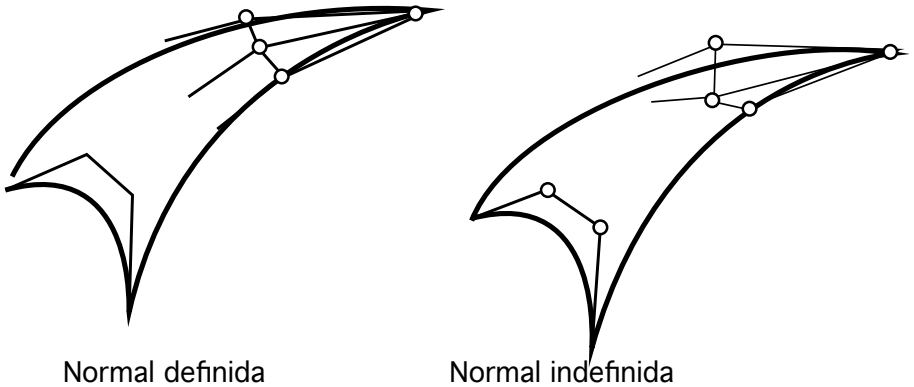


Fig. 3.4.Singularidades en una derivada parcial

La misma ventaja que ya se ha indicado al tratar la forma matricial de las curvas de Bézier en lo referente a su buena adaptación a problemas de programación se da cuando se trabaja con superficies de Bézier. La evaluación de puntos de la superficie y de sus derivadas es muy sencillo si se trabaja con la expresión matricial. Observando la ecuación de la superficie 3.5 se ve que las dependencias con los parámetros están solamente en el primer y último vectores.

Por la sencillez del problema, se pondrá solamente la expresión de las primeras derivadas.

$$\frac{\partial}{\partial u} S_{n,m}(u,v) = (0,1,2u..nu^{n-1}) N^T \begin{pmatrix} p_{00} & .. & p_{0m} \\ .. & .. & .. \\ p_{n0} & .. & p_{nm} \end{pmatrix} M \begin{pmatrix} v^0 \\ \\ v^m \end{pmatrix} \qquad 3.13.$$

$$\frac{\partial}{\partial u} S_{n,m}(u,v) = (u^0 ..u^n) N^T \begin{pmatrix} p_{00} & .. & p_{0m} \\ .. & .. & .. \\ p_{n0} & .. & p_{nm} \end{pmatrix} M \begin{pmatrix} 0 \\ 1 \\ 2v \\ .. \\ mv^{m-1} \end{pmatrix}$$

3.14.

Superficies B-Spline

Del mismo modo que una curva de Bézier no se adapta bien para modelar curvas complejas, las superficies de Bézier sufren el mismo problema, pues estas no son mas que una extensión bidimensional de las anteriores. Como en el caso de curvas, las superficies spline dan la posibilidad de generar superficies formadas por la conexión de sucesivas superficies de Bézier.

El problema es mas sencillo en el caso de curvas pues solo existe un punto de unión y unos valores perfectamente determinados de sus derivadas. En el caso de superficies, la conexión se ha de hacer a lo largo de una curva en todos los puntos de la misma, y las derivadas ya no están tan claramente determinadas.

Continuidad entre superficies de Bézier

Sean $x(u,v)$ e $y(u,v)$ dos parches de Bézier definidos sobre $[u_{I-1}, u_I]x[v_J , v_{J+1}]$ y $[u_I,u_{I+1}]$ $x[v_J , v_{J+1}]$ respectivamente. La superficie será r veces derivable en la unión entre ambas superficies $x(u_I,v)$ e $y(u_I,v)$ si coincide el valor de todas las derivadas hasta el orden r. Esto es

$$\left.\frac{\partial^r}{\partial u^r} x(u,v)\right| = \left.\frac{\partial^r}{\partial u^r} y(u,v)\right|_{u=u_I}$$

3.15.

Supóngase ahora que ambos parches están dados en su forma de Bézier y que sus polígonos de control son $\{p_{ij}\}$; $0 \le i \le m$, $0 \le j \le n$ y $\{p_{ij}\}$; $m \le i \le 2m$, $0 \le j \le n$. Como los $B_j m(v)$ son linealmente independientes, se pueden comparar directamente los coeficientes

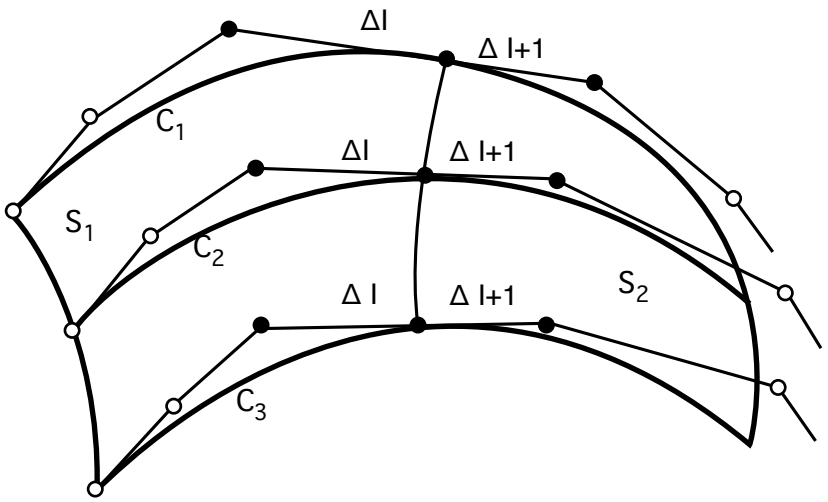
3.16. $$\left(\frac{1}{\Delta_{I-1}}\right)^r \sum_{j=0}^{m} (\Delta^{r,0} p_{n-r,j}) B_j^m(v) = \left(\frac{1}{\Delta_I}\right)^r \sum_{j=0}^{m} (\Delta^{r,0} p_{m,j}) B_j^m(v)$$

3.17. $$\left(\frac{1}{\Delta_{I-1}}\right)^r (\Delta^{r,0} p_{n-r,j}) B_j^m(v) = \left(\frac{1}{\Delta_I}\right)^r (\Delta^{r,0} p_{m,j}); j = 0..n$$

Esta es la condición para continuidad hasta la derivada *r* ésima aplicada a las n+1 filas de puntos de control de ambos parches de Bézier. Por lo tanto la condición de continuidad entre parches de Bézier se puede definir como: *Dos parches adyacentes son Cr continuos en su frontera común si y solo si todas las filas de puntos de control forman curvas de Bézier con continuidad Cr*.

Nuevamente el problema de superficies se reduce a la solución separada de varios problemas de curvas. Se ha visto la condición de continuidad en *u*, pero esta aplica igualmente en la dirección *v*.

La condición de continuidad en la primera derivada, ilustrada en la figura 3.5, obliga a que todos los polígonos formados por los puntos p_{0j}, p_{1j}, ... p_{2mj} sean polígonos de control de curvas a trozos de Bézier con continuidad en la primera derivada. Para que se de esta condición, los puntos de control $p_{m-1,j}$, $p_{m,j}$ y $p_{m+1,j}$ han de ser colineales y estar en la relación ΔI : ΔI+1. Esta relación ha de ser la misma para las *j* curvas que intervienen en el diseño de la superficie.



C_1 C_2 C_3 Curvas de Bézier C^1 Continuas

S_1 S_2 Superficies de Bézier C^1 Continuas

Fig. 3.5.Parches de Bézier con continuidad en la primera derivada

Subdivisión de superficies de Bézier

El algoritmo de Casteljau, en el caso de curvas sirve para obtener puntos de una curva de Bézier así como para subdividirla. En superficies, este algoritmo produce un resultado similar.

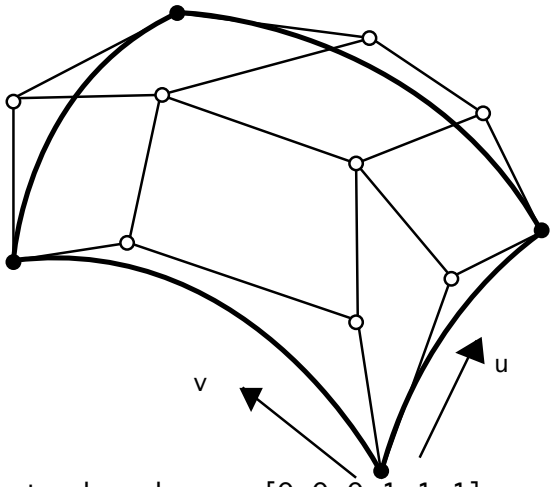
Aplicando el algoritmo de Casteljau en un valor fijo del parámetro u de la superficie a todas las j curvas que forman su polígono de control, se produce una subdivisión del parche de Bézier en dos nuevos parches. La unión entre estos parches es continua pues lo son asimismo las *j* curvas. Si se desea subdividir un parche en cuatro nuevos subparches, se aplicará el algoritmo de subdivisión en un parámetro para luego aplicar la subdivisión en el otro sobre el resultado anterior. No importa el orden en que se hagan las subdivisiones; el resultado es el mismo.

Expresión de superficies B-Spline

Una superficie B-spline se expresa como

$$x(u,v) = \sum_i \sum_j p_{i,j} N_i^n(u) N_j^m(v) \qquad 3.18.$$

en donde se asume la existencia de un vector de nodos en *u* y otro en *v*. Como en el caso de las curvas B-spline, si los vectores de nodos tienen nodos con multiplicidades *n* y *m* en sus extremos inicial y final, la superficie B-spline pasará por los puntos de control de los vértices de la superficie. Un ejemplo es la figura 3.6 en la que se muestra una superficie B-spline bicúbica y sus vectores de nodos.



Vector de nodos en u [0, 0, 0, 1, 1, 1]

Vector de nodos en v[0, 0, 0, 1, 2, 2, 2]

Fig. 3.6.Superficie B-spline bicúbica con multiplicidades en el vector de nodos de forma que la superficie llega a los puntos de control extremos

Expresión como superficies a trozos de Bézier

A la hora de trabajar con superficies suele ser mas cómodo hacerlo con las mismas definidas como conexión de parches de Bézier. El proceso puede entenderse mejor si la ecuación 3.18 se reescribe como

$$x(u,v) = \sum_i N_i^n(u) \left[\sum_j p_{ij} N_j^m(v) \right] \qquad 3.19.$$

en donde puede verse que para cada término en *i* del sumatorio, la ecuación entre corchetes describe una curva B-spline. Se puede convertir esta ecuación a la forma de Bézier, para luego convertir la ecuación en *v*. Esto se corresponde con la interpretación del polígono de control de la superficie B-spline como formado por filas de puntos de control de curvas B-spline y columnas de puntos de control de curvas B-spline. El proceso para pasar

de superficies B-spline a superficies a trozos de Bézier consiste en la conversión por separado de los polígonos de control en las direcciones u y v separadamente en polígonos de Bézier

Superficies Racionales

La expresión de una superficie racional en forma de Bézier es

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} p_{i,j} B_i^n(u) B_j^m(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} B_i^n(u) B_j^m(v)}$$

3.20.

en donde el numerador representa una superficie con polígono de control formado por los puntos $w_{ij}p_{ij}$ y el denominador es un número real obtenido con la misma expresión que la superficie del numerador y con 'polígono de control' formado por los valores w_{ij}.

El comportamiento de estas superficies es análogo al de las curvas racionales de Bézier debido a que son simplemente una extensión bidimensional de las expresiones obtenidas para el caso de curvas. En ellas se han de tener en cuenta los mismos criterios en cuanto a la no negatividad de los pesos, a la forma de evaluación o a la aplicación del algoritmo de Casteljau o el de inserción de nodos.

Modelado de sólido

El Modelado Sólido es una rama relativamente reciente del Modelado Geométrico, que hace hincapié en la aplicabilidad general de los modelos, e insiste en crear solamente modelos "completos" de los sólidos, es decir, modelos que son adecuados para responder algorítmicamente (sin la ayuda externa del usuario) a cualquier pregunta geométrica que se formule.

Los principales esquemas de Modelado Sólido desarrollados son el de Representación de Fronteras (Boundary Representation o B-Rep) y el de la Geometría Constructiva de

Sólidos (Constructive Solid Geometry o CSG), aunque existen muchos otros, como el modelado de barrido translacional y rotacional, o los esquemas de modelado híbridos.

El objetivo de "aplicabilidad general" diferencia los esquemas de Modelado Sólido de otros esquemas de modelado geométricos, los cuales se utilizan en casos especiales. Así, los modelos gráficos (graphical models) se utilizan para describir el dibujo técnico de los objetos, por ejemplo en ingeniería.

Los modelos de formas (shape models) representan imágenes de los objetos. Pueden ser una colección sin estructurar de elementos de una imagen, o poseer cierta estructura para, por ejemplo, realizar operaciones de tratamiento de imágenes. Los modelos de superficie (surface models) dan información detallada sobre superficies, pero no siempre proporcionan la información suficiente para determinar todas las propiedades geométricas.

Fundamentos del Modelado Sólido

Un sistema de Modelado Sólido maneja dos tipos de información: los datos geométricos y datos topológicos. Los datos geométricos son aquellos que representan geométricamente los objetos (coordenadas de vértices, ecuaciones de superficies...). En cambio, los topológicos se refieren a cómo conectar componentes geométricos para conseguir un modelo.

A continuación veremos las características topológicas más útiles en modelado (en 2 y 3D), que sirven principalmente para validar la integridad de los modelos.

Teorema del camino cerrado

El giro total a lo largo de un camino cerrado es un entero múltiplo de 360°. A este entero se le conoce como número de rotaciones (Nr) del camino. El valor de Nr es independiente de dónde se comience el recorrido del camino y de cómo esté orientado.

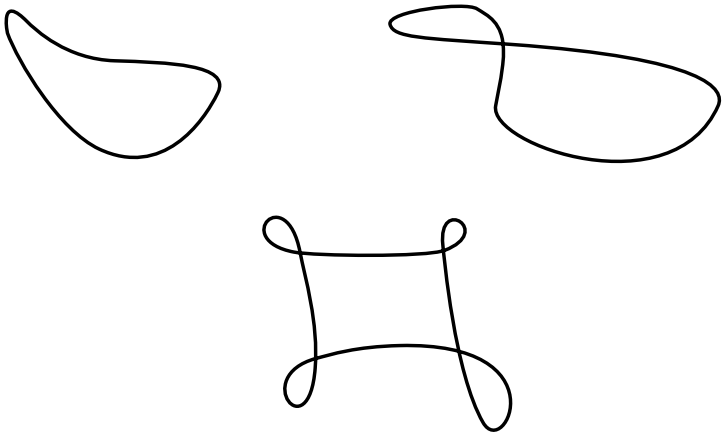


Fig. 3.7.Ejemplo de caminos cerrados con valores de Nr diferentes

Teorema del camino cerrado simple

Un camino cerrado simple es aquél que no se corta a sí mismo.

El teorema del camino cerrado simple dice que el giro total en un camino cerrado que no se corte a sí mismo es ±360°. En otras palabras, el número de rotación Nr de un camino cerrado simple es ±1.

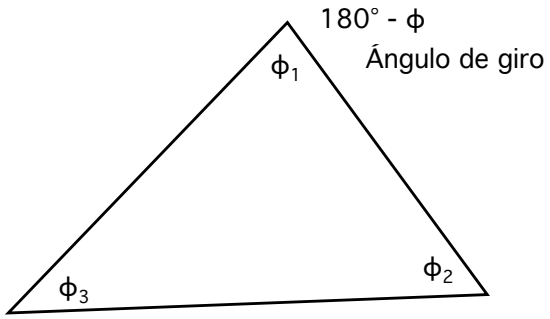


Fig. 3.8.La suma de los ángulos de giro en un camino cerrado simple es ±360°

Por ejemplo, en el triángulo de la figura anterior, siendo φ1, φ2, y φ3 sus ángulos internos, la suma de los ángulos de giro es:

$$(180^\circ-\phi_1) + (180^\circ-\phi_2) + (180^\circ-\phi_3) = 3\cdot180 - (\phi_1+\phi_2+\phi_3) = 540-180 = 360^\circ$$

Recordemos que la suma de los ángulos interiores de un triángulo (en la geometría euclidiana) es siempre 180°.

La importancia del teorema anterior en modelado radica en que es relativamente fácil medir los giros de los camino cerrados (ej., los ángulos de los polígonos), aplicando métodos

locales. De este modo, si al finalizar el recorrido de un camino cerrado (aristas de un polígono) el acumulado final del giro es diferente a 360° podemos asegurar que el modelo no es íntegro (está mal construido), ya que uno de sus polígonos se autointerseca, como ocurre en el ejemplo de la figura 3.9.

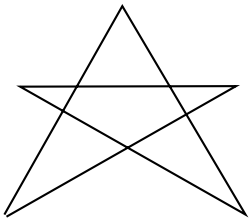


Fig. 3.9.La suma de los ángulos de giro en un camino cerrado no simple es diferente de 360°

Deformación de curvas y planos

El teorema de deformación de curvas y planos dice que *cualquier curva cerrada simple de un plano se puede transformar mediante una isotopía de ambiente en un cuadrado*.

En 1936, H.Whitney y W.C.Graustein observaron que dos caminos cerrados coplanarios pueden transformarse el uno en el otro sólo si tienen el mismo giro total. Este resultado junto con el teorema anterior se puede utilizar para probar el teorema del camino cerrado simple: por una parte, cualquier curva cerrada simple en un plano se puede transformar en un cuadrado, y además, un camino cerrado sólo puede transformarse en otro si tiene el mismo giro total. Entonces podemos deducir que cualquier camino cerrado simple tiene el mismo giro total que un cuadrado, es decir, ±360°, con lo que hemos demostrado el teorema.

Las transformaciones pueden ser de dos tipos:

- Homotopía regular. Es una deformación que produce cambios en el camino.
- Isotopía de ambiente. Supongamos que tenemos el camino dibujado sobre una lámina de goma. Si estiramos o encogemos la lámina por diferentes puntos, podemos conseguir que varíe la trayectoria del camino, o sea, una isotopía de ambiente o transformación de hoja de goma.

La homotopía regular es más violenta (no conservadora) que la isotopía de ambiente (conservadora), ya que puede crear o eliminar puntos de corte, cosa que no hace ésta

última. De hecho, una isotopía de ambiente es un caso especial de la homotopía regular, en la que no se añaden ni eliminan puntos de corte.

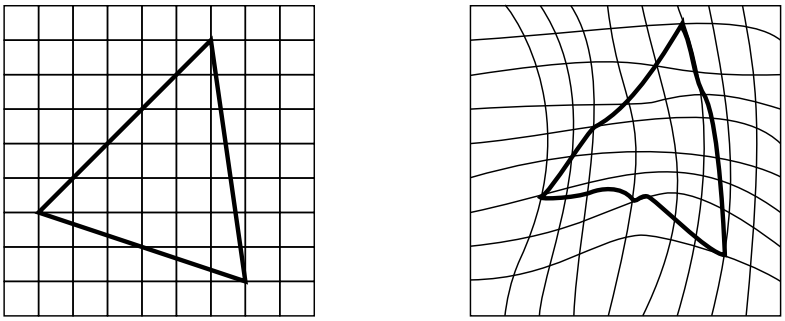


Fig. 3.10.Deformación conservadora en una hoja de goma

Teorema de la curva de Jordan

Se deduce del teorema de deformación de curvas y planos y dice que *cualquier curva cerrada simple en un plano, divide a éste en dos regiones: una interior y otra exterior*.

Da igual de qué curva cerrada simple se trate, pues siempre va a dividir al plano en una parte interior a la curva y otra exterior. Este teorema es válido sólo para curvas que estén en superficies planas, ya que podemos trazar una curva cerrada simple en un toro, por ejemplo, y no dividirlo en dos regiones.

El interior de la curva es deformable mediante una isotopía de ambiente y puede transformarse en un cuadrado. Una región, plana o no, que pueda ser transformada mediante una isotopía de ambiente en un cuadrado se llama disco topológico y se caracteriza porque no tiene huecos ni puntos aislados en él.

Ángulo de exceso

Se conoce como ángulo de exceso (o simplemente exceso), *el ángulo de giro implícito a un camino cerrado trazado sobre una superficie*. También se define como el giro que experimenta un puntero de referencia cuando es llevado alrededor de un camino cerra- do. Como pronto se verá, el ángulo de exceso está íntimamente relacionado con la curvatura de las superficies.

El concepto de ángulo de exceso sirve para generalizar el teorema del camino cerrado simple, de modo que también sea válido para caminos cerrados simples sobre superficies

curvas. La generalización de este teorema nos dice que el giro total a lo largo de un camino cerrado simple, más el ángulo de exceso, debe ser igual a 360°, es decir, que

3.21. $$T + E = 360$$

donde:

T = Giro total a lo largo del camino

E = Ángulo de exceso a lo largo del camino

Para aclarar este concepto obsérvese la figura siguiente.

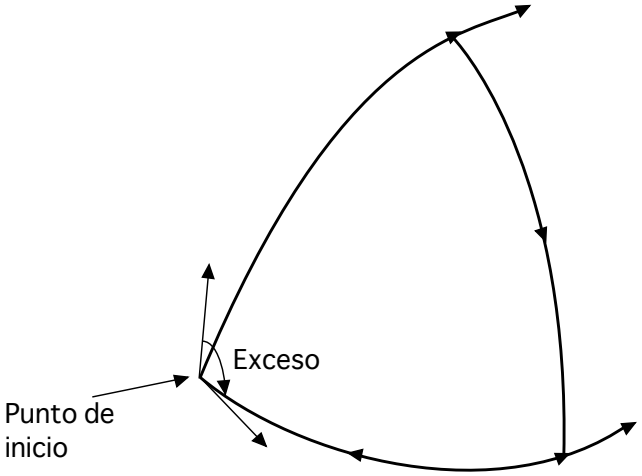


Fig. 3.11.Concepto de ángulo de exceso

En esta figura se representa un cuadrante esférico. Supongamos que comenzamos a andar desde el punto de inicio, con un puntero sobre nuestra cabeza apuntando hacia el polo. Al llegar a éste, giramos 90° para volver al ecuador (obsérvese que sólo giramos nosotros, no el puntero, por lo que entre nuestra nariz y el puntero hay un ángulo de 90°).

De vuelta en el ecuador, damos otro giro de 90° para regresar al punto de inicio (ahora nuestra nariz y el puntero miran en direcciones opuestas). Una vez allí damos otro giro de 90° que nos dejará en la posición inicial.

Se puede ver que el giro total que hemos realizado es de 90° x 3 = 270°. Luego *T* = 270°. En cambio lo que ha girado el puntero (ángulo de exceso) es E = 90o. Sumando ambas cantidades obtenemos 360° que es lo que afirmaba el teorema.

Cuanto más curva es la superficie mayor es el ángulo de exceso. Si la superficie es plana, el ángulo de exceso es cero.

Propiedades del ángulo de exceso

1. Para un determinado camino, su ángulo de exceso es siempre el mismo, independientemente de donde comience el recorrido.
2. El ángulo de exceso es aditivo. Así, el exceso de cualquier polígono es igual a la suma de los excesos de los subpolígonos en que se subdivida.

Curvatura total de las superficies

Para cualquier disco topológico en una superficie cualquiera, el exceso que se obtiene alrededor de su frontera es igual a la curvatura total del interior.

Como el exceso posee la propiedad aditiva, si se subdivide una superficie cualquiera en piezas o trozos poligonales, y para cada polígono se calcula el ángulo de exceso, la suma de los excesos de cada polígono se conoce como curvatura total de la superficie, y se representa por *K*.

La curvatura total es una constante topológica para todas las superficies cerradas; por ejemplo, todas las esferas tienen la misma curvatura total. Por tanto, podemos usar el ángulo de exceso para averiguar la curvatura total de un objeto, simplemente recorriendo caminos cerrados dibujados en la superficie y calculando sus ángulos de exceso.

Objetos de Euler.

A los objetos cerrados tridimensionales de orden $g \geq 0$ que verifican ciertos requisitos de construcción, se les denomina objetos de Euler. Dichas normas son:

1. Todas sus caras (curvas o planas) han de ser discos topológicos.
2. Cada arista une sólo dos caras y todas finalizan en un vértice en cada extremo.
3. Por lo menos tres aristas se unen en un vértice.

El los objetos de Euler se cumple que:

3.22. $$V - A + C = 2(S - P)$$

donde

S: número de superficies inconexas del objeto.

P: total de pasajes (túneles) en el objeto.

Como el número de pasajes en un objeto es igual al total de “asas” o agujeros que posea, ocurre que en la ecuación anterior *P* es igual al orden del objeto (*P* = *g*). Para entender mejor el significado de *S*, la figura 3.12 muestra tres objetos euleria- nos, con una, dos y tres superficies inconexas.

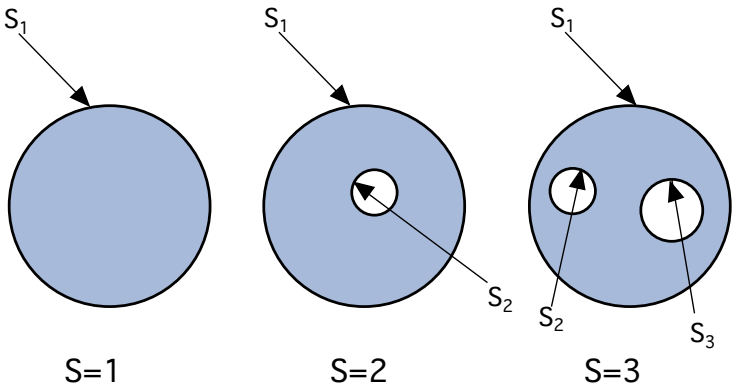


Fig. 3.12.Objetos de Euler con diferentes valores de S

En la figura 3.13 se pueden ver algunos ejemplos de objetos eulerianos, cuando S = 1 y P = 0. Ver que en este caso la Ec. 16 se reduce a la ecuación de Euler.

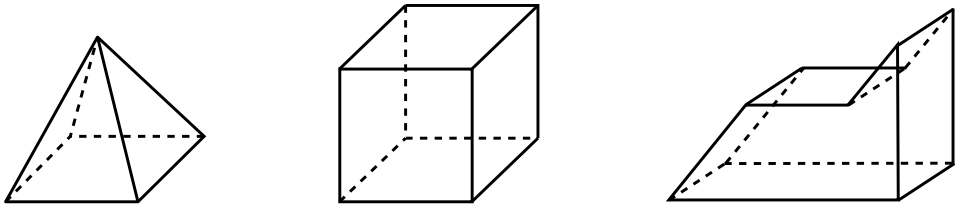
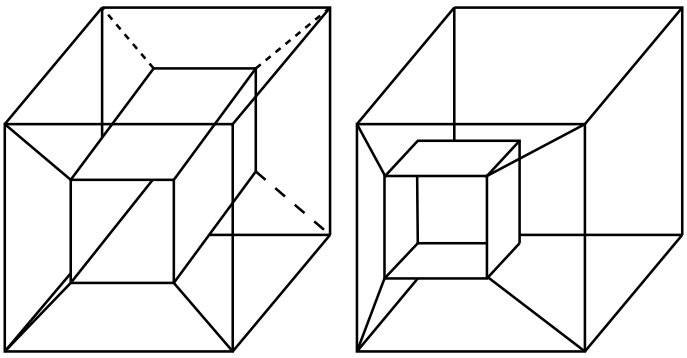


Fig. 3.13.Ejemplos de objetos de Euler

En la figura 3.14 hay otros dos objetos de Euler. El izquierdo es de grado 1, ya que tiene un túnel que lo cruza; el otro, aunque posee una cavidad, ésta no llega a atravesar el modelo. Por tanto, se trata de un objeto topológicamente equivalente a una esfera, es decir, de grado 0.



V - A + C = 2(S - P)
16 - 32 +16 = 2(1 - 1)

V - A + C = 2(S - P)
16 - 28 +14 = 2(1 - 0)

Fig. 3.14.Objetos eulerianos de grado 1 y 0, respectivamente

Concepto generalizado de frontera

Si tenemos una región n-dimensional R^n en un espacio n-dimensional E^n, sus puntos pueden ser clasificados como: los que están dentro de la región y los que están en su límite o frontera. Por tanto, el conjunto de puntos de R^n estará formado por el subconjunto de puntos interior (iR), más el conjunto de los puntos de la frontera (fR).

3.23. $$R = [iR, fR]$$

Según lo anterior, cualquier punto en el espacio E^n, tiene la siguiente propiedad con respecto a R^n:

- Pertenecer al interior de R^n, es decir a iR.
- Pertenecer a la frontera de R^n, es decir a fR.
- No pertenecer a R^n.

En la figura 3.15, podemos ver un ejemplo de estos conjuntos.

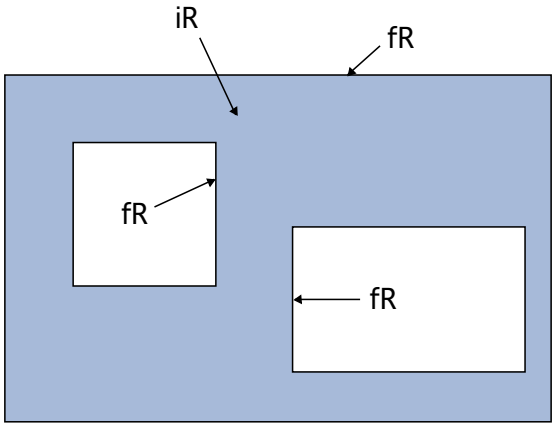


Fig. 3.15.Figura plana con sus puntos frontera e interiores

Operadores booleanos

Como acabamos de ver, los objetos geométricos pueden considerarse como conjuntos de puntos, tanto de interior como de frontera. En Modelado Sólido, para construir objetos complejos se utilizan operaciones entre conjuntos, tales como unión (∪), intersección (∩) y diferencia (–). A estos operadores se les conoce en modelado como operadores Booleanos.

A la hora de aplicar los operadores booleanos sobre objetos simples para formar modelos más complejos, es importante que los modelos obtenidos sean íntegros (topológicamente válidos), y que además sean dimensionalmente homogéneos, es decir, que su dimensión sea igual a la de los objetos que se combinan. La figura 3.16 muestra dos ejemplos donde se pierde la homogeneidad dimensional, después de efectuar la intersección entre objetos válidos de 2 y 3 dimensiones.

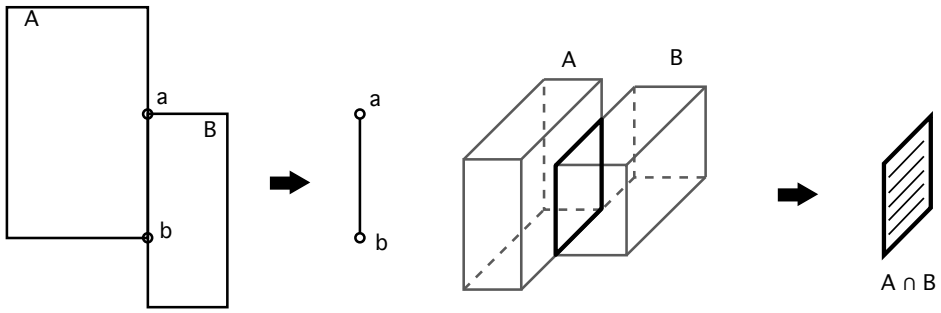


Fig. 3.16.Intersección que presenta un resultado degenerado

Operación de conjuntos booleanos

Independientemente de cómo representemos los objetos, nos gustaría poder combinarlos para formar nuevos. Uno de los métodos más intuitivos y comunes para combinar objetos son las operaciones booleanas de conjuntos, como la unión, la diferencia y la intersección, ilustradas en la figura 3.17.

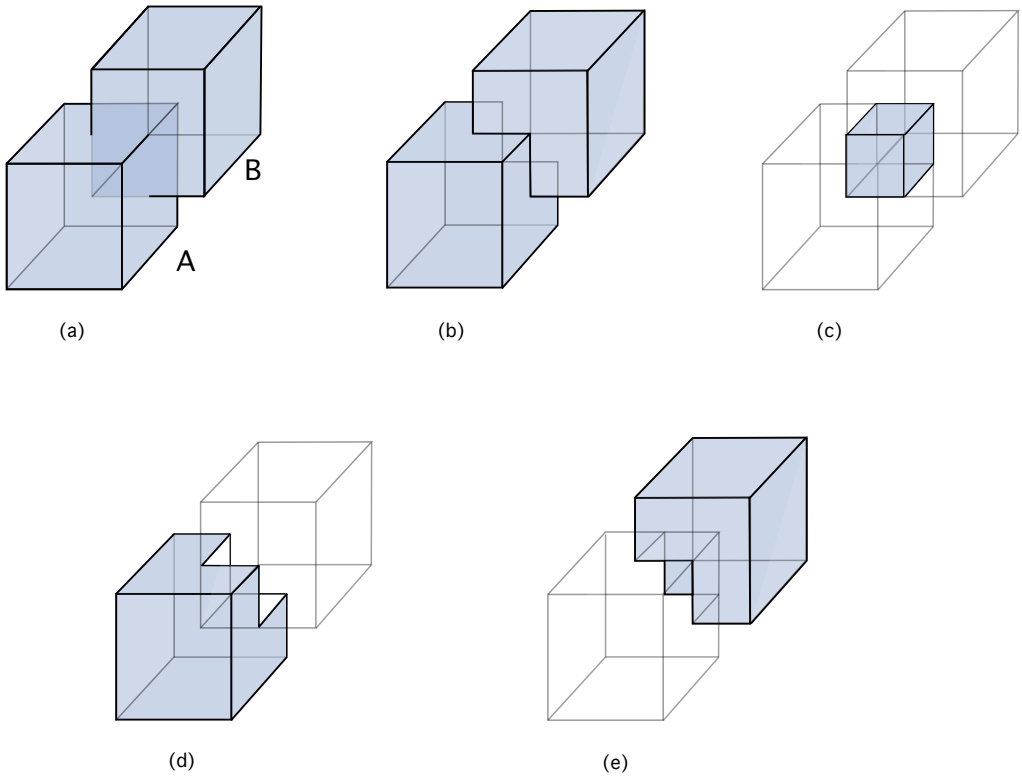


Fig. 3.17.Operaciones booleanas (a) Objetos A y B, (b) A ∪ B, (c) A ∩ B, (d) A − B, (e) B −A

Estas operaciones son los equivalentes tridimensionales de las operaciones booleanas bidimensionales familiares. Sin embargo, la aplicación de una operación booleana ordinaria de conjuntos a dos objetos sólidos no necesariamente produce un objeto sólido. Por ejemplo, la interseccion ordinaria de dos cubos que se unen en un solo vértice es un punto.

En lugar de emplear los operadores booleanos ordinarios de conjuntos, usaremos los operadores booleanos regularizados de conjuntos, denotados como ∪*, ∩* y −*, y definidos de manera que las operaciones con sólidos siempre generen sólidos. Por ejemplo, la

intersección booleana regularizada de dos cubos que se unen en un solo vértice es el objeto nulo.

Para examinar la diferencia entre los operadores ordinarios y los regularizados, podemos considerar cualquier objeto como definido por un conjunto de puntos y dividido en puntos interiores y puntos de frontera.

Los puntos de frontera son aquellos cuya distancia al objeto y al complemento del objeto es cero. Los puntos de frontera no tienen que formar parte del objeto. Un conjunto cerrado contiene todos sus puntos de frontera, mientras que un conjunto abierto no contiene ninguno. La unión de un conjunto con el conjunto de sus puntos de frontera se conoce como cerradura del conjunto.

La frontera de un conjunto cerrado es el conjunto de sus puntos de frontera, mientras que el interior, consiste en todos los demás puntos del conjunto y por ende es el complemento de la frontera con respecto al objeto. La regularización de un conjunto se define como la cerradura de los puntos interiores del conjunto.

Un conjunto que es igual a su propia regularización se conoce como conjunto regular. Observe que un conjunto regular no puede contener ningún punto de frontera que no sea adyacente a un punto interior; es decir no puede tener puntos, líneas o superficies de frontera "colgantes", Podemos definir cada operador booleano regularizado de conjuntos en función de un operador ordinario, como

3.24. $$A\ op^*\ B = cerradura(\text{interior}(A\ op\ B))$$

donde op es ∪, ∩ ó −. Los operadores booleanos regularizados de conjuntos producen sólo conjuntos regulares cuando se aplican a conjuntos regulares.

Comparemos ahora las operaciones booleanas ordinarias y regularizadas de conjuntos aplicadas a conjuntos regulares. Considere los dos objetos de la figura 3.18(a), ubicados como se muestra en la figura 3.18(b). La intersección booleana ordinaria de los dos objetos contiene la intersección del interior y la frontera de cada objeto con el interior y la frontera del otro, como se muestra en la figura 3.18(c).

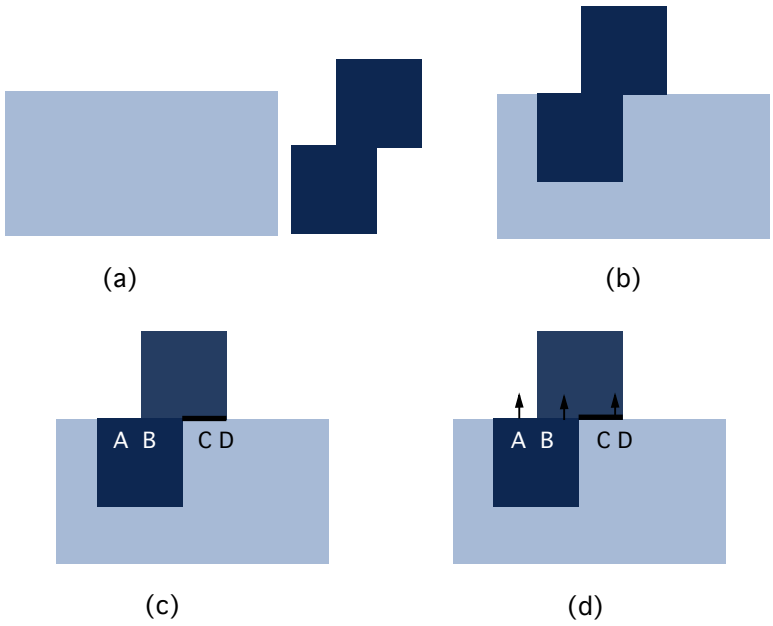


Fig. 3.18.Intersección booleana. (a) Dos objetos, mostrados en sección transversal. (b) Posiciones de los objetos antes de la intersección. (c) Intersección booleana ordinaria quw produce una cara colgante, mostrada como la línea CD en la sección transversal. (d) Intersección booleana regularizada que incluye una porción de la frontera resultante si los dos objetos yacen en el mismo lado (AB), y la excluye si los objetos yacen en los lados opuestos (CD). Las transacciones frontera-interior (BC) siempre se incluyen.

En cambio, la intersección booleana regularizada de los dos objetos presentada en la figura 3.18(d), contiene la intersección de sus anteriores y la intersección del interior de cada una con la frontera de la otra, pero sólo un subconjunto de la intersección de sus fronteras. El criterio que se emplea para definir este subconjunto determina la forma en que la intersección booleana regularizada difiere de la ordinaria, en la cual se incluyen todas las partes de la intersección de las fronteras,

Por intuición, una porción de la intersección frontera-frontera será incluida en la intersección booleana regularizada si y sólo si los interiores de ambos objetos se encuentran en el mismo lado de esta porción de frontera compartida. Como los puntos interiores de los dos objetos que son directamente adyacentes a la porción de la frontera están en la intersección, también hay que conservar la porción de la frontera para mantener la cerradura.

Considere el caso de una porción de frontera compartida que se encuentre en caras coplanares de dos poliedros. Es fácil determinar si los interiores se hallan en el mismo lado de la frontera compartida si los dos objetos se definen de manera que las normales a sus superficies apunten hacia afuera (o hacia adentro). Los interiores estarán del mismo lado si sus normales apuntan en la misma dirección. Por lo tanto, se incluye el segmento AB en la figura 3.18(d). Recuerde que siempre se incluyen aquellas partes de la frontera de un objeto que intersectan el interior del otro objeto, como el segmento BC.

Considere lo que sucede si los interiores de los objetos se encuentran en lados opuestos de la frontera compartida, como ocurre con el segmento CD. En estos casos no se incluye en la intersección ninguno de los puntos interiores adyacentes a la frontera, Por consiguiente, la porción de la frontera compartida no es adyacente a ninguno de los puntos interiores del objeto resultante y. por ende, no se incluye en la intersección regularizada.

Esta restricción adicional con respecto a cuáles son las porciones incluidas de la frontera compartida garantiza que el objeto resultante sea un conjunto regular. La normal a la superficie de cada cara de la frontera del objeto resultante es la normal a la superficie que contribuyó dicha parte de la frontera. Una vez que se ha determinado cuáles caras se encuentran en la frontera, se incluye una arista o un vértice de la intersección frontera-frontera en la frontera de la intersección si la arista o el vértice es adyacente a una de dichas caras.

Los resultados de cada operador regularizado se pueden definir en función de los operadores ordinarios aplicados a las fronteras y a los interiores de los objetos. En la tabla 3.19 se indica la manera en que se definen los operadores regularizados para objetos A y B cualesquiera; en la figura 3.20 se presenta el resultado de efectuadas operaciones. Ab y Ai son la frontera y el interior de A, respectivamente. Ab ∩ Bb iguales es la parte de la frontera compartida por A y B donde Ai y Bi están en el mismo lado. Este resultado es el caso de un punto b en la frontera compartida si al menos un punto i adyacente a él pertenece a Ai y a Bi. Ab ∩ Bb diferentes es la parte de la frontera compartida por A y B para la cual Ai y Bi caen en lados opuestos. Esto se aplica a b si no es adyacente a ningún punto i. Cada uno de los operadores regularizados está definido por la unión de los conjuntos asociados con las filas que tienen un asterisco (*) en la columna del operador.

OPERACIONES BOOLEANAS REGULARIZADAS DE CONJUNTOS			
Conjunto	A ∪* B	A ∩* B	A −* B
$A_i \cap B_i$	*	*	
$A_i - B$	*		*
$B_i - A$	*		
$A_b \cap B_i$		*	
$B_b \cap A_i$		*	*
$A_b - B$	*		
$B_b - A$	*		
$A_b \cap B_b$ iguales	*	*	
$A_b \cap B_b$ diferentes			*

Fig. 3.19.Tabla de operaciones booleanas regularizadas

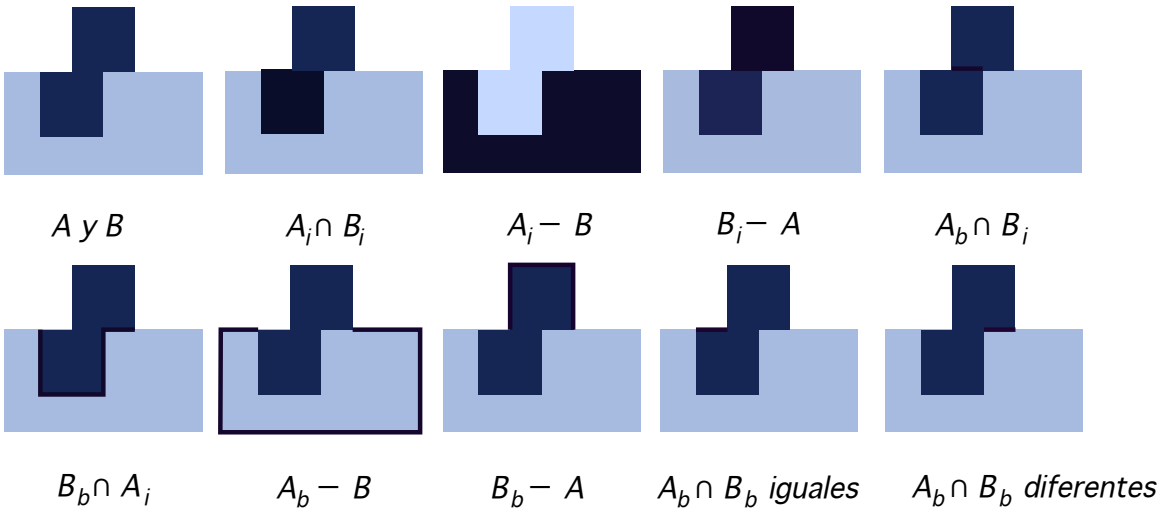


Fig. 3.20.Operaciones booleanas ordinarias aplicadas a subconjuntos de dos objetos

Observe que, en todos los casos, cada porción de la frontera del objeto resultante se encuentra en la frontera de uno o de los dos objetos originales. Al calcular A U* B ó A ∩* B,

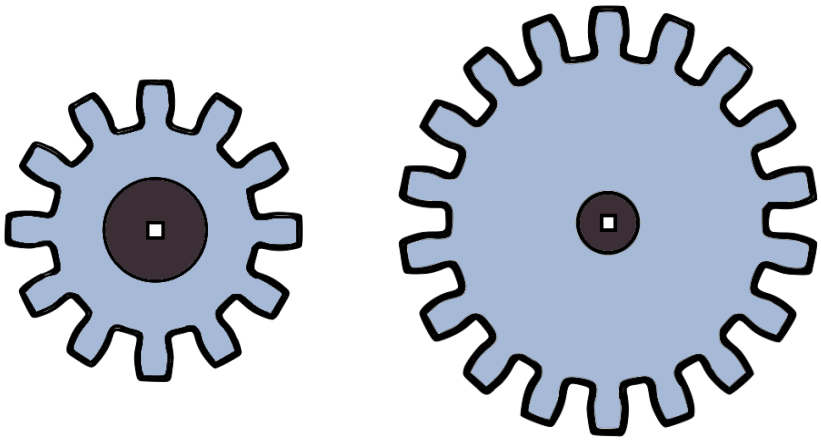
la normal a la superficie de cada cara del resultado se hereda de la normal a la superficie de la cara correspondiente de uno o ambos objetos originales. Sin embargo;en el caso de A −* B, la normal de la superficie decada cara del resultado en donde se ha usado B para excavar A, debe apuntar en la dirección opuesta a la normal a la superficie de B en dicha cara. Esto corresponde a las porciones de frontera Ab ∩ Bb, diferentes y Bb ∩ Ai. Alternativamente. A −* B se puede reescribir como A ∩* . Podemos obtener (el complemento de B) complementando el interior de las B e invirtiendo las normales a su frontera.

En la mayoría de los métodos que analizaremos, los operadores booleanos regularizados de conjuntos se han utilizado como técnica de interfaz con el usuario para construir objetos complejos a partir de objetos sencillos. Estos operadores también se incluyen explícitamente en uno de los métodos, la geometría sólida constructiva. En las secciones que siguen describiremos diversas maneras de representar objetos sólidos sin ambigüedades.

Generación de ejemplares de primitivas

En la generación de ejemplares de primitivas, el sistema de modelado define un conjunto de formas sólidas primitivas tridimensionales que son relevantes para el área de aplicación. Estas primitivas suelen parametrizarse no sólo en función de las transformaciones, sino también con base en otras propiedades. Por ejemplo, un objeto primitivo puede ser una pirámide regular con un número (definido por el usuario) de caras que se unen en el ápice. Los ejemplares de primitivas son similares a objetos parametrizados, excepto que los objetos son sólidos.

Una primitiva parametrizada se puede considerar como la definición de una familia de partes cuyos miembros varían en unos cuantos parámetros, un importante concepto de CAD conocido como tecnología de grupos. La generación de ejemplares de primitivas se 'usa con frecuencia en objetos de complejidad relativa, como engranes o tuercas, que es tedioso definir en función de combinaciones booleanas de objetos más sencillos pero que se pueden caracterizar fácilmente con unos cuantos parámetros de alto nivel. Por ejemplo, un engrane se puede parametrizar por su diámetro o su número de dientes, como se ilustra en la figura 3.21.



Engrane
Diámetro = 4.3
Eje = 2.0
Grosor = 0.5
Dientes = 12
Agujero = 0.3

Engrane
Diámetro = 6.0
Eje = 1.0
Grosor = 0.4
Dientes = 18
Agujero = 0.3

Fig. 3.21.Engrane parametrizado

Aunque podemos construir una jerarquía de ejemplares de primitivas, cada ejemplar de nodo hoja sigue siendo un objeto definido por separado. En la generación de ejemplares de primitivas no hay medios para combinar objetos y crear uno nuevo de mayor nivel usando, por ejemplo, las operaciones booleanas regularizadas de conjuntos.

Por lo tanto, la única forma de crear un nuevo tipo de objeto es escribiendo el código que lo define. Así mismo, es necesario escribir individualmente, para cada primitiva, las rutinas que dibujan el objeto o determinan sus propiedades de masa.

Representaciones de barrido

Al barrer un objeto a lo largo de una trayectoria por el espacio se define un objeto nuevo, llamado barrido. El tipo de barrido más sencillo es el definido por un área bidimensional barrida por una trayectoria lineal normal al plano del área para crear un volumen. Este proceso se conoce como barrido traslacional o extrusión y es una forma natural de representar objetos formados por la extrusión de metal o plástico a través de un molde con la sección transversal deseada.

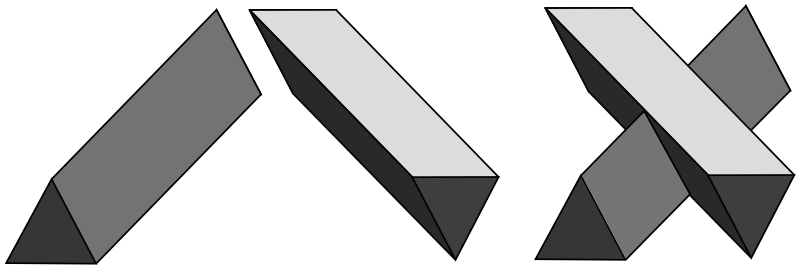
En los casos más sencillos cada volumen de barrido no es más que el área del objeto que se barre multiplicada por la longitud del barrido. Las extensiones sencillas comprenden

el escalamiento de la sección transversal durante el barrido para producir un objeto ahusado o barrer la sección transversal a lo largo de una trayectoria lineal que no es la normal. Los barridos rotacionales se definen mediante la rotación de un área con respecto a un eje.

El objeto que se barre no tiene que ser bidimensional. Los barridos de sólidos son útiles para modelar la región barrida por la cabeza de corte de una máquina herramienta o un robot que sigue una trayectoria. Los barridos donde el área o el volumen que generan cambia de tamaño, forma u orientación y que siguen una trayectoria curva arbitraria se denominan barridos generales.

Los barridos generales de secciones transversales bidimensionales se conocen como cilindros generalizados en visión computadorizada, y generalmente se modelan como secciones transversales bidimensionales parametrizadas barridas con ángulos rectos sobre una curva arbitraria. Los barridos generales son muy difíciles de modelar en forma eficiente. Por ejemplo, la trayectoria y la forma del objeto que producen pueden ocasionar que el objeto se interseque, complicando los cálculos de volumen. Además, los barridos generales no siempre generan sólidos. Por ejemplo, el barrido de un área bidimensional en su propio plano genera otra área bidimensional.

En términos generales, es difícil aplicar las operaciones regularizadas de conjuntos booleanos a los barridos sin antes convertirlos a otra representación. Incluso los barridos más simples no son cerrados con las operaciones booleanas regularizadas de conjuntos. Por ejemplo, al unión de dos barridos simples generalmente no es un barrido simple, como se ilustra en la figura 3.22.



(a) (b)

Fig. 3.22.(a) Dos barridos simples de objetos tridimensionales (triángulos). (b) La unión de los barridos representados en (a) no es un barrido simple de un objeto tridimensional.

Sin embargo, a pesar de los problemas de cerradura y de cálculo, los barridos son una forma natural e intuitiva de construir diversos objetos. Por ello, muchos sistemas de modelado de sólidos permiten a los usuarios construir objetos como barridos, pero los almacenan con alguna representación.

Representación de fronteras.

Las representaciones de fronteras (también conocidas como b-rep) describen un objeto en función de sus fronteras superficiales: vértices, áreas y caras. Algunas representaciones de fronteras están limitadas a fronteras poligonales planas y pueden requerir que las caras sean triángulos o polígonos convexos.

La determinación de lo que constituye una cara puede ser difícil si se permiten superficies curvas, como en la figura 3.23. Las caras curvas muchas veces se aproximan con polígonos. Alternativamente, las superficies curvas también se pueden representar como parches de superficie si los algoritmos que procesan la representación pueden tratar las curvas de intersección resultantes, las cuales por lo general, serán de orden mayor que las superficies originales.

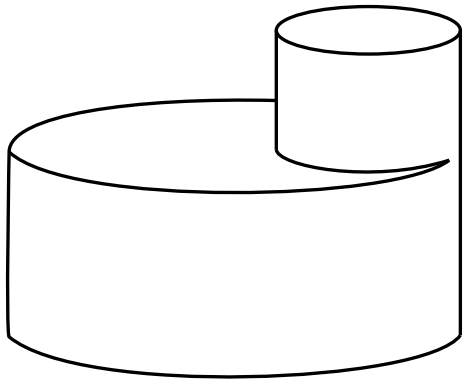
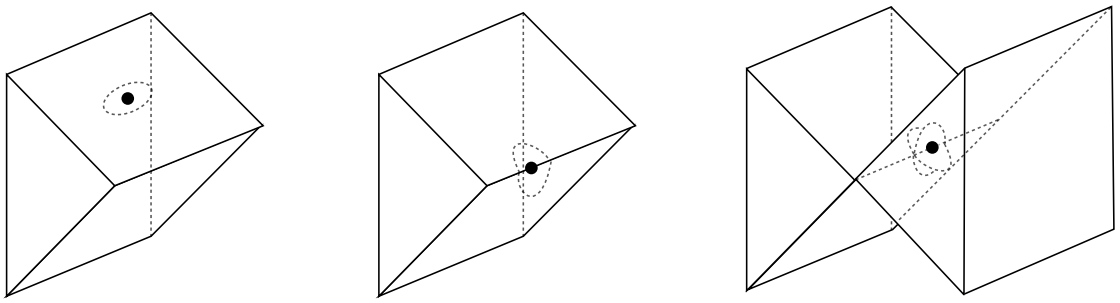


Fig. 3.23.¿Cuántas caras tiene este objeto?

Las representaciones de fronteras se obtienen de las sencillas representaciones vectoriales y se emplean en muchos sistemas de modelado actuales. Como son comunes en la graficación, se han desarrollado varias técninas eficientes para crear imágenes de objetos poligonales con sombreado suave.

Muchos sistemas dc representación de fronteras sólo permiten usar sólidos cuyas fronteras sean 2 - variedades. Por definición, cada punto en una 2-variedad tiene una

vecindad, arbitrariamente pequeña, de puntos que pueden considerarse topológicamente iguales a un disco en el plano. Esto quiere decir que hay una correspondencia continua uno a uno entre la vecindad de los puntos y el disco; como se ilustra en la figura 3.24(a) y (b).



(a) (b) (c)

Fig. 3.24.En una 2-variedad cada punto tiene una vecindad de puntos que lo rodean y forman un disco topológico, mostrado en gris (a) y (b). (c) Si un objeto no es una 2-variedad, entonces no tiene puntos cuya vecindad sea un disco topológico.

Por ejemplo, si más de dos caras comparten una arista, como en la figura 3.24(c), cualquier vecindad de un punto en la arista contiene puntos de cada una de esas caras. Por intuición, es obvio que no hay una correspondencia uno a uno continua entre esta vecindad y un disco en el plano, aunque la demostración matemática no es trivial.

Por lo tanto, la superficie en la figura 3.24(c) no es una 2-variedad. Aunque algunos sistemas actuales no tienen esta restricción, limitaremos nuestro análisis de las representaciones de fronteras a las 2-variedades, excepto si se indica lo contrario.

Poliedros y fórmula de Euler

Un poliedro es un sólido acotado por un conjunto dc polígonos cuyas aristas pertenecen a un número par de polígonos (exactamente dos, en el caso de las 2-variedades) y que satisfacen algunas restricciones adicionales (que veremos más adelante). Un poliedro simple es aquel que se puede deformar para obtener una esfera, es decir, un poliedro que, a diferencia de un toro, no tiene agujeros. La representación de frontera de un poliedro simple satisface la fórmula de Euler, que expresa una relación invariable entre el número de vértices. aristas y caras de un poliedro simple:

3.25. $$V - E + F = 2$$

donde V es el número de vértices, E es el número de aristas y F es el número de caras. En la figura 3.25 se presentan algunos poliedros simples y su número de vértices, aristas y caras. Observe que la fórmula aún es aplicable si se permiten aristas curvas y caras no planas. Por si sola, la fórmula de Euler establece condiciones necesarias, aunque no suficientes, para que un objeto sea un poliedro simple.

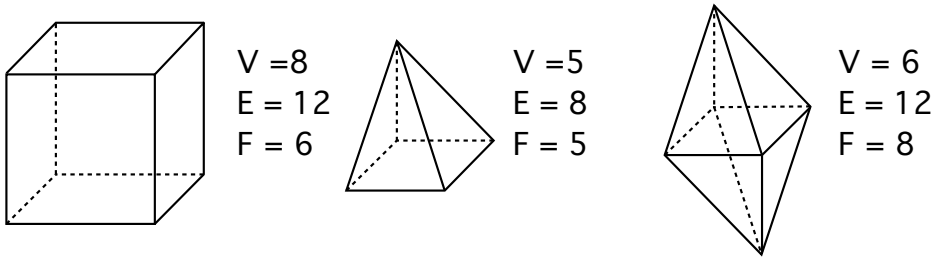


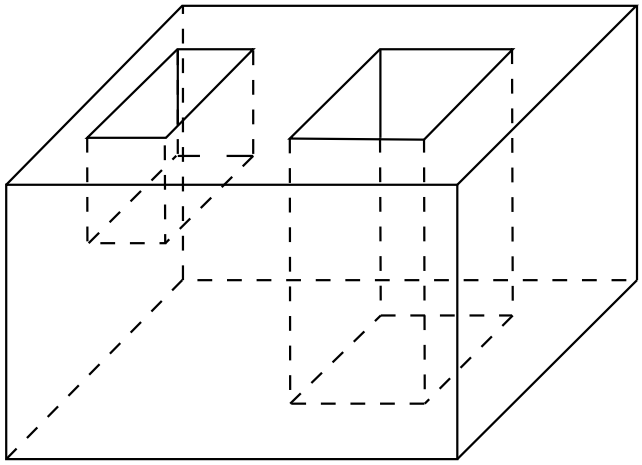
Fig. 3.25.Algunos poliedros simples con sus valores de vértices (V), aristas (E) y caras (F). En todos los casos, V - E + F =2

Se pueden construir objetos que satisfagan la fórmula pero no acoten un volumen, añadiendo una o más caras ó aristas colgantes a un sólido que de otra manera seria válido. Se requieren otras retricciones para garantizar que el objeto sea un sólido: cada arista debe conectar dos vértices y estar compartida por exactamente dos caras, al menos tres aristas deben unirse en cada vértice y las caras no deben ser interpenetrantes.

Una generalización de la fórmula de Euler se aplica a 2-variedades que tienen caras con agujeros:

3.26. $$V - E + F - H = 2(C - G)$$

donde H es el número de agujeros en las caras, G es el número de agujeros que atraviesan el objeto y C es el número de componentes separados (partes) del objeto, como se muestra en la figura 3.26. Si un objeto tiene un solo componente, su G se conoce como género; si tiene varios componentes, entonces su G es la suma de los géneros de sus componentes. Como antes, se necesitan restricciones adicionales para garantizar que los objetos sean sólidos.



V - E + F - H = 2(C - G)
24 36 15 3 1 1

Fig. 3.26.Poliedro clasificado de acuerdo con la ecuación 3.26,
con dos agujeros en la cara superior y uno en la cara inferior.

Baumgart introdujo el concepto de operadores de Euler, los cuales operan con objetos que satisfacen la fórmula de Euler y transforman los objetos para formar otros que también obedecen a la fórmula, añadiendo vértices, aristas y caras. Braid, Hillyard y Stroud mostraron cómo se puede combinar un pequeño número de operadores de Euler para transformar objetos, siempre y cuando no se requiera que los objetos intermedios sean sólidos válidos, mientras que Mäntylä demuestra que todas las representaciones de fronteras válidas se pueden construir con una secuencia finita de operadores de Euler. Se pueden definir también otros operadores que realizan ajustes finos a un objeto, moviendo los vértices, aristas o caras existentes sin afectar su número.

Quizás la representación de frontera más simple sea una lista de caras poligonales, cada una representada por una lista de coordenadas de vértices. Para representar la dirección hacia la cual queda la cara, los vértices del polígono se listan en un orden acorde al sentido del giro de las manecillas del reloj, visto desde el exterior del sólido.

Para evitar la duplicación de coordenadas compartidas por las caras, cada vértice se puede representar con un índice en una lista de coordenadas. En esta representación, las aristas se representan implícitamente con pares de vértices adyacentes en las listas de vértices de los polígonos. Las aristas también se pueden representar en forma explícita como pares de vértices, definiendo cada cara como una lista de índices en la lista de vértices.

Operaciones de conjuntos booleanos

Las representaciones de fronteras se pueden combinar usando los operadores regularizados de con conjuntos booleanos para crear así nuevas representaciones de fronteras. Algunos autores analizan algoritmos para determinar las intersecciones entre superficies cuádricas.

Un método consiste en inspeccionar los polígonos de ambos objetos, dividiéndolos de ser necesario para asegurar que la intersección de un vértice, arista o cara de ambos. Después se clasifican los polígonos de cada objeto con respecto al otro para determinar si están dentro, fuera o en la frontera.

Si nos remitimos a la tabla de la figura 3.19. Observamos, que por tratarse de una representación de frontera, sólo nos interesan las seis últimas filas que representan una parte de una o ambas fronteras de los objetos originales, Ab y Bb. Después de la división, cada polígono de un objeto está totalmente dentro (Ab ∩ Bi ó Bb ∩ Ai) o fuera (Ab − B o Bb − A)del otro, o bien forma parte de la frontera compartida (Ab ∩ Bb iguales o Ab ∩ Bb diferentes).

Podemos clasificar el polígono construyendo un vector en la dirección de la normal a la superficie del polígono, desde un punto en el interior del polígono, para luego hallar el polígono más cercano que interseca el vector en el otro objeto. Si no se interseca ningún polígono, el polígono original se encuentra fuera del otro objeto. Si el polígono intersecante más cercano es coplanar con el original, esta intersección es de tipo frontera-frontera y la comparación de las normales de los polígonos indica el tipo de intersección (Ab ∩ Bb iguales o Ab ∩ Bb diferentes).

En caso contrario, se revisa el producto punto de las normales de los dos polígonos. Un producto punto positivo indica que el polígono original, se halla dentro del otro objeto; si el resultado es negativo, el polígono original está fuera. El producto punto igual a cero puede ocurrir si el vector se encuentra en el plano del polígono intersecado; en este caso, el vector se perturba ligeramente e interseca de nuevo con los polígonos del otro objeto.

Es posible usar la información de adyacencia de vértices para evitar el procesa miento adicional necesario para clasificar los polígonos de esta manera. Si un polígono es adyacente

a un polígono clasificado (o sea, comparte vértices con él) y no tiene contacto con la superficie del otro objeto, se asigna al polígono la misma clasificación. Todos los vértices en la frontera común entre objetos se pueden marcar durante la fase inicial de división de polígonos. Podemos determinar si un polígono entra en contacto con la superficie del otro revisando si tiene vértices de frontera.

La clasificación de cada polígono determina si se conserva o descarta el polígono durante la operación de creación del objeto compuesto. Por ejemplo, al formar la unión, la operación descarta todo polígono que pertenezca a un objeto que esté dentro del otro. La operación conserva los polígonos de cualquiera de los objetos que no se encuentre dentro del otro, excepto en el caso de los polígonos coplanares.

Los polígonos coplanares se descartan si tienen normales a la superficie opuestas, y sólo se conserva uno de los dos si estas direcciones son iguales. Es importante la decisión de cuál polígono conservar si los objetos están formados por materiales diferentes.

Aunque A ∪* B tiene el mismo significado geométrico que B ∪* A, en este caso las dos operaciones pueden tener resultados visiblemente distintos, de manera que la operación se pueda definir para favorecer a uno de los operandos en el caso de los polígonos coplanares.

Representaciones de partición espacial

En las representaciones de partición espacial, un sólido se descompone en una colección de sólidos adjuntos, no intersecantes, que son mas primitivos que el sólido original, aunque no necesariamente del mismo tipo. Las primitivas pueden variar en tipo, tamaño, posición, parametrización y orientación, casi como las piezas de un juego de bloques de construcción para niños. El alcance de la descomposición de los objetos depende de cuán primitivos deben ser los sólidos para poder efectuar con facilidad las operaciones que nos interesan.

Descomposición en celdas

Una de las formas más generales de la partición espacial se denomina descomposición en celdas. Cada sistema de descomposición en celdas define un conjunto de celdas primitivas que por lo general se parametrizan y con frecuencia son curvas. La

descomposición en celdas difiere de la generación de ejemplares de primitivas en que se pueden componer objetos más complejos a partir de otros más sencillos y primitivos, "pegándolos" en forma ascendente.

La operación de pegado se puede considerar como una forma restringida de la unión, en la cual los objetos no pueden intersecarse . Otras restricciones que se aplican al pegado de objetos pueden requerir que dos celdas compartan un punto, arista o cara.

Aunque la representación de descomposición en celdas no es ambigua, es posible que no sea única, como se ilustra en la figura 3.27 La descomposición en celdas también es difícil de validar, ya que en potencia hay que evaluar cada par de celdas para determinar si hay intersecciones. No obstante, la descomposición en celdas es una representación importante para el análisis de elementos finitos.

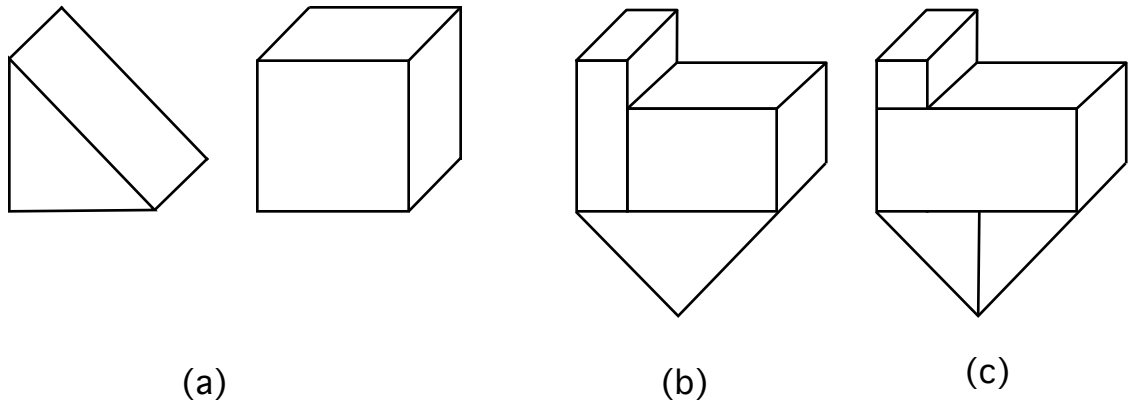


Fig. 3.27.Las celdas representadas en (a) se pueden transformar de distintas maneras para construir el mismo objeto que aparece en (b) y (c). Incluso un solo tipo de celda es suficiente para ocasionar ambigüedades.

Enumeración de ocupación espacial

La enumeración de ocupación espacial es un caso especial de la descomposición en celdas, en la cual el sólido se descompone en celdas idénticas dispuestas sobre una lilaila regular lija, Estas celdas con frecuencia se denominan elementos de volumen (o voxeles, por analogía con los pixeles). En la figura 3.28 se muestra un objeto representado con enumeración de ocupación espacial.

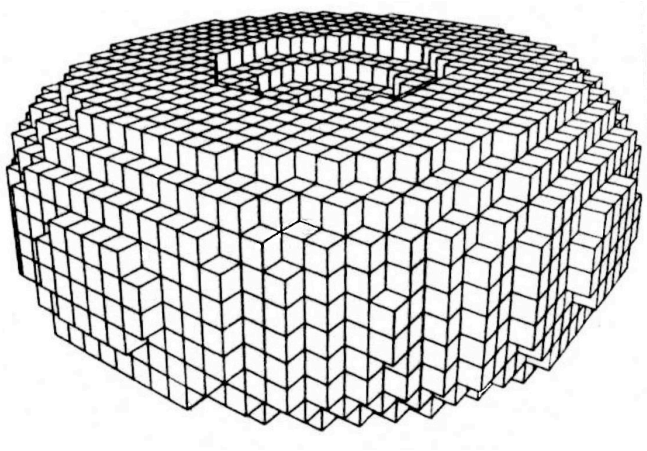


Fig. 3.28.Toro representado por numeración de ocupación espacial.

El tipo de celda más común es el cubo, y la representación del espacio como un arreglo regular de cubos se denomina cuberil. Cuando se representa un objeto con enumeración de ocupación espacial, sólo se controla la presencia o la ausencia de una celda en cada posición de la malla. De esta manera, un objeto se puede codificar con una lista única y no ambigua de celdas ocupadas.

Es más sencillo determinar si una celda está dentro o fuera del sólido, y también si dos objetos son adyacentes. La enumeración de ocupación espacial se usa con frecuencia en aplicaciones biomédicas para representar datos volumétricos obtenidos de fuentes como rastreo de tomografía axial computadorizada (CAT, computerized axial tomography).

No obstante todas sus ventajas, la enumeración de ocupación espacial tiene varias deficiencias obvias, similares a las de la representación de una imagen bidimensional con un arreglo bidimensional de un bit de profundidad. No existe el concepto de ocupación parcial; por lo tanto, varios sólidos, como el toro de la figura 3.28 sólo pueden aproximarse.

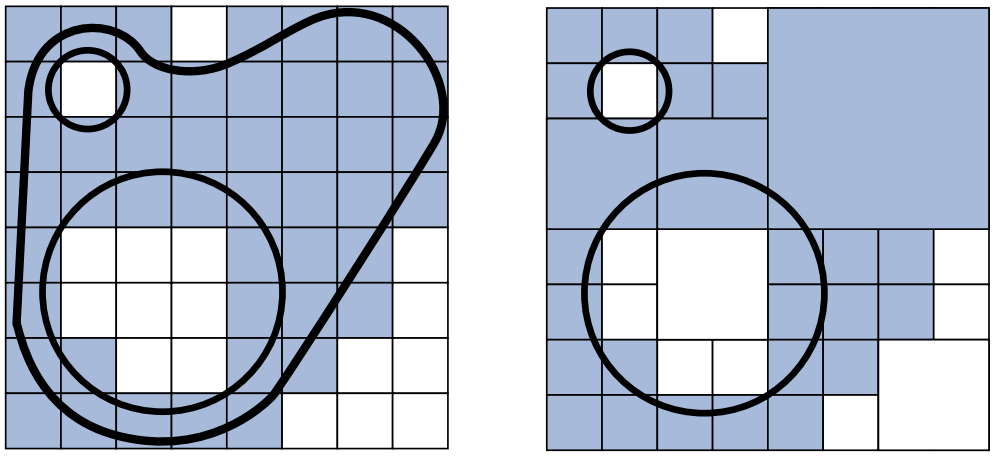
Si las celdas son cubos, los únicos objetos que se pueden representar con exactitud son aquellos cuyas caras son paralelas a los lados de cubo y cuyos vértices corresponden a la malla. Las celdas, como los pixeles en un arreglo bidimensional de bits, pueden en principio ser tan pequeñas como se desee para aumentar la precisión de la representación.

Sin embargo, el espacio se convierte en un asunto de importancia, ya que se requieren hasta n^3 celdas ocupadas para representar un objeto con una resolución de n elementos de volumen en cada una de las tres dimensiones,

Árboles de octantes

Los árboles de octantes (octrees) constituyen una variante jerárquica de la enumeración de ocupación espacial diseñada para considerar los exigentes requisitos de almacenamiento del método. Los árboles de octantes se derivan de los árboles de cuadrante (quadtrees), un formato de representación bidimensional que se utiliza para codificar imágenes.

La idea que sustenta los árboles de cuadrantes y de octantes es el poder "divide y vencerás" de la subdivisión binaria. Un árbol de cuadrantes se forma dividiendo sucesivamente un plano bidimensional en ambas direcciones para formar cuadrantes, como se ilustra en la figura 3.29.



(a) (b)

Fig. 3.29.Objeto representado con (a) enumeración de ocupación espacial para obtener (b) un árbol de cuadrantes.

Cuando se emplea un árbol de cuadrante para representar un área en un plano, cada cuadrante puede estar lleno, parcialmente lleno o vacío (también llamado negro, gris o blanco respectivamente), dependiendo de la cantidad de área que interseque el cuadrante. Un cuadrante parcialmente lleno se subdivide de manera recursiva en subcuadrantes. La subdivisión continúa hasta que todos los cuadrante son homogéneos (llenos o vacíos) o hasta alcanzar una profundidad límite previamente determinada.

Cuando cuatro cuadrantes hermanos están llenos o vacíos, se eliminan y su padre parcialmente lleno se reemplaza con un nodo totalmente lleno o vacío. (Se puede utilizar en

cambio un enfoque ascendente (bottom-up) para evitar este problema de eliminación y fusión.

En la figura 3.29, cualquier nodo parcialmente lleno en la profundidad límite se clasifica como lleno, Las subdivisiones sucesivas se pueden representar como un árbol con cuadrantes parcialmente llenos en los nodos internos y cuadrantes llenos o vacíos en las hojas, como se presenta en la figura 3.30.

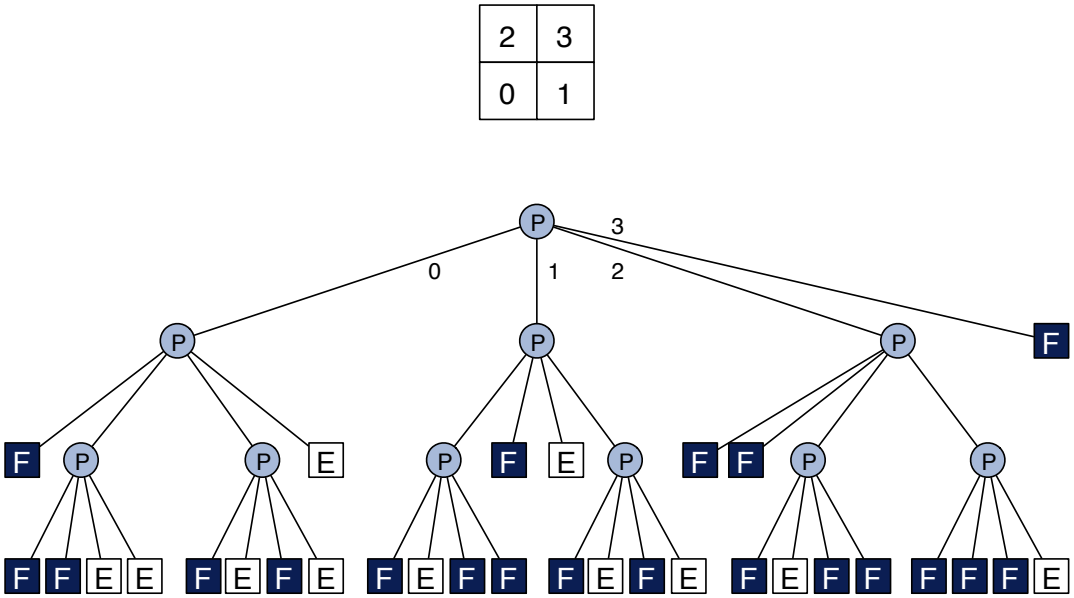


Fig. 3.30.Estructura de árbol de cuadrantes para el objeto de la figura 3.29. F = lleno, P = parcialmente lleno, E = vacío.

Si relajamos los criterios para clasificar un nodo como homogéneo, de manera que los nodos que se encuentren encima o debajo de cierto límite se puedan clasificar como llenos o vacíos, la representación será más compacta, aunque menos precisa. El árbol de octantes es similar al de cuadrantes, excepto que en sus tres dimensiones se subdivide para obtener octantes, como se ilustra en la figura 3.31.

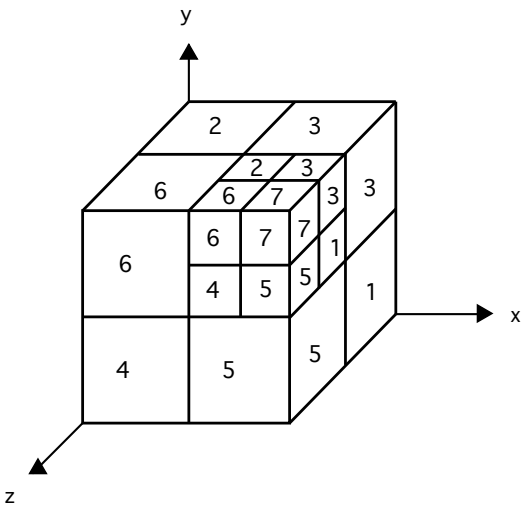


Fig. 3.31.Numeración de árbol de octantes. El octante cero no es visible.

Los cuadrantes generalmente se indican con los números 0 a 3 y los octantes con los números 0 a 7. Como no se ha desarrollado ningún mecanismo estándar para la numeración, también se emplean nombres mnemónicos. Los cuadrantes se nombran de acuerdo con su dirección de brújula respecto al centro de su padre: NO, NE, SO y SE (o NW, NE, SW y SE, en inglés.

Los octantes se nombran e forma similar, distinguiendo entre la izquierda (L) y la derecha (R), arriba (U) y abajo (D), y anterior (F) y posterior (B): LUF, LUB, LDF, LDB, RUF, RUB, RDF y RDB.

Con excepción de los peores casos, se puede demostrar que el número de nodos en una representación de árbol de cuadrantes u octantes es proporcional al perímetro o a la superficie del objeto, respectivamente. Esta relación existe porque la subdivisión de nodos surge exclusivamente por la necesidad de representar la frontera del objeto que se codifica. Los únicos nodos internos que se dividen, so aquellos por los cuales pasa una parte de la frontera, Por ende, cualquier operación con una de estas estructuras de datos que sea lineal en el número de nodos que contiene también se ejecuta en un tiempo proporcional al tamaño del perímetro o área.

Transformaciones y operaciones booleanas de conjuntos

Se ha efectuado mucho trabajo en el desarrollo de algoritmos eficaces para almacenar y procesar árboles de cuadrantes y octantes. Por ejemplo, las operaciones booleanas de conjuntos tienen una aplicación bastante directa en árboles tanto de cuadrantes como de octantes. Para calcular la unión o intersección U de dos árboles, S y T, se efectúa un recorrido paralelo descendente por los dos árboles.

En la figura 3.32 se muestran las operaciones para los árboles de cuadrantes; la generalización a los árboles de octantes es bastante sencilla. Se examina cada par corespondiente de nodos. Considere el caso de la unión. Si alguno de los nodos en el par es azul oscuro, se agrega un nodo negro correspondiente a U. Si uno de los nodos del par es blanco, se crea en U el nodo correspondiente con el valor del otro nodo en el par.

Si ambos nodos en el par son azul claro, se añade un nodo azul claro a U y se aplica recursivamente el algoritmo a los hijos del par. En este caso hay que inspeccionar los hijos del nuevo nodo en U después de aplicar el algoritmo. Si todos son azul oscuro, se eliminan y su padre en U cambia de azul claro a azul oscuro.

Es fácil efectuar transformaciones sencillas con los árboles de cuadrantes y de octantes. Por ejemplo, la rotación sobre un eje múltiplos de 90° se logra rotando recursivamente los hijos en cada nivel.

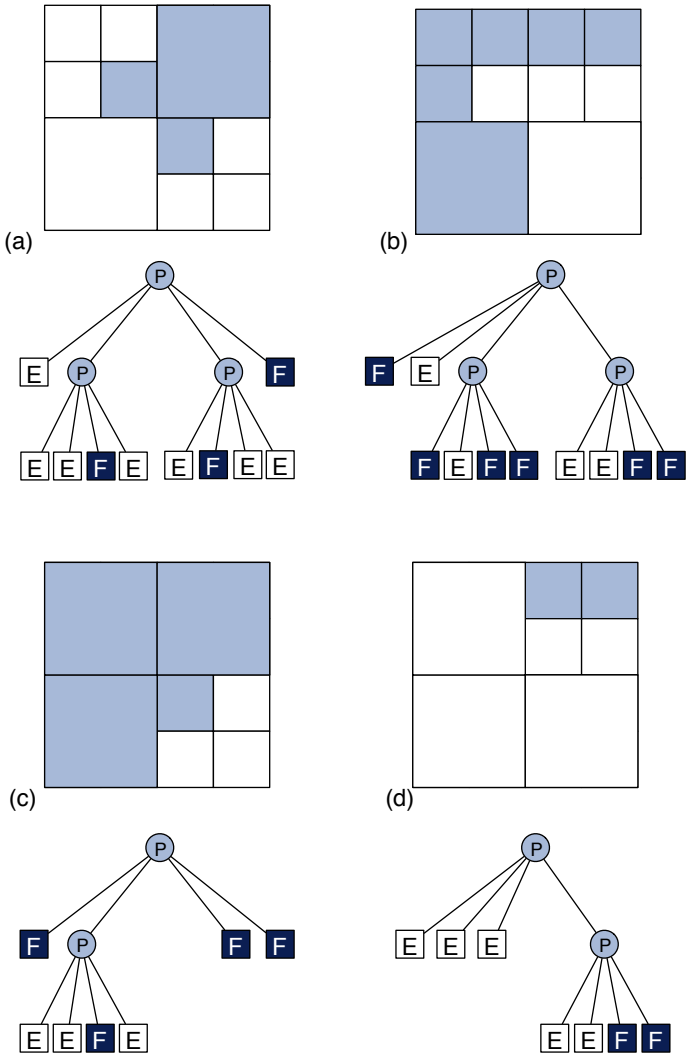


Fig. 3.32.Operaciones booleanas de conjuntos con árboles de cuadrantes. (a) El objeto S y su árbol de cuadrantes. (b) El objeto T y su árbol de cuadrantes. (c) S ∪ T. (d) S ∩ T. (F = lleno, P = parcialmente lleno, E = vacío).

El escalamiento por potencias de 2 y las reflexiones también son bastante sencillos. Las traslaciones pueden ser mas complejas, al igual que las transformaciones generales. Así mismo, como ocurre por lo general en la numeración de ocupación espacial, el problema del artefacto de discretización en las transformaciones generales es serio.

Detección de vecinos

Una operación importante en los árboles de cuadrantes y octantes es detectar el vecino de un nodo, es decir, hallar un nodo que se adyacente al original (que comparta una

cara, una arista o un vértice) y cuyo tamaño sea igual o mayor. Un nodo de árbol de cuadrantes tiene vecinos en ocho direcciones posibles. Sus vecinos en N, S, E y O lo son sobre un arista común, mientras que sus vecinos en No, Ne, So y Se lo son con respecto a un vértice común. Un nodo de árbol de octantes tiene vecinos en 26 direcciones posibles: 6 vecinos sobre una cara, 12 sobre una arista y 8 sobre un vértice.

Árboles binarios de partición de espacio

Los árboles de octantes dividen recursivamente el espacio en planos que siempre son mutuamente perpendiculares y que bisecan las tres dimensiones de cada nivel del árbol. En cambio, los arboles binarios de partición de espacio (BSP, binary space-partitioning trees) dividen recursivamente el espacio n pares de subespacios, cada uno separado un plano de orientación y posición arbitrarias.

La estructura de datos de árbol binario que se crea se utilizó orginalmente en la determinación de superficies visibles de gráficos. Después se introdujo la utilización de los árboles BSp para representar poliedros arbitrarios. Cada nodo interno del árbol BSP está relacionado un un plano y tiene dos apuntadores a hijos, uno para cada lado del plano. Suponiendo que las normales apunten hacia afuera del objeto, el hijo de la izquierda está detrás o dentro del plano, mientras que el de la derecha se encuentra frente o fuera del plano.

Si se divide aún más el medio espacio a un lado del plano, entonces su hijo es la raíz de un subarbol; si el medio espacio es homogéneo, entonces el hijo es una hoja y representa una región que esta completamente dentro o fuera del poliedro. Estas regiones homogéneas se denominan celdas dentro y fuera. Para considerar la presición numérica limitada con la cual se realizan las operaciones, cada nodo tiene un grosor relacionado con su plano. Cualquier punto que se encuentre dentro de esta tolerancia del plano se considera dentro del plano.

El concepto de subdivisión que sirve de base a los arboles BSP, como el de los árboles de cuadrantes y octantes, es independiente de la dimensión. De esta manera en la figura 3.33(a) se muestra un polígono cóncavo en dos dimensiones, acotado por líneas negras. Las celdas dentro se sombrean en color gris claro y las líneas que definen los medios

espacios aparecen en gris oscuro, con las normales apuntando hacia afuera. El árbol BSP correspondiente se muestra en la figura 3.33(b).

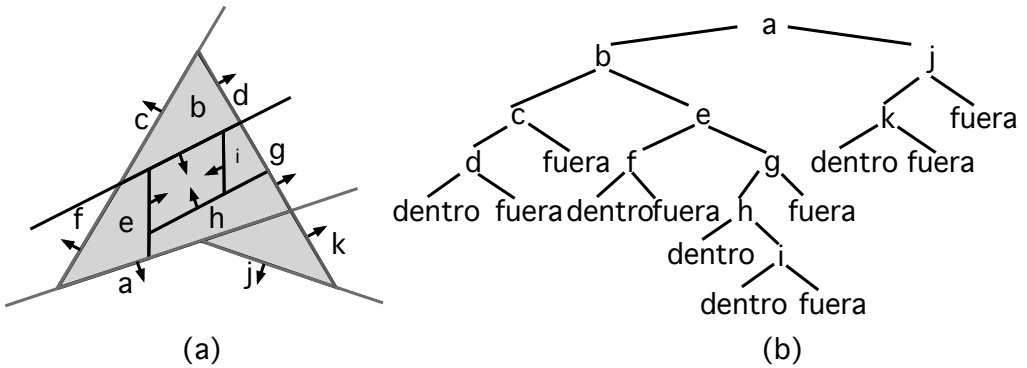


Fig. 3.33.Representación de árbol BSP en dos dimensiones (a) Polígono cóncavo acotado por líneas negras. Las líneas que definen los medios espacios están en gris oscuro y las celdas dentro están en gris claro. (b) Árbol BSP.

En dos dimensiones, las regiones dentro y fuera forman un teselado poligonal convexo del plano; en tres dimensiones, estas regiones forman un teselado poliédrico convexo del espacio tridimensional. Por lo tanto un árbol BSP puede representar un sólido cóncavo arbitrario con agujeros como la unión de regiones dentro convexas.

Considere la tarea de determinar su un punto está dentro, fuera o sobre un sólido, un problema conocido como clasificación de puntos. Se puede usar un árbol BSP para clasificar un punto, filtrando el punto por el árbol a partir de la raíz. En cada nodo se sustituye el punto en la ecuación del plano del nodo y se pasa recursivamente al hijo de la derecha si está enfrente (fuera) del plano. Si el nodo es una hoja, el punto está indicado por el valor de la hoja, ya sea como dentro o fuera. Si el punto se halla en el plano de un nodo, se pasa a ambos hijos y se comparan las clasificaciones. De ser iguales, el punto recibe ese valor; cuando son diferentes, el punto se ubica en la frontera entre las regiones fuera y dentro y se clasifica como sobre.

Este método se puede extender a la clasificación de líneas y polígonos. Sin embargo, a diferencia de los puntos, una línea o un polígono puede estar parcialmente en ambos lados de plano. Por consiguiente, en cada nodo cuyo plano interseca la línea o el polígono, hay que dividir (recortar) la línea o el polígono y obtener partes que estén enfrente, detrás o sobre el plano, para luego clasificar las partes por separado.

Aunque los árboles BSP constituyen una forma de representación sencilla y elegante, los polígonos se subdividen conforme se construye el árbol y al efectuar las operaciones de conjuntos booleanos, por lo cual la notación quizás no sea tan compacta como otras representaciones. Sin embargo, si se aprovecha la inherente independencia de la dimensión del árbol BSP, es posible desarrollar un álgebra booleana cerrada para árboles BSP tridimensionales que se base recursivamente en las representaciones de polígonos como árboles bidimensionales, aristas como árboles unidimensionales y puntos como árboles de cero dimensiones.

Geometría sólida constructiva

En la geometría sólida constructiva (CSG, constructive solid geometry), las primitivas simples se combinan a través de operadores boolenanos regularizados de conjuntos que se incluyen directamente en la representación. Un objeto se almacena como un árbol con operadores en los nodos internos y primitivas simples en las hojas (Fig. 3.34). Algunos nodos representan operadores booleanos mientras que otros llevan a cabo traslaciones, rotaciones y escalamientos. Como las operaciones booleanas por lo general no son conmutativas, las aristas del árbol están ordenadas.

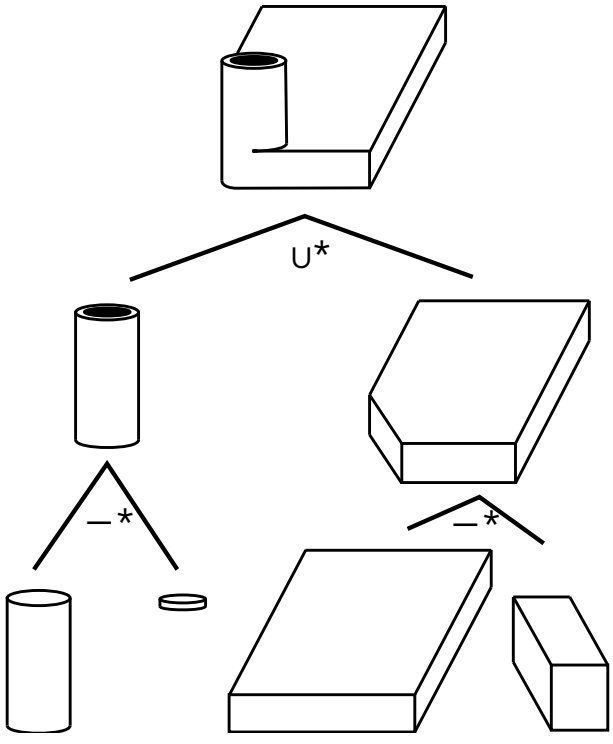


Fig. 3.34.Objeto definido por CSG y su árbol

Para determinar las propiedades físicas o para crear imágenes, debemos ser capaces de combinar las propiedades de las hojas para obtener las propiedades de la raíz. La estrategia general del procesamiento es recorrido por profundidades a lo largo del árbol para combinar los nodos a partir de la hoja.

La complejidad de esta tara depende de la representación de los objetos y hoja y de si debe producirse realmente una representación completa del objeto compuesto en la raíz del árbol. Por ejemplo, los algoritmos de operaciones de conjuntos booleanos para la representación de fronteras analizados previamente, combinan las representaciones de fronteras de dos nodos para crear una tercera representación de frontera y son difíciles de implantar. El algoritmo CSG es mucho más sencillo y produce una imagen procesando las representaciones de las hojas sin combinarlas de manera explícita.

En algunas implantaciones, las primitivas son sólidos simples, como cubos o esferas, ara asegurar que todas las combinaciones regularizadas también sean sólidos válidos. En otros sistemas, las primitivas incluyen medios espacios que en sí no son sólidos acotados. Por ejemplo, un cubo se puede definir como la intersección de seis medios espacios, o un cilindro finito como un cilindro infinito cerrado en las partes superior e inferior por un medio espacio plano.

La utilización de medios espacios introduce un problema de validez ya que no todas las combinaciones producen sólidos. Sin embargo, los medios espacios son útiles para operaciones como el rebanado de un objeto por un plano, operación que de otra manera se llevaría a cabo usando la cara de otro objeto sólido. Si no se emplean medios planos surge la necesidad de tiempo de procesamiento adicional, ya que las operaciones regularizadas de conjuntos booleanos deben realizarse con el objeto completo durante la división, incluso si sólo hay que cortar una cara.

Podemos considerar la descomposición en celdas y la numeración de ocupación espacial como casos especiales de la CSG en los cuales el único operador es el operador implícito de pegado: la unión de dos objetos que se pueden tocar pero que deben tener interiores disjuntos (es decir, los objetos deben tener una intersección booleana regularizada nula).

La CSG no ofrece una representación única. Esta característica puede ser muy confusa en un sistema que permite al usuario manipular los objetos hoja con operaciones de ajuste

fino. Al aplicar la misma operación a dos objetos que son inicialmente iguales se pueden generar resultados diferentes, como se muestra en la figura 3.35. No obstante, la capacidad de editar modelos a través de la eliminación, adición, reemplazo y modificación de subárboles, aunada a la forma relativamente compacta de almacenamiento de los modelos, han hecho de CSG una de las formas predominantes para el modelado de sólidos.

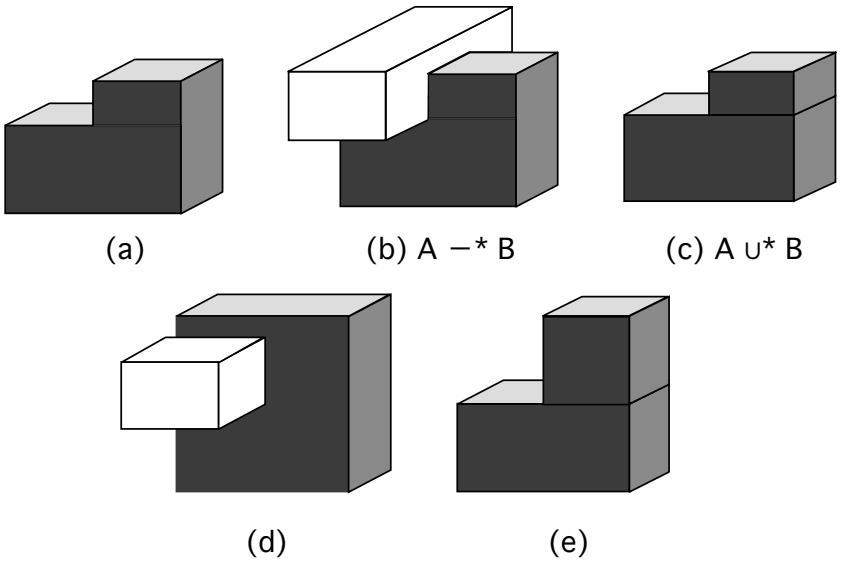


Fig. 3.35.El objeto presentado en (a) se puede definir con varias operaciones CSG, como se ilustra en (b) y (c). El ajuste fino hacia arriba de la cara superior (b) y (c) produce objetos diferentes, presentados en (d) y (e).

3.2.Proyecciones

En términos generales, las proyecciones transforman puntos en un sistema de coordenadas de dimensión n a puntos en un sistema de coordenadas con dimensión menor que n. De hecho, durante mucho tiempo se ha usado la graficación por computador para estudiar objetos n-dimensionales por medio de su proyección sobre dos dimensiones. Aquí nos limitaremos a la proyección de tres dimensiones a dos.

La proyección de objetos tridimensionales es definida por rayos de proyección rectos, llamados proyectores, que emanan de un centro de proyección, pasan por cada punto del objeto e intersecan un plano de proyección para formar la proyección. Por lo general, el centro de proyección se encuentra a una distancia finita del plano de proyección. Sin

embargo, en algunos tipos de proyecciones es conveniente pensar en función de un centro de proyección que tienda a estar infinitamente lejos.

En la figura 3.36 se presentan dos proyecciones diferentes de la misma línea. Afortunadamente, la proyección de una línea es en sí una línea, de manera que sólo hay que proyectar los puntos extremos. La clase de proyecciones que trataremos aquí se conoce como proyecciones geométricas planas, ya que la proyección es sobre un plano y no sobre una superficie curva y porque usa proyectores rectos y no curvos. Varias proyecciones cartográficas son no planas o no geométricas.

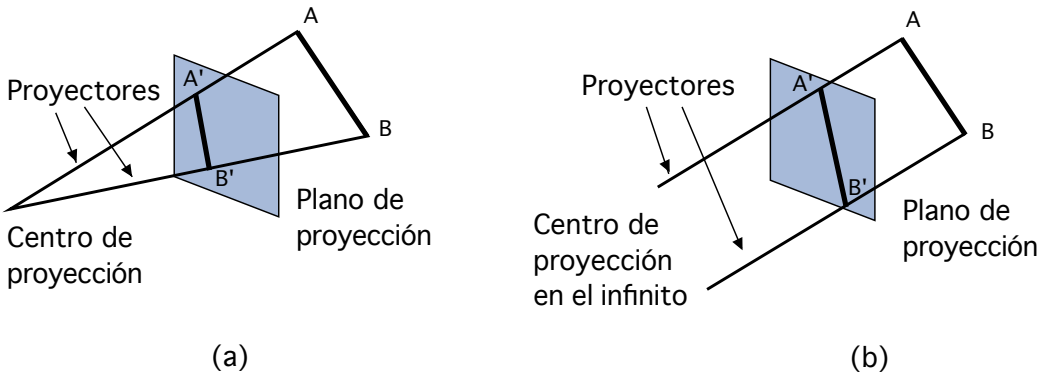


Fig. 3.36.Dos proyecciones diferentes de la misma línea. (a) Línea AB y su proyección de perspectiva A'B. (b) Línea AB y su proyección paralela AB. Los proyectores AA' y BB' son paralelos.

Las proyecciones geométricas planas, que a partir de ahora llamaremos simplemente proyecciones, se pueden dividir en dos clases básicas: de perspectiva y paralelas. La diferencia se debe a la relación entre el centro de proyección y el plano de proyección. Si la distancia entre uno y otro es finita, la proyección es de perspectiva; conforme se aleja el centro de proyección, los proyectores que pasan por un objeto tienden cada vez más a ser- paralelos. En la figura 3.36 se ilustran los dos casos.

La proyección paralela recibe ese nombre porque los proyectores son paralelos si el centro de proyección se encuentra a una distancia infinita. Al definir una proyección de perspectiva se especifica explícitamente su centro de proyección; en el caso de una proyección paralela, se indica su dirección de proyección. El centro de proyección es un punto, por lo cual tiene coordenadas homogéneas de la forma (*x, y, z, 1*). Como la dirección de proyección es un vector (es decir, la diferencia entre dos puntos), lo podemos calcular res- tando los dos puntos $d = (x, y, z, 1) - (x', y', z', 1) = (a, b, c, 0)$.

Por lo tanto, las direcciones y los puntos en el infinito tienen una correspondencia natural. En el límite, una proyección de perspectiva cuyo centro de proyección tienda a un punto en el infinito se convierte en una proyección paralela,

El efecto visual de una proyección de perspectiva es similar al de los sistemas fotográficos y al del sistema visual humano y se conoce como reducción frontal de perspectiva. El tamaño de la proyección de perspectiva de un objeto varía inversamente con la distancia entre el objeto y el centro de proyección.

Por lo tanto, aunque la proyección de perspectiva de los objetos tiende a parecer realista, no es muy útil para registrar la forma y las medidas exactas de los objetos; no se pueden tomar las distancias de la proyección, los ángulos sólo se conservan en las caras de los objetos paralelos al plano de proyección y por lo general las líneas paralelas no se proyectan como tales.

La proyección paralela es una vista menos realista porque no existe la reducción frontal de perspectiva, aunque pueden existir diversas reducciones frontales constantes sobre cada eje. La proyección se puede usar para mediciones exactas, y las líneas paralelas permanecen como tales. Como sucede en la proyección de perspectiva, los ángulos únicamente se conservan en las caras de los objetos paralelos al plano de proyección.

Proyección de perspectiva

Las proyecciones de perspectiva de cualquier conjunto de líneas paralelas que no sean paralelas al plano de conversión convergen en un punto de fuga. En el espacio tridimensional, las líneas paralelas sólo se unen en el infinito, de manera que el punto de fuga se puede considerar como la proyección de un punto en el infinito.

Por supuesto, hay una infinidad de puntos de fuga, uno para cada una de la infinidad de direcciones en que puede orientarse una línea.

Si el conjunto de líneas es paralelo a uno de lo, tres ejes principales, el punto de fuga se conoce como punto de fuga de eje. A lo sumo hay tres de estos puntos, correspondientes al número de ejes principales cortados por el plano de proyección. Por ejemplo, si el plano de proyección sólo corta el eje *z* (y por consiguiente es normal a él),

sólo el eje z tendrá un punto de fuga principal, ya que las líneas paralelas a los ejes *y* o *x* también serán paralelas al plano de proyección y no tendrán puntos de fuga.

Las proyecciones de perspectiva se clasifican de acuerdo con el número de puntos de fuga principales y por ende con respecto al número de ejes que corta el plano de proyección. En la figura 3.37 se muestran dos proyecciones de perspectiva de un punto para un cubo. Es obvio que hay más proyecciones de un punto, ya que las líneas paralelas a los ejes *x* y *y* no convergen; esto sólo ocurre con las líneas paralelas al eje *z*.

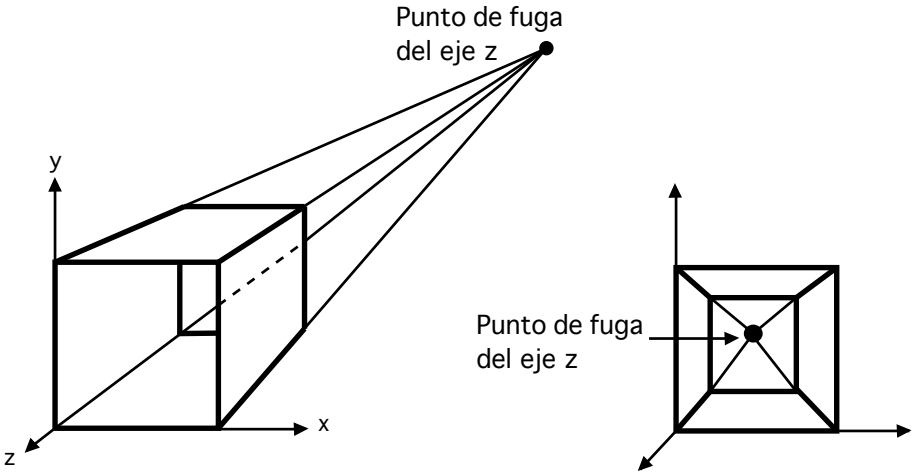


Fig. 3.37.Proyecciones de perspectiva de un punto de un cubo sobre un plano que corta el eje z, mostrando el punto de fuga de las líneas perpendiculares al plano de proyección.

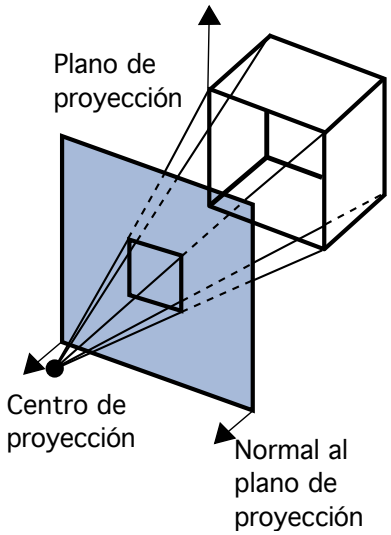


Fig. 3.38.Construcción de una proyección de perspectiva de un punto para un cubo sobre el plano que corta el eje z. La normal al plano de proyección es paralela al eje z.

En la figura 3.38 se presenta la construcción de una perspectiva de un punto con algunos de los proyectores y con un plano de proyección que sólo corta el eje *z*.

En la figura 3.39 se muestra la construcción de una perspectiva de dos puntos. Observe que las líneas paralelas al eje y no convergen en la proyección. La perspectiva de dos puntos se usa comúnmente en dibujos de arquitectura, ingeniería, diseño industrial y publicidad. Las perspectivas de tres puntos son menos frecuentes,ya que añaden muy poco al realismo que se puede obtener con la perspectiva de dos puntos.

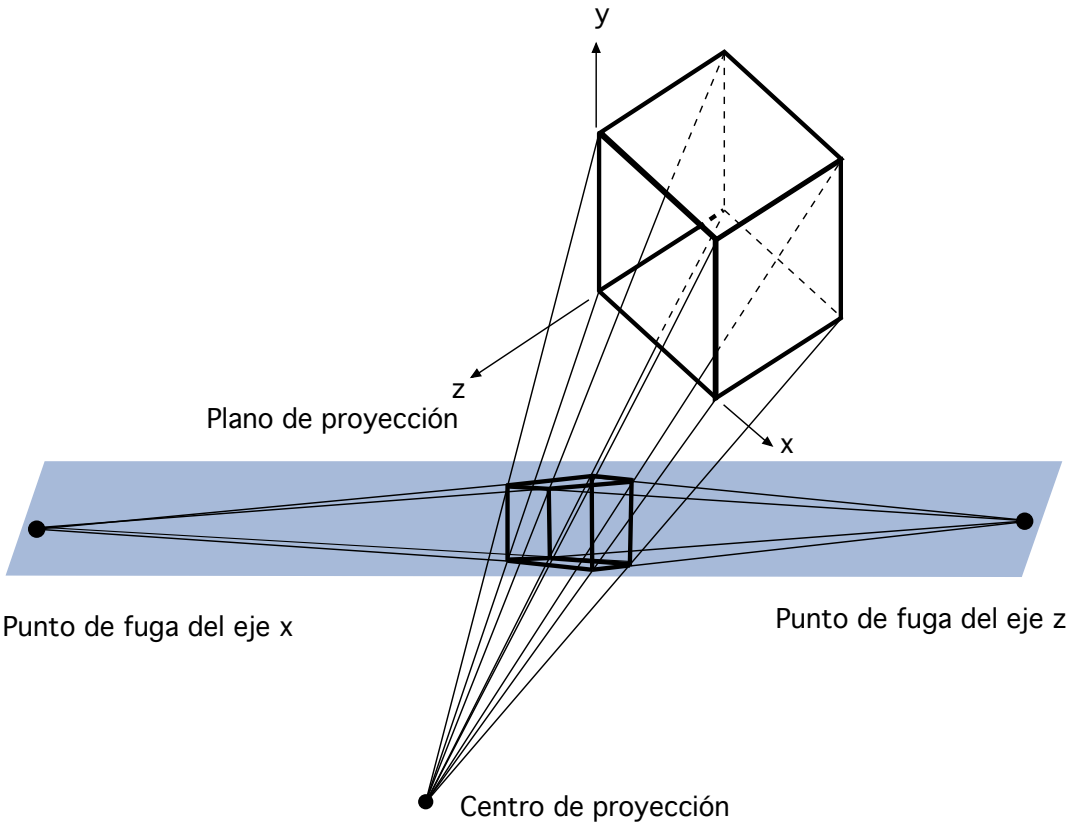


Fig. 3.39.Proyección de perspectiva de dos puntos en un cubo. El plano de proyección corta los ejes x y z.

Proyecciones paralelas

Las proyecciones paralelas se clasifican en dos tipos, dependiendo de la relación entre la dirección de la proyección y la normal al plano de proyección. En las proyecciones paralelas ortográficas, estas direcciones son las mismas (o en sentido contrario): de manera que la dirección de la proyección es normal al plano de proyección. Esto no ocurre en la proyección paralela oblicua.

Los tipos más comunes de proyecciones ortográficas son la de relación frontal, elevación superior o elevación de plano y la de elevación lateral. En todas ellas, el plano de proyección es perpendicular al eje principal, que por lo tanto es la dirección de la proyección. En la 3.40 se presenta la construcción de estas tres proyecciones, que se usan comúnmente en dibujos de ingeniería para representar piezas de maquinaria, montajes y edificios, ya que las distancias y los ángulos se pueden medir a partir de las representaciones.

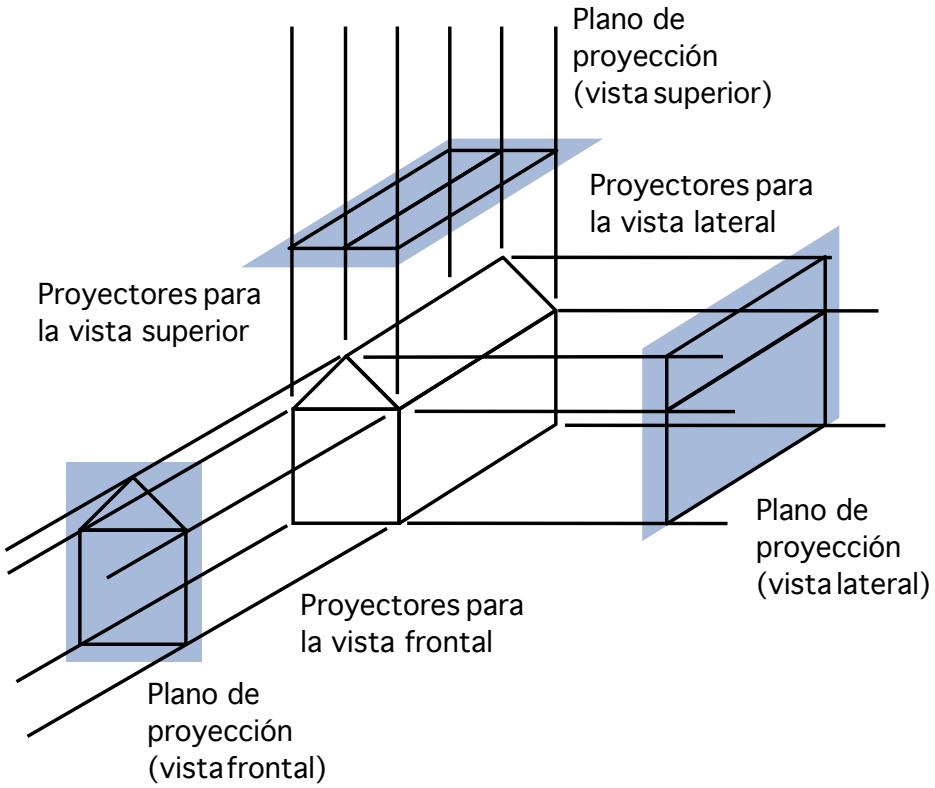


Fig. 3.40.Construcción de tres proyecciones ortográficas.

Sin embargo, cada proyección sólo muestra una. cara del objeto, de manera que puede ser difícil deducir la naturaleza tridimensional del objeto proyectado, incluso si se estudian simultáneamente varias proyecciones del mismo objeto.

Las proyecciones ortográficas axonométricas usan planos de proyección que no son normales a un eje principal y que por ende muestran varias caras de un objeto al mismo tiempo. En este aspecto se parecen a la proyección de perspectiva, pero difieren en lo referente al recorte frontal, que es uniforme y no relacionado con la distancia al centro de proyección.

Se conserva paralelismo de las líneas, pero no los ángulos, y las distancias se pueden medir sobre cualquiera de los ejes principales (por lo general, con factores de escalamiento diferentes).

Las proyecciones oblicuas, la segunda clase de proyecciones paralelas, difieren de las proyecciones ortográficas en que la normal al plano de proyección y la dirección de la proyección son diferentes. Las proyecciones oblicuas combinan las propiedades de las proyecciones ortográficas frontal, superior y lateral con las de una proyección axonométrica: el plano de proyección es normal a un eje principal, de manera que la proyección de la cara del objeto paralela a este plano permite medir ángulos y distancias.

También se proyectan otras caras del objeto, lo que permite medir las distancias sobre los ejes principales, aunque no los ángulos. En la figura 3.41 se muestra la construcción de una proyección oblicua. Observe que la normal al plano de proyección la dirección de la proyección no son iguales.

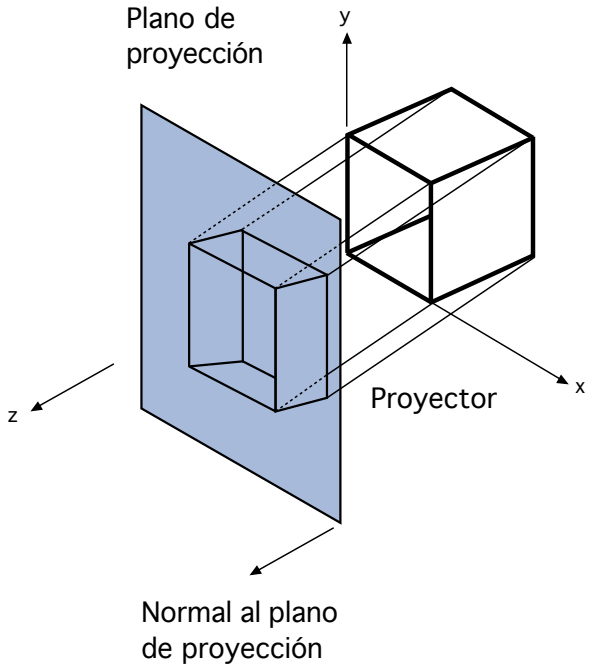


Fig. 3.41.Construcción de una proyección oblicua.

En la figura 3.42 se muestran las relaciones lógicas entre los diversos tipos de proyecciones. El lazo común de todas las proyecciones es que comprenden un plano de proyección y un centro de proyección para las proyecciones de perspectiva o una dirección de proyección para las proyecciones paralelas.

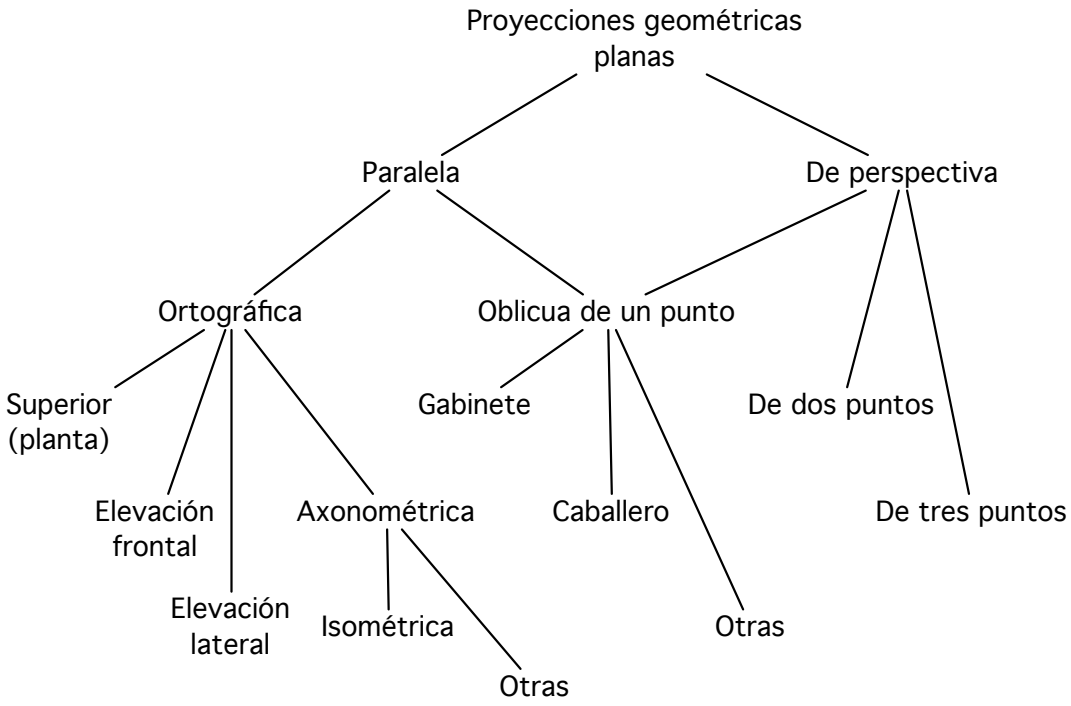


Fig. 3.42.Subclases de las proyecciones geométricas planas. Las vistas de planta es otro término para la vista superior. Las vistas frontal y lateral muchas veces se emplean sin el término elevación.

Podemos unificar los casos paralelos y de perspectiva si consideramos que el centro de proyección está definido por la dirección al centro de proyección desde un punto de referencia y la distancia a este punto. Cuando esta distancia aumenta hasta el infinito, la proyección se convierte en paralela. Por lo tanto, podemos decir que el aspecto común de estas proyecciones es que comprenden un plano de proyección, una dirección al centro de proyección y una distancia al centro de proyección.

Proyección isométrica

La proyección isométrica es una proyección axonométrica de uso común. La normal al plano de proyección (y por consiguiente la dirección de la proyección) forma ángulos iguales con respecto a cada eje principal.

Si la normal al plano de proyecciones es (*dx, dy, dz*), requerimos que *|dx|* = *|dy|* = *|dz|* o *±dx* = *±dy* ± *dz*. Sólo hay ocho direcciones (una en cada octante) que satisfacen esta condición. En la figura 6.8 se muestra la construcción de tina proyección isométrica a lo largo de una de estas direcciones, (1, -1, -1),

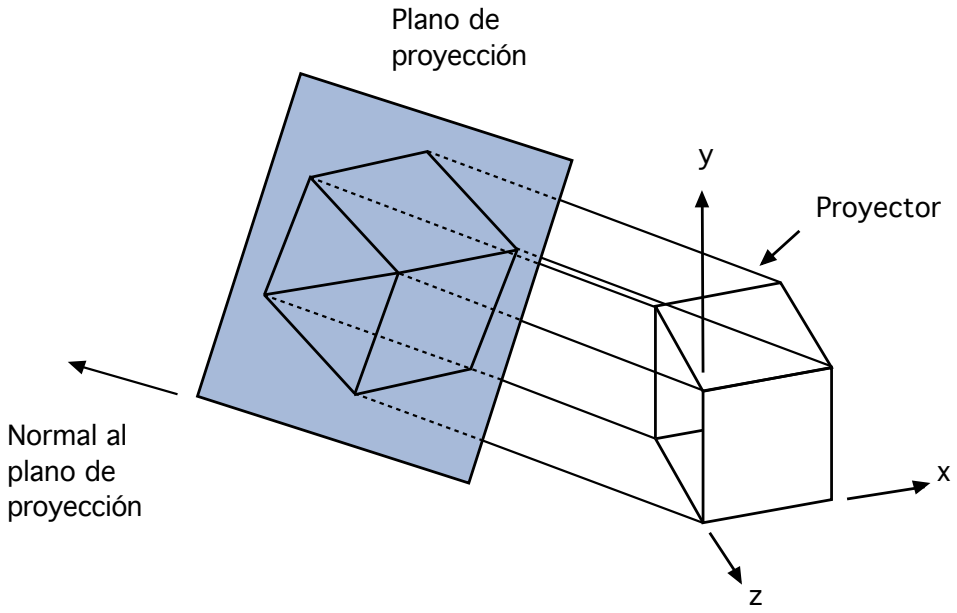


Fig. 3.43.Construcción de una proyección isométrica de un cubo.

La proyección isométrica tiene una útil propiedad: los tres ejes principales tienen la misma reducción frontal, lo que permite que las mediciones sobre los ejes se realicen con la misma escala (de aquí proviene el nombre: iso, que quiere decir “igual” y métrico de medición). Además, las proyecciones de los ejes principales forman ángulos iguales de 120° entre sí.

Identificación de superficies y línea visible

También es posible aclarar las relaciones de profundidad en un despliegue de armazón al identificar de alguna manera las líneas visibles. el método más sencillo consiste en realzar las líneas visibles o desplegarlas en un color diferente. Otra técnica que se utiliza en forma común para los diseños de ingeniería, es el despliegue de áreas no visibles como líneas de rayas.

Otro planteamiento consiste en eliminar no sólo las líneas visibles, como en las figuras 3.44 (b) y 3.44 (c). Pero al eliminar las líneas ocultas también se elimina la información

acerca de la forma de las superficies traseras de un objeto. Estos métodos de línea visible también identifican alas superficies visibles de los objetos.

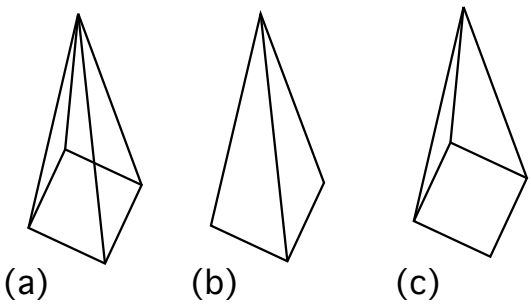


Fig. 3.44.Las representación en armazón de la pirámide en la parte (a) no contiene información de profundidad para indicar si la dirección de la vista es (b) hacia abajo desde una posición sobre el ápice o (c) hacia arriba desde una posición abajo de la base.

Cuando se deben desplegar objetos con color o superficie sombreadas, aplicamos procedimientos de representación de superficie para las superficies visibles, de modo que se oscurezcan las superficies ocultas. Algunos algoritmos de superficie visible establecen la visibilidad pixel por pixel a lo largo del plano de vista; otros algoritmos determinan la visibilidad para las superficies de un objeto como un todo.

3.3.Representación tridimensional de objetos

Superficies de polígonos

La representación de frontera que más se utiliza para un objeto gráfico tridimensional es un conjunto de polígonos de superficie que encierran el interior del objeto. Muchos sistemas gráficos almacenan todas las descripciones de objetos como conjuntos de polígonos de superficie. Esto facilita y acelera la representación de superficie y el despliegue de objetos, ya que todas las superficies se describen con ecuaciones lineales. Por esta razón, con frecuencia nos referimos a las descripciones de polígonos como “objetos gráficos estándar”.

En algunos casos, una representación de polígonos es la única disponible, pero muchos paquetes permiten que los objetos se describan como otros esquemas, como superficies de spline, que se convierten en representaciones de polígonos para el procesamiento.

Una representación de polígono para un poliedro define con precisión las características de superficie del objeto. Pero para otros objetos, las superficies se teselan (o tejan) para producir la aproximación del enlace polígonos. En la figura 3.45, la superficie de un cilindro se representa como un enlace de polígonos. Tales representaciones son comunes en las aplicaciones de diseño y modelado de sólidos, ya que el contorno de armazón se puede desplegar con rapidez para dar una indicación general de la estructura de la superficie.

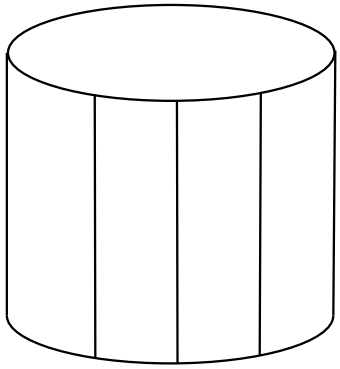


Fig. 3.45.Representación de estructura de alambre de un cilindro
con eliminación de las lineas posteriores (ocultas).

Las representaciones realistas se producen al interpolar los patrones de sombreado a lo ancho de las superficies de los polígonos para eliminar o reducir la presencia de fronteras de aristas de polígonos. Y la aproximación del enlace de polígonos a una superficie curva se puede mejorar al dividir la superficie en facetas de polígonos más pequeños.

Tablas de polígono

Especificamos una superficie de polígono con un conjunto de coordenadas de vértice y parámetros de atributos asociados. Conforme se da entrada a la información para cada polígono, los datos se colocan en tablas que se van a utilizar en el subsecuente procesamiento, despliegue y manipulación de objetos en una escena.

Las tablas de datos de polígonos se pueden organizar en dos grupos: tablas geométricas y tablas de atributos. Las tablas geométricas de datos contienen las coordenadas de vértices y los parámetros para identificar ña orientación espacial de las superficies del polígono.

La información de los atributos para un objeto incluye los parámetros que especifican el grado de transparencia del objeto y las características de reflectividad y textura de su superficie.

Una organización conveniente para almacenar los datos geométricos es crear tres listas: una tabla de vértices, una tabla de aristas y una tabla de polígonos. Los valores de las coordenadas para cada vértice en el objeto se almacenan en la tabla de vértices. La tabla de aristas contiene indicadores en la tabla de vértices para identificar los vértices para cada arista del polígono.

La tabla de polígonos contiene indicadores en la tabla de aristas para identificar las aristas para cada polígono. Este esquema se ilustra en la figura 3.46 para dos polígonos adyacentes sobre la superficie de un objeto. Además, a los objetos individuales y sus caras de polígonos componentes se pueden asignar identificadores de objeto y de faceta para una referencia fácil.

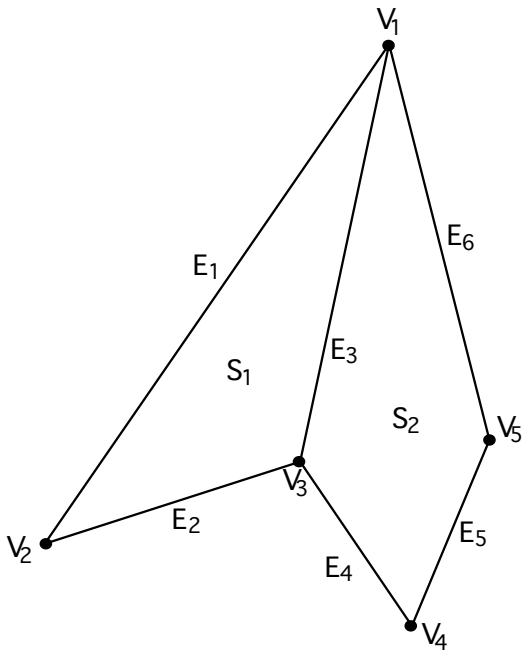


TABLA DE VÉRTICES
V_1: x_1, y_1, z_1
V_2: x_2, y_2, z_2
V_3: x_3, y_3, z_3
V_4: x_4, y_4, z_4
V_5: x_5, y_5, z_5

TABLA DE ARISTAS
E_1: V_1, V_2
E_2: V_2, V_3
E_3: V_3, V_1
E_4: V_3, V_4
E_5: V_4, V_5
E_6: V_5, V_1

TABLA DE SUPERFICIE DE POLÍGONOS
S_1: E_1, E_2, E_3
S_1: E_3, E_4, E_5, E_6

Fig. 3.46.Superficies de polígonos adyacentes, que se forman con seis aristas y cinco vértice y su representación de tabla de datos geométricos.

Al listar los datos geométricos en tres tablas, como en la figura anterior, se proporciona una referencia conveniente a los componentes individuales (vértices, aristas y polígonos) de cada objeto. También, el objeto se puede desplegar de manera eficiente al utilizar los datos de la tabla de aristas para trazar las líneas de los componentes. Un orden posible es emplear solo dos tablas: una de vértices y una de polígonos. Pero este sistema es menos conveniente y algunos aristas se pueden trazar dos veces.

Otra posibilidad es utilizar solo una tabla de polígonos pero esto duplica la información sobre las coordenadas, ya que los valores explícitos de las coordenadas se listan para cada vértice en cada polígono. También la información sobre las aristas se tendrían que reconstruir de las listas de vértices en las tabla de polígonos

Podemos agregar información adicional a las tablas de datos de la figura anterior para una extracción más rápida de la información Por ejemplo, podríamos ampliar la tabla de aristas para incluir más indicadores en la tabla de polígonos de modo que las aristas comunes entre los polígonos se pudieran identificar con mayor rapidez (figura 3.47). Esto es en particular útil para los procedimientos de representación que deben variar con suavidad el sombreado de la superficie a través de las aristas de un polígono al siguiente. En

forma similar, la tabla de vértices se podría- expandir de manera que se haga una referencia cruzada de los vértices con las aristas correspondientes.

E_1: V_1, V_2, S_1
E_2: V_2, V_3, S_1
E_3: V_3, V_1, S_1, S_2
E_4: V_3, V_4, S_2
E_5: V_4, V_5, S_2
E_6: V_5, V_1, S_2

Fig. 3.47.Tabla de aristas para las superficies de la figura anterior expandidas para incluir apuntadores a la tabla de polígonos

La información geométrica adicional que por lo general se almacena en las tablas de datos incluye la inclinación para cada arista y las extensiones de las coordenadas para cada polígonos Al dar entrada a los vértices podemos calcular las inclinaciones de las aristas y rastrear los valores de las coordenadas para identificar los valores mínimo y máximo de x, y y z para los polígonos individuales. Las inclinaciones de las aristas y la información sobre el recuadro de entrelazado para los polígonos se necesitan en el procesamiento subsecuente, por ejemplo, en la representación de superficies. Las extensiones de las coordenadas también se utilizan en algunos algoritmos de determinación de superficie visible.

Ya que las tablas geométricas de datos pueden contener extensas listas de vértices y aristas para los objetos complejos, es importante que se verifique la consistencia y la totalidad de los datos. Cuando se especifican las definiciones de los vértices,arista y polígonos es posible, sobre todo en las aplicaciones interactivas, que se cometan ciertos errores de captura que distorsionarían el despliegue del objeto. Mientras más información se incluya en las tablas de datos, será más fácil verificar los errores. Por tanto, la verificación de los errores es más fácil cuando se utilizan tres tablas de datos (vértices aristas y polígonos) ya que este esquema proporciona la mayor cantidad de información. Algunas de las pruebas que se pueden llevar a cabo mediante un paquete de graficación son (1) que

cada vértice se lista como un extremo para dos aristas por lo menos, (2) que cada arista es parte de un polígono por lo menos, (3) que cada polígono está cerrado, (4) que cada polígono tiene por lo menos una arista compartida y (5) que si la tabla de aristas contiene indicadores para los polígonos cada arista a la que se hace referencia mediante un indicador de polígono tiene un indicador recíproco hacia el polígono.

Ecuaciones de plano

Para producir un despliegue de un objeto tridimensional, debemos procesar la representación de los datos de entrada para el objeto a través de varios procedimientos. Estos pasos del procesamiento incluyen la transformación de las descripciones de modelado y de coordenadas mundiales a coordenadas de vista, después a coordenadas de dispositivo; la identificación de las superficies visibles; y la aplicación de los procedimientos de representación de superficie. Para algunos de estos procesos, necesitamos la información acerca de la orientación espacial de los componentes individuales de superficie del objeto. Esta información se obtiene de los valores de las coordenadas de los vértices y las ecuaciones que describen los planos de los polígonos.

3.27. $$Ax + By + Cz + D = 0$$

La ecuación para una superficie de plano se puede expresar en la forma donde (x, y, z) es cualquier punto en el plano y los coeficientes A, B, C y D son constantes que describen las propiedades espaciales del plano. Podemos obtener los valores de A, B, C y D al resolver un conjunto de tres ecuaciones de plano al utilizar los valores de las coordenadas para tres puntos no colineales en el plano. Para este propósito podemos seleccionar tres vértices sucesivos de polígono (x_1, y_1, z_1), (x_2, y_2, z_2) y (x_3, y_3, z_3) y resolver el siguiente conjunto de ecuaciones lineales del plano simultáneas para las relaciones A/D, B/D y C/D:

3.28. $$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1 \qquad k = 1,2,3$$

La solución para este conjunto de ecuaciones se puede obtener en forma de determinante, utilizando la regla de Cramer, como

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

3.29.

Al ampliar los determinantes, podemos escribirlos cálculos para los coeficientes del plano en la forma

$$\begin{aligned} A &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\ B &= z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \\ C &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\ D &= -x_1(y_2 z_3 - y_3 z_2) - x_2(y_3 z_1 - y_1 z_3) - x_3(y_1 z_2 - y_2 z_1) \end{aligned}$$

3.30.

Conforme se da entrada en la estructura de datos del polígono a los valores de los vértices y otra información los valores para A, B, C y D se calculan para cada polígono y se almacenan con los demás datos del polígono.

La orientación en el espacio de la superficie de un plano se puede describir con el vector normal para el plano,como se muestra en la figura anterior. Este vector normal de superficie tiene los componentes cartesianos (A, B, C) donde los parámetros A, B y C son los coeficientes de plano que se calculan en las ecuaciones 3.30.

Ya que por lo general manejamos superficies de polígonos que encierran el interior de un objeto, necesitamos distinguir entre los dos lados de la superficie. El lado del plano que da al interior del objeto se llama cara "interna" y el lado visible o de afuera es la cara "externa". Si los vértices del polígono se especifican en una

dirección opuesta a la de las manecillas del reloj cuando se ve el lado externo del plano en un sistema de coordenadas del lado derecho, la dirección del vector normal será de adentro hacia afuera. Esto se demuestra para un plano de un cubo de unidad en la figura siguiente.

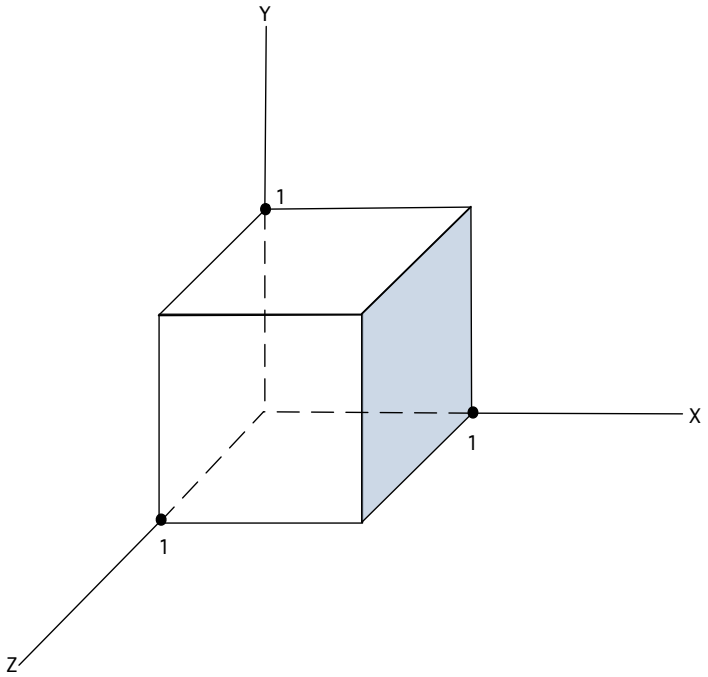


Fig. 3.48.La superficie sombreada de polígono del cubo unitario tiene la ecuación de plano x -1 = 0 y el vector normal N = (1, 0, 0)

Para determinar los componentes del vector normal para la superficie sombreada que se muestra en la figura anterior, seleccionamos tres de los cuatro vértices a lo largo de la frontera del polígono. Estos puntos se seleccionan en dirección opuesta a las manecillas del reloj al ver desde afuera el cubo hacia el origen. Las coordenadas para estos vértices en el orden seleccionado, se pueden utilizar en las ecuaciones 3.30 para obtener los coeficientes del plano: A = 1, B =O, C = O, D = -1. De esta forma, el vector normal para este plano está en la dirección del eje positivo de x.

Los elementos de la normal del plano también se pueden obtener al utilizar un cálculo vectorial de producto cruz. Una vez más seleccionamos tres posiciones de vértice V_l, V_2 y V_3, que se toman en orden opuesto a la dirección de las manecillas del reloj cuando se ve la superficie de afuera hacia adentro en un sistema cartesiano del lado derecho. Al formar dos vectores, uno de V_1 a V_2 y el otro de V_1, a V_3, calculamos N como el producto cruz de los vectores:

3.31. $$N = (v_2 - v_1) \times (v_3 - v_1)$$

Esto genera valores para los parámetros de plano A, B y C. De esta manera podemos obtener el valor para el parámetro D al sustituir estos valores y las coordenadas para uno de los vértices del polígono en la ecuación del plano 3.21 y despejar D. La ecuación del plano se puede expresar en la forma vectorial utilizando la normal N y la posición P de cualquier punto en el plano como

3.32. $$N \cdot P = -D$$

Las ecuaciones del plano se utilizan también para identificar la posición de los puntos espaciales con respecto de las superficies planas de un objeto. Para cualquier punto (*x, y, z*) que no se encuentra en un plano con parámetro A, B, C, D, tenemos

$$Ax + By + Cz + D \neq 0$$

Podemos identificar el punto como adentro o afuera de la superficie del plano de acuerdo con el signo (negativo o positivo) de *Ax + By + Cz + D*:

si *Ax + By + Cz + D < O*, el punto (x, y, z) está adentro de la superficie

si *Ax + By + Cz + D > O*, el punto (x, y, z) está afuera de la superficie

Estas pruebas de desigualdad son válidas en un sistema cartesiano del lado derecho, siempre y cuando los parámetros del plano A, B, C y D se calculen utilizando los vértices seleccionados en un orden contrario al sentido de las manecillas del reloj cuando se ve la superficie en una dirección de afuera hacia adentro. Por ejemplo, en la figura 3.48,cualquier punto afuera del plano sombreado satisface la desigualdad *x – 1 > O*, en tanto que cualquier punto adentro del plano tiene un valor de coordenada de *x* menor que 1,

Enlaces de polígono

Algunos paquetes de gráficos proporcionan varias funciones de polígonos para el modelado de objetos. Una superficie de plano sencilla se puede especificar con una función como fillArea. Pero cuando las superficies de los objetos se deben tejer, es más conveniente especificar las facetas de la superficie con una función de enlace. Un tipo de enlace de

polígonos es la secuencia de triángulos Esta función produce n - 2 triángulos conectados, como se muestra en la figura siguiente, dadas las coordenadas para n vértices.

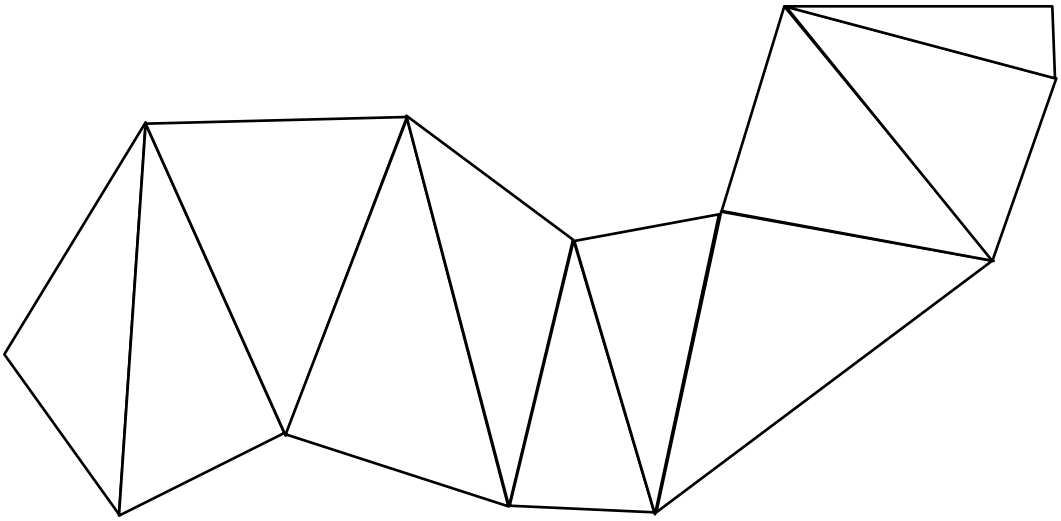


Fig. 3.49.Hilera de triángulos que se forma con 11 triángulos, los cuales conectan 13 vértices.

Otra función similar es el enlace cuadriteral que genera un enlace de (*n – 1*) por (*m – 1*) cuadrilaterales, dadas las coordenadas para una matriz, de vértices de *n* por *m*. La figura 3.50 muestra 20 vértices que forman un enlace de 12 cuadriláteros.

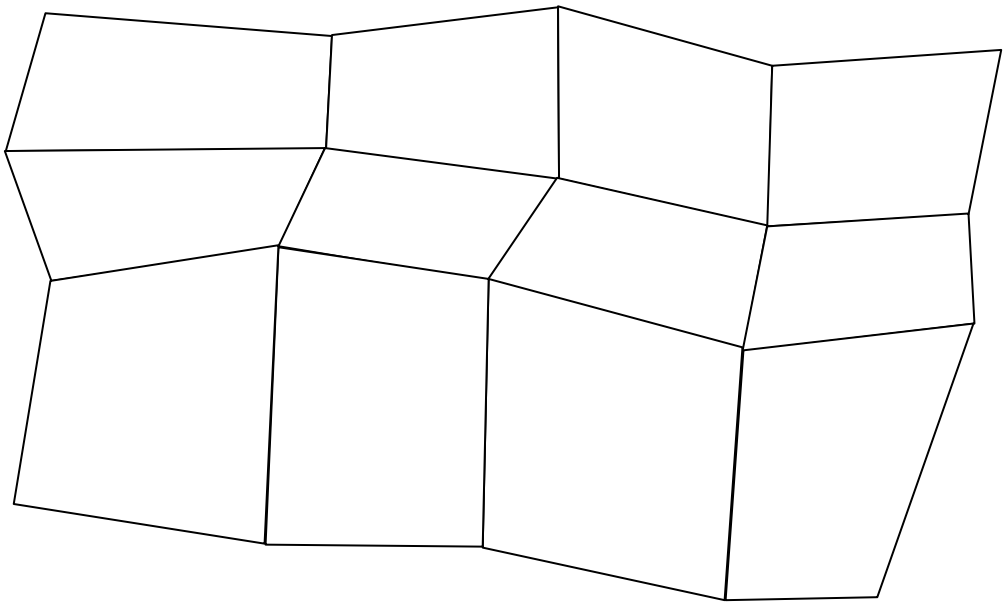


Fig. 3.50.Enlace cuadriteral que contiene 12 cuadriterales que se construyen partir de una matriz de entrada de vértices de 5 por 4.

Cuando los polígono se especifican con más de tres vértices, es posible que no todos los vértices caigan en un plano. Esto se puede deber a errores numéricos o a errores al seleccionar las posiciones de las coordenadas para los vértices. Una manera de controlar esta situación es simplemente dividir los polígono en triángulos. Otra estrategia que en ocasiones se sigue es aproximar los parámetros de los planos A, B y C. Podemos realizar esta operación al promediar los métodos o podemos proyectar el polígono en los planos de las coordenadas. Al utilizar el método de la proyección, tomamos A proporcional al área de la proyección del polígono en el plano de *yz*, B proporcional al Área de proyección en el plano de *xz* y *C* proporcional al área de proyección en el plano *xy*.

Los sistemas de graficación de alta calidad por lo general modelan objetos con enlaces de polígono y establecen una base de datos de información geométrica y de atributos para facilitar el procesamiento de las facetas del polígono. Los sistemas de presentación de polígonos implementados con hardware rápido se incorporan en esos sistemas con la capacidad de desplegar desde cientos de miles hasta un millón o más de polígonos sombreados por segundo(por lo general triángulos), incluyendo la aplicación de la textura de superficies y los efectos especiales de iluminación.

Líneas y superficies curvas

Los despliegues de líneas y superficies curvas tridimensionales se pueden generar a partir de un conjunto de entrada de funciones matemáticas que definen los objetos o de un conjunto de puntos de datos específicos para el usuario. Cuando las funciones se especifican, un paquete puede proyectar las ecuaciones de definición para una curva hacia el plano de despliegue y trazar las posiciones de pixel a lo largo de la trayectoria de la función proyectada Para las superficies, con frecuencia se tesela una descripción funcional para producir una aproximación de enlace de polígonos a la superficie. Por lo general, esto se realiza con parches de polígono triangulares para asegurar que todos los vértices de cualquier polígono estén en un solo plano. Los polígonos específicos con cuatro o más vértices tal vez no tengan todos éstos en el mismo plano. Algunos ejemplos de superficies de despliegue que se generan a partir de descripciones funcionales incluyen los cuádricos y supercuádricos.

Cuando un conjunto de puntos de coordenadas discretos se utiliza para especificar la forma de un objeto, se obtiene una descripción funcional que se adapte mejor a los puntos designados de acuerdo con las restricciones de la aplicación. Las representaciones de spline son ejemplos de esta clase de curvas y superficies. Estos métodos se utilizan por lo general para diseñar formas nuevas de objetos, para digitalizar trazos y describir trayectorias de animación. Los métodos de adaptación de las curvas se utilizan también para desplegar gráficas de valores de datos al adaptar funciones específicas de curva al conjunto discreto de datos, empleando técnicas de regresión como el método de mínimos cuadrados.

Las ecuaciones de curva y superficie se pueden expresar ya sea en forma paramétrica o no paramétrica Para las aplicaciones de las gráficas por computadora, por lo general, las representaciones paramétricas son más convenientes.

Superficies cuadráticas (cuádricas)

Una clase de objetos que se utiliza con frecuencia son las superficies cuádricas, que se describen con ecuaciones de segundo grado (cuadráticas). Incluyen esferas, elipsoides, toros, paraboloides e hiperboloides. Las superficies cuádricas en particular las esferas y elipsoides, son elementos comunes de las escenas gráficas y es frecuente que estén disponibles en los paquetes de gráficos como primitivos de los cuales se pueden elaborar objetos más complejos.

Esfera

En las coordenadas cartesianas, una superficie esférica con radio r que se centra en el origen de las coordenadas se define como el conjunto de puntos (*x, y, z*) que satisfacen la ecuación

$$x^2 + y^2 + z^2 = r^2 \tag{3.33.}$$

También podemos describir la superficie esférica en forma paramétrica al utilizar los ángulos de latitud y longitud:

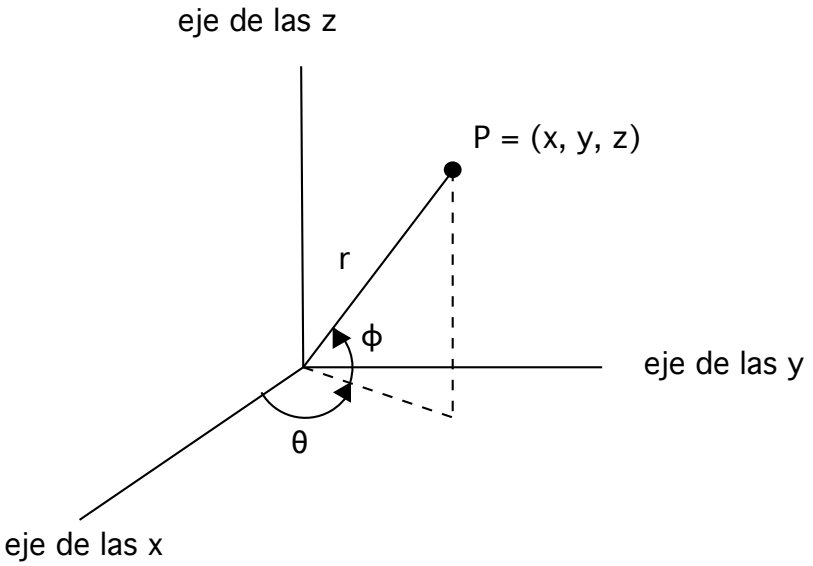


Fig. 3.51.Posición de las coordenadas paramétricas (r, θ, ϕ) sobre la superficie de una esfera con radio r.

$$x = r\cos\phi\cos\theta, -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}$$
$$y = r\cos\phi\sin\theta, -\pi \le \theta \le \pi$$
$$z = r\sin\phi$$

3.34.

La representación paramétrica en las ecuaciones 3.34 proporciona un rango simétrico para los parámetros angulares θ y ϕ. En forma alternativa, podríamos despejar las ecuaciones paramétricas al utilizar coordenadas esféricas estándar, donde el ángulo θ se especifica como la colatitud (figura 3.52). Entonces, θ se define sobre el rango $0 \le \phi \le \pi$, y θ con frecuencia se toma en el rango $0 < \theta < 2\pi$. También podríamos establecer la representación al utilizar los parámetros u y v definidos sobre el rango de 0 a 1 al sustituir $(\phi)=\pi u$ y $\theta = 2\pi u$.

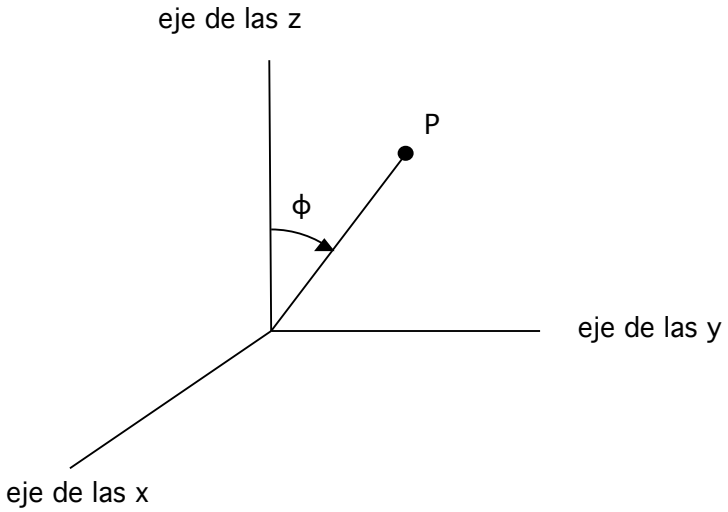


Fig. 3.52.Parámetros de coordenadas esféricas (r, θ, ϕ), al utilizar la colatitud para el ángulo ϕ.

Elipsoide

Una superficie elipsoide se puede describir como una extensión de una superficie esférica donde los radios en tres direcciones mutuamente perpendiculares pueden tener diferentes valores (figura 3.53). La representación cartesiana para los puntos sobre la superficie de un elipsoide que se centra en el origen es

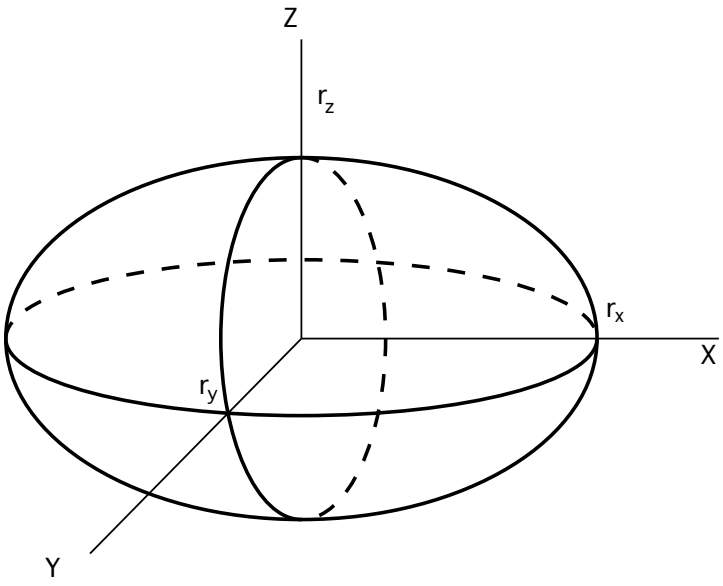


Fig. 3.53.Elipsoide con radios r_x, r_y, r_x que se centra en el origen de las coordenadas

$$\left(\frac{x}{r_x}\right)^2+\left(\frac{y}{r_y}\right)^2+\left(\frac{z}{r_z}\right)^2=1 \tag{3.35.}$$

Y una representación paramétrica ara la elipsoide en términos del ángulo de latitud (ϕ) y el ángulo de longitud θ en la figura 3.53 es

$$\begin{aligned} x &= r_x\cos\phi\cos\theta, -\frac{\pi}{2}\le\phi\le\frac{\pi}{2} \\ y &= r_y\cos\phi\sin\theta, -\pi\le\theta\le\pi \\ z &= r_z\sin\phi \end{aligned} \tag{3.36.}$$

Toro

Un toro es un objeto en forma de dona, como se ilustra en la figura 3.37. Se puede generar al girar una circunferencia u otro cónico alrededor de un eje específico. La representación cartesiana para los puntos sobre la superficie de un toro se pueden expresar en la forma

$$\left[r-\sqrt{\left(\frac{x}{r_x}\right)^2+\left(\frac{y}{r_y}\right)^2}\right]^2+\left(\frac{z}{r_z}\right)^2=1 \tag{3.37.}$$

donde r es cualquier valor de compensación o descentrado específico. Las representaciones paramétricas para un toro son similares a las de una elipse, excepto que el ángulo se extiende a lo largo de 360°. Al utilizar ángulos de latitud y longitud ϕ y θ,podemos describir la superficie del toro como el conjunto de puntos que satisfacen

$$\begin{aligned} x &= r_x(r+\cos\phi)\cos\theta, -\frac{\pi}{2}\le\phi\le\frac{\pi}{2} \\ y &= r_y(r+\cos\phi)\sin\theta, -\pi\le\theta\le\pi \\ z &= r_z\sin\phi \end{aligned} \tag{3.38.}$$

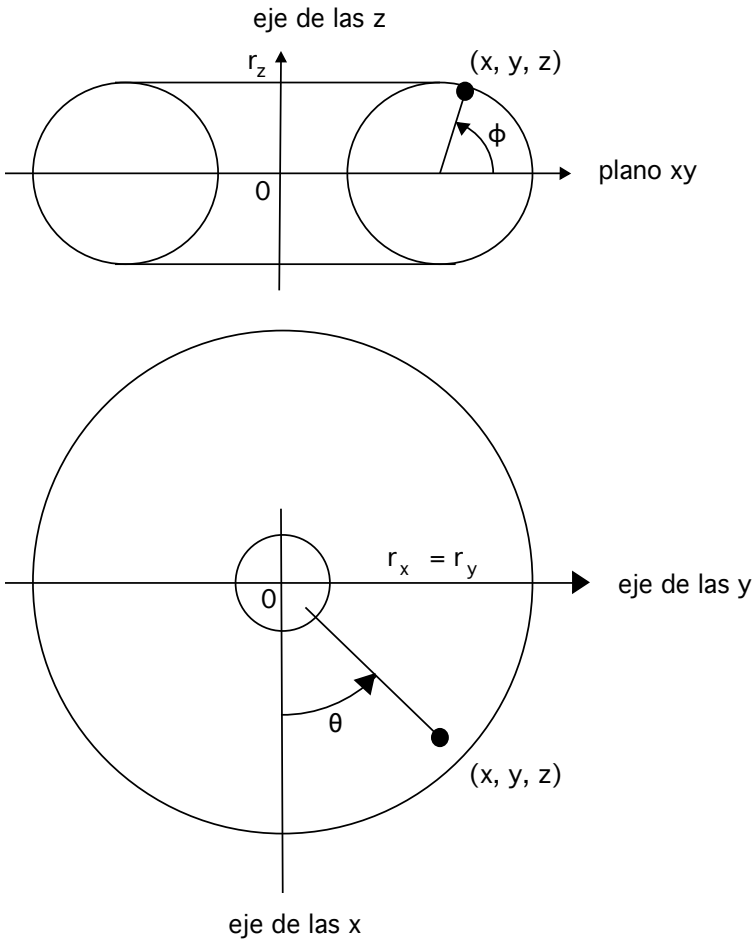


Fig. 3.54.Toro con corte transversal circular que se entra en el origen de las coordenadas

Representaciones de Spline

En la terminología del dibujo mecánico una spline es una banda flexible que se utiliza para producir una curva suave a través de un conjunto de puntos designados. Varios pesos pequeños se distribuyen a lo largo de la banda para mantenerla en posición sobre la mesa de dibujo mientras se traza la curva. El término curva de spline al principio se refería a una curva que se traza de esta manera. Podemos describir esa curva en forma matemática con una función cúbica polinómica cuyas primera y segunda derivadas son continuas a través de las distintas secciones de la curva.

En las gráficas por computadora, el término curva de spline ahora se refiere a cualquier curva compuesta que se forma con secciones polinómicas que satisfacen condiciones específicas de continuidad en la frontera de las piezas. Una superficie de spline se puede

describir con dos conjuntos de curvas ortogonales de spline. Existen varias clases de especificaciones de spline que se utilizan en las aplicaciones gráficas. Cada especificación individual se refiere sólo a un tipo particular de polinomio con ciertas condiciones específicas de frontera.

Las splines se utilizan en las aplicaciones gráficas para diseño formas curvas y de superficie, para digitalizar trazos para el almacenamiento en la computadora y especificar trayectorias de animación para los objetos o la cámara en una escena. Las aplicaciones de CAD típicas para las splines incluyen el diseño de carrocería de automóviles, superficies de aeronaves y naves espaciales, así como cascos de embarcaciones.

Interpolación y aproximación de splines

Especificamos una curva de spline al proporcionar un conjunto de posiciones de coordenadas, que se conocen como puntos de control, que indican la forma general de la curva. Estos puntos de control se ajustan después con funciones polinómicas paramétricas continuas en el sentido de la pieza en una de dos maneras.

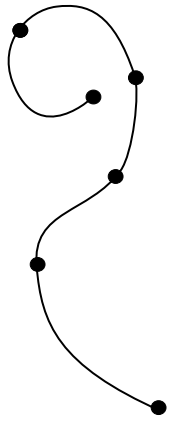


Fig. 3.55.Conjunto de sis puntos de control que se interpolan con secciones polinómicas continuas en el sentido de la pieza.

Cuando las secciones polinómicas se ajustan de modo que la curva pasa a través de cada punto de control, como en la figura 3.55, se dice que la curva que resulta realiza la interpolación del conjunto de puntos de control. Por otra parte, cuando los polinomios se ajustan a la trayectoria general del punto de control sin pasar necesariamente a través de ningún punto de control,se dice que la curva que resulta se aproxima al conjunto de puntos de control (figura 3.56).

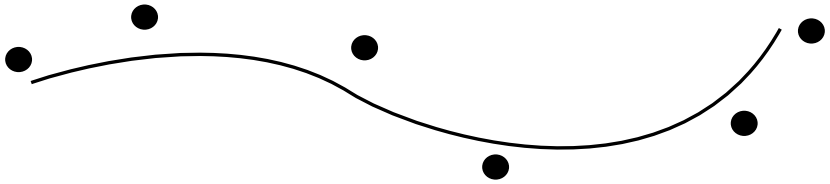


Fig. 3.56.Conjunto de seis puntos de control que se aproximan con secciones polinómicas continuas en el sentido de la pieza.

Las curvas de interpolación se utilizan por lo general para digitalizar trazos o especificar trayectorias de animación. Las curvas de aproximación se utilizan sobre todo como herramientas de diseño para estructurar las superficies de los objetos.

Una curva de spline se define, modifica y manipula con operaciones en los puntos de control. Al seleccionar en forma interactiva las posiciones espaciales para los puntos de control, un diseñador puede establecer una curva inicial. Después de que se despliega el ajuste polinomial para un conjunto determinado de puntos de control, el diseñador puede cambiar de posición todos o algunos de los puntos de control para reestructurar la forma de la curva. Ademas, la curva se puede trasladar, girar o escalar con las transformaciones que se aplican a los puntos de control. Los paquetes de CAD también pueden insertar los puntos de control adicionales para ayudar a un diseñador a ajustar las formas de la curva.

La frontera de polígono convexo que encierra un conjunto de puntos de control se conoce como casco convexo. Una forma de imaginar la forma de un casco convexo considera una banda de hule que se estira alrededor de las posiciones de los puntos de control de modo que cada uno de éstos se encuentra ya sea en el perímetro del casco o adentro de éste (figura 3.57).

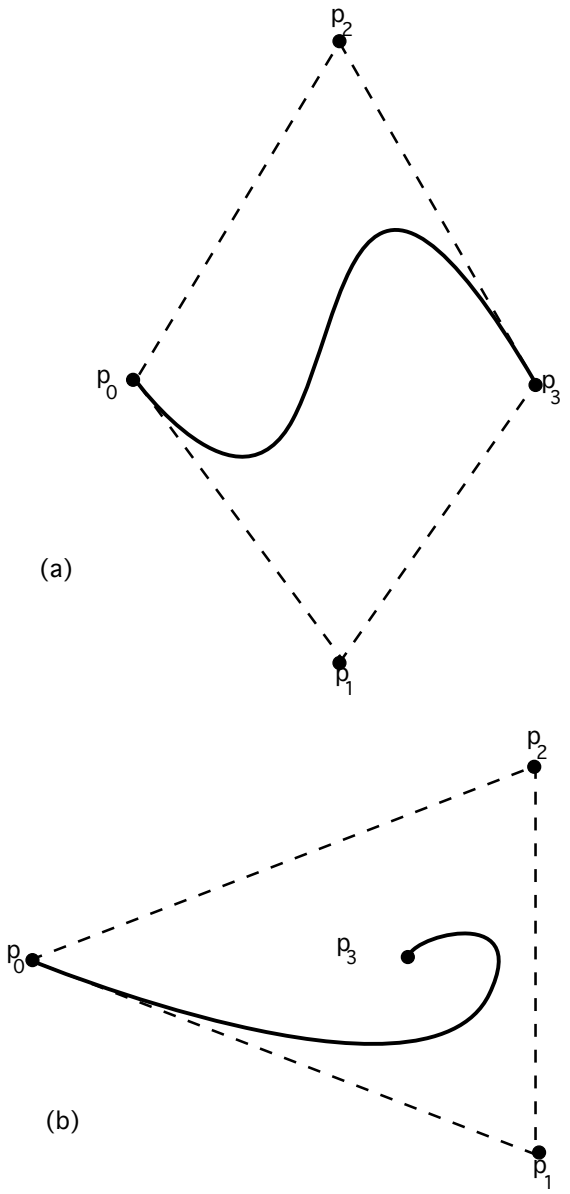


Fig. 3.57.Formas de casco convexo (líneas de rayas) para dos conjuntos de puntos de control.

Los cascos convexos proporcionan una medida para la desviación de una curva o superficie desde la región que limita los puntos de control. Algunas splines están limitadas por el casco convexo, que asegura que los polinomios sigan con suavidad los puntos de control sin oscilaciones erráticas. También la región de polígono adentro del casco convexo es útil en algunos algoritmos como una región de recorte.

Por lo general, una polilínea que conecta la secuencia de puntos de control para una spline de aproximación se despliega para recordar a un diseñador el orden de los puntos de

control. Con frecuencia,este conjunto de segmentos de la línea conectados se conoce como la gráfica de control de la curva.

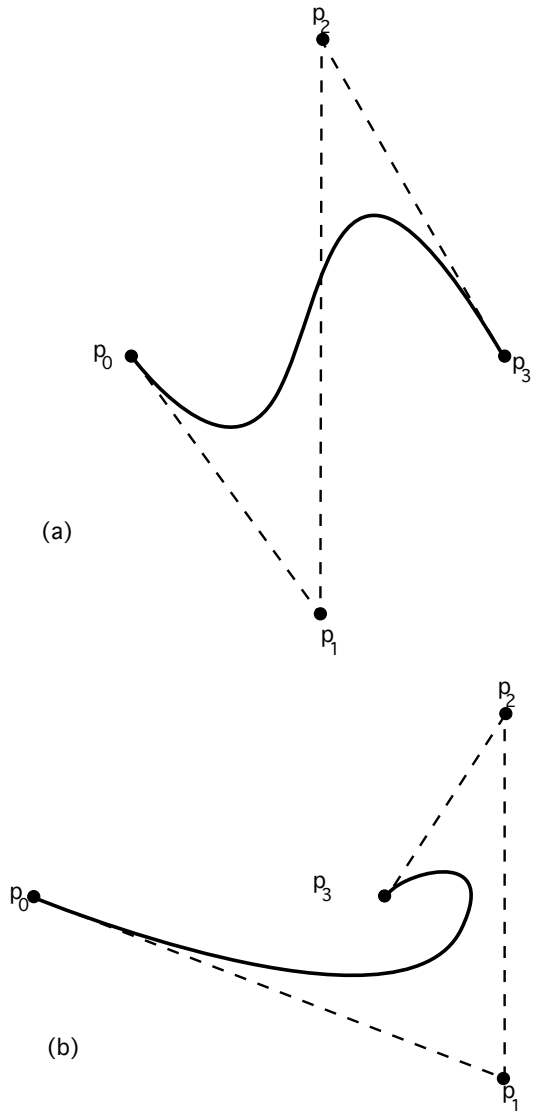


Fig. 3.58.Formas de gráfica de control (líneas de rayas) para dos conjuntos diferentes de puntos de control

Otros nombres para la serie de secciones de la línea recta que conectan los puntos de control en el orden específico son polígono de control y polígono característico. La figura 3.58 ilustra la forma de la gráfica de control para las secuencias del punto de control en la figura 3.57.

Condiciones de continuidad paramétrica

Para asegurar una transición suave de una sección de una curva paramétrica alrededor de una pieza a la siguiente, podemos imponer varias condiciones de continuidad en los puntos de conexión. Si se describe cada sección de una spline como un conjunto de funciones de coordenadas paramétricas de la forma

$$\begin{aligned} x &= x(u), \\ y &= y(u), \\ z &= z(u) \\ u_1 &\le u \le u_2 \end{aligned} \qquad 3.39.$$

establecemos la continuidad paramétrica al comparar las derivadas de secciones curvas adyacentes en su frontera común.

La continuidad paramétrica de orden cero, que se describe como continuidad C^0, sólo implica que las curvas se unen. Es decir, los valores de *x, y* y *z* que se evalúan en u para la primera sección de la curva son iguales, de modo respectivo, a los valores *x, y* y *z* que se evalúan en u_1 para la siguiente sección de la curva.

La continuidad paramétrica de primer orden, que se denomina continuidad C^1, significa que las primeras derivadas paramétricas (línea de tangente) de las funciones de coordenadas en la ecuación 3.39 para dos secciones curvas sucesivas son iguales en su punto de unión. La continuidad paramétrica de segundo orden, o continuidad C^2, implica que tanto la primera como la segunda derivada paramétrica de las dos secciones curvas son las mismas en la intersección.

Las condiciones de continuidad parametrica de orden superior se definen de manera similar. En la figura 3.59 se presentan ejemplos de la continuidad C^0, C^1 y C^2.

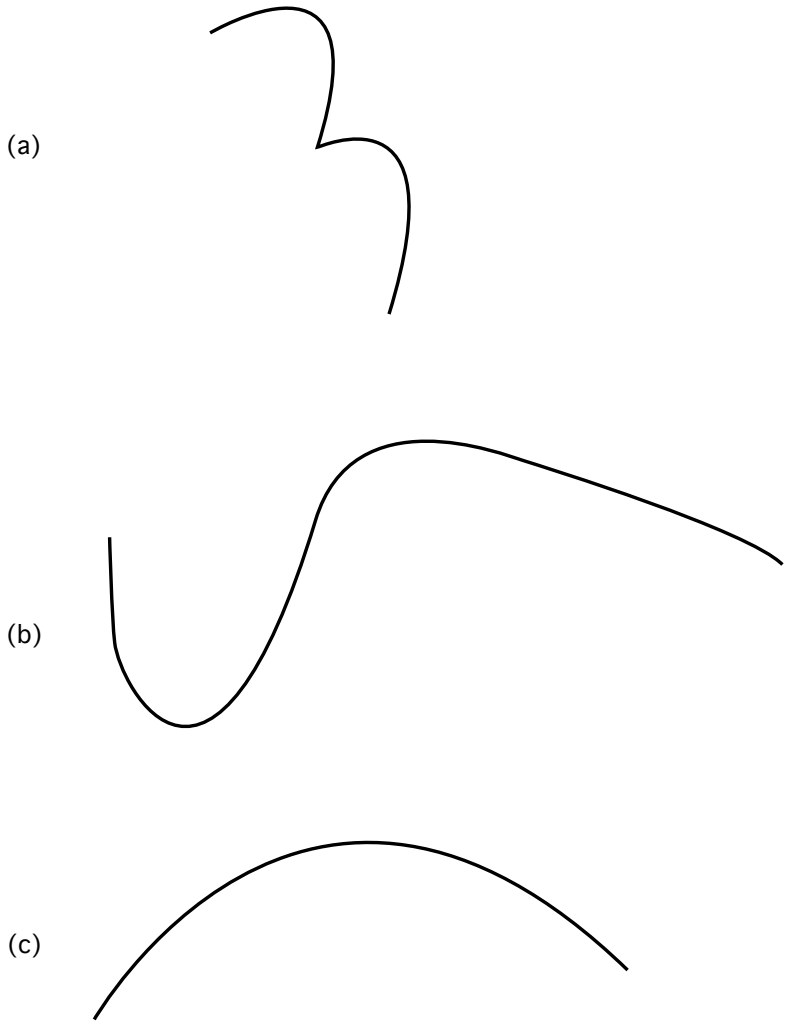


Fig. 3.59.Construcción por piezas de una curva al unir dos segmentos de curva utilizando distintos órdenes de continuidad: (a) sólo continuidad de orden cero; (b) continuidad de primer orden y (c) continuidad de segundo orden.

Con la continuidad de segundo orden, los índices de cambio de los vectores de tangente para las secciones que se conectan son equivalentes en su intersección. Así la línea de tangente realiza una transición suave de una sección de la curva a la siguiente (figura 3.59(c)).

Pero con la continuidad de primer orden, los índices de cambio de los vectores de tangente para las dos secciones son muy diferentes (figura 3.59(b)), de modo que es posible que las formas generales de las dos secciones adyacentes presenten un cambio abrupto.

Con frecuencia, la continuidad de primer orden es suficiente para digitalizar trazos y algunas aplicaciones de diseño,en tanto que la continuidad de segundo orden es útil para establecer trayectorias de animación par a el movimiento de la cámara y para muchos requerimientos de precisión del CAD.

Una cámara que se desplaza a lo largo de la trayectoria de la curva que aparece en la figura 3.59 con pasos iguales en el parámetro u experimentará un cambio brusco en la aceleración en la frontera de las dos secciones y provocará una discontinuidad en la secuencia del movimiento. No obstante, si la cámara se desplazara a lo largo de la trayectoria de la figura 3.59(c), la secuencia de marco para el movimiento efectuará una transición suave a través de la frontera.

Condiciones de continuidad geométrica

Un método alternativo para unir dos secciones curvas sucesivas consiste en especificar condiciones para la continuidad geométrica. En este caso, sólo necesitamos que las derivadas paramétricas de las dos secciones sean proporcionales entre sí en su frontera común en vez de ser equivalentes.

La continuidad geométrica de orden cero, que se describe como continuidad G^0, es la misma que la continuidad paramétrica de orden cero. Es decir, las dos secciones curvas deben tener la misma posición de coordenadas en el punto de la frontera. La continuidad geométrica de primer orden, o continuidad G^1, implica que las primeras derivadas paramétricas son proporcionales en la intersección de dos secciones sucesivas.

Si expresamos la posición paramétrica en la curva como P(u), la dirección del vector de tangente P'(u), pero no necesariamente su magnitud, será la misma para dos secciones curvas sucesivas en su punto de unión de acuerdo con la continuidad G^1. La continuidad geométrica de segundo orden,o continuidad G^2, implica que tanto la primera como la segunda derivada paramétrica de las dos secciones curvas son proporcionales en su frontera. De acuerdo con la continuidad G^2, las curvaturas de la segunda curva se ajustarán en la posición de unión.

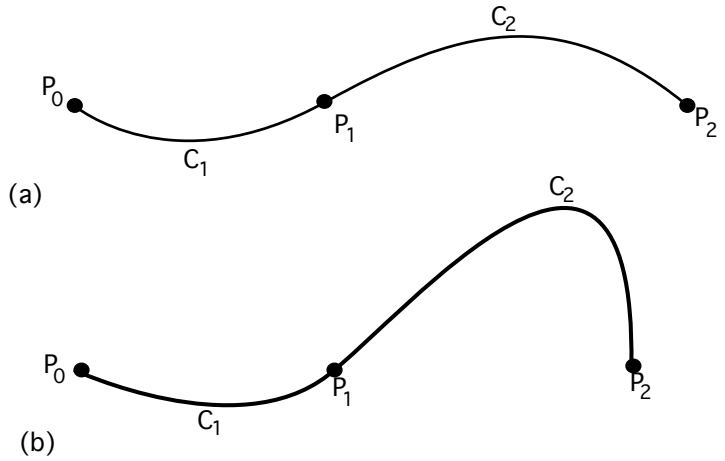


Fig. 3.60.Tres puntos de control que se ajustan con dos secciones curvas unidas con (a) continuidad paramétrica y (b) continuidad geométrica donde el vector tangente de la curva C_3 en el punto P_1 tiene una magnitud mayor que el vector tangente de la curva C_1 en P_1.

Una curva que se genera con condiciones de continuidad geométrica es similar a una que se genera con continuidad paramétrica pero con leves diferencias en la forma de la curva. La figura 3.60 ilustra una comparación de continuidad geométrica y paramétrica. Con la continuidad geométrica la curva se jala hacia la sección que tiene el vector de tangente más alto.

Especificaciones de spline

Existen tres métodos equivalentes para especificar una representación de spline particular: (1) podemos establecer el conjunto de condiciones de frontera que se imponen en la spline; (2) podemos establecer la matriz que caracteriza la spline; o (3) podemos establecer el conjunto de funciones de combinación (o funciones base) que determinan la manera en que se combinan las restricciones geométricas en la curva para calcular posiciones a lo largo de la trayectoria de la curva.

A fin de ilustrar estas tres especificaciones equivalentes, suponga que tenemos la siguiente representación polinómica cúbica paramétrica para la coordenada de x a lo largo de la trayectoria de una sección de la spline:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x, \quad 0 \le u \le 1 \tag{3.40}$$

Por ejemplo, se podrá establecer las condiciones de frontera para esta curva, en las coordenadas de extremo *x(0)* y *x(1)* y en las primeras derivadas paramétricas en los extremos *x'(0)* y *x'(1)*. Estas cuatro condiciones de frontera son suficientes para determinar los valores para los cuatro coeficientes a_x, b_x. c_x y d_x.

A partir de las condiciones de frontera, obtenemos la matriz que caracteriza esta curva de spline al volver a expresar de nuevo la ecuación 3.40 como el producto matricial

$$x(u) = \begin{bmatrix} u^3 u^2 u \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = U \cdot C \tag{3.41}$$

donde U es la matriz renglón de potencia del parámetro u y C es la matriz columna de coeficientes. Al utilizar la ecuación 3.41, es posible expresar las condiciones de frontera en forma de matriz y despejar la matriz de coeficientes C como

$$C = M_{spline} \bullet M_{geom} \tag{3.42}$$

donde M_{geom} es una matriz de columnas de cuatro elementos que contiene los valores de restricción geométrica (condiciones de frontera) en la spline y M_{spline} es la matriz de 4 por 4 que transforma los valores de restricción geométrica a los coeficientes polinómicos y ofrece una caracterización para la curva de spline.

La matriz M_{geom} contiene los valores de coordenadas del punto de control y otras restricciones geométricas que se han especificado. Por tanto, podemos sustituir la representación de matriz para C en la ecuación 3.42 para obtener

$$x(u) = U \bullet M_{spline} \bullet M_{geom} \tag{3.43}$$

La matriz M_{spline}, que caracteriza la representación de una spline, en ocasiones llamada matriz base,es de particular utilidad para transformar de una representación de spline a otra.

Por último podemos ampliar la ecuación 3.43 con el propósito de obtener una representación polinómica para la coordenada de x en término de parámetro de restricción geométrica

$$x(u) = \sum_{k=0}^{3} g_k \bullet BF_k(u) \qquad 3.44.$$

donde g_k son los parámetros de restricción como las coordenadas del punto de control y la pendiente de la curva en los puntos de control, y $BF_k(u)$ son las funciones de combinación polinómica. En las secciones siguientes, analizamos algunas splines que se utilizan en forma común así como su matriz y especificaciones de funciones de combinación.

Curvas y Superficies de Bezier

Pierre Bézier, ingeniero francés desarrolló este método de aproximación de splines para utilizarlo en el diseño de las carrocerías de los automóviles Renault. Las splines de Bézier tienen varias propiedades que hacen que sean muy útiles y convenientes para el diseño de curvas y superficies. Asimismo, es fácil implementarlas. Por estos motivos, las splines de Bézier están disponibles en forma común en varios sistemas de CAD, en paquetes generales de gráficas y en paquetes seleccionados de dibujo y pintura.

Curvas de Bézier

En general, es posible ajustar una curva de Bézier para cualquier número de puntos de control. El número de puntos de control que se debe aproximar y su posición relativa determinan el grado del polinomio de Bézier. Del mismo modo que con las splines de interpolación se puede especificar una curva de Bézier con condiciones de frontera con una matriz característica o con funciones de combinación.

Para curvas generales de Bézier la especificación más conveniente es la función de combinación.

Suponga que tenemos n + 1 posiciones de puntos de control: $p_k = (x_k, y_k, z_k)$ con k en el rango de O a n. Es posible combinar estos puntos de coordenadas para producir el siguiente vector de posición P(u), que describe la trayectoria de una función polinómica de Bézier aproximada entre p_0 y p_n.

$$P(u) = \sum_{k=0}^{n} p_k BEZ_{k,n}(u) \qquad (0 \le u \le 1)$$

3.45.

Las funciones de combinación de Bézier $BEZ_{k,n}$ (u) son los polinomios de Bernstein:

$$BEZ_{k,n}(u) = C(n,k)u^k(1-u)^{n-k}$$

3.46.

donde C(n, k) son los coeficientes del binomio:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

3.47.

De la misma manera, podemos definir las funciones de combinación de Bézier con

el cálculo recursivo

$$BEZ_{k,n}(u) = (1-u)BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u), \qquad n \ge k \ge 1$$

3.48.

con $BEZ_{k,k} = u^k$ y $BEZ_{0,k} = (1 - u)^k$. La ecuación vectorial 10-40 representa un conjunto de tres ecuaciones paramétrica para las coordenadas individuales de la curva:

$$x(u) = \sum_{k=0}^{n} x_k BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k BEZ_{k,n}(u)$$

3.49.

Como una regla, una curva de Bézier es un polinomio de grado uno menos el número de puntos de control que se utilizan: tres puntos generan una parábola, cuatro puntos una curva cúbica y así en forma sucesiva. La figura 3.61 demuestra la apariencia de algunas curvas de Bézier para varias selecciones de puntos de control en el plano *xy(z = 0)*. Sin embargo, con ciertas posiciones de los puntos de control obtenemos polinomios de Bézier degenerados. Por ejemplo, una curva de Bezier que se genera con tres puntos de control colineales es un segmento de línea recta. Y un conjunto de puntos de control que se encuentran en su totalidad en la misma posición de coordenadas produce una “curva” de Bézier que es un solo punto.

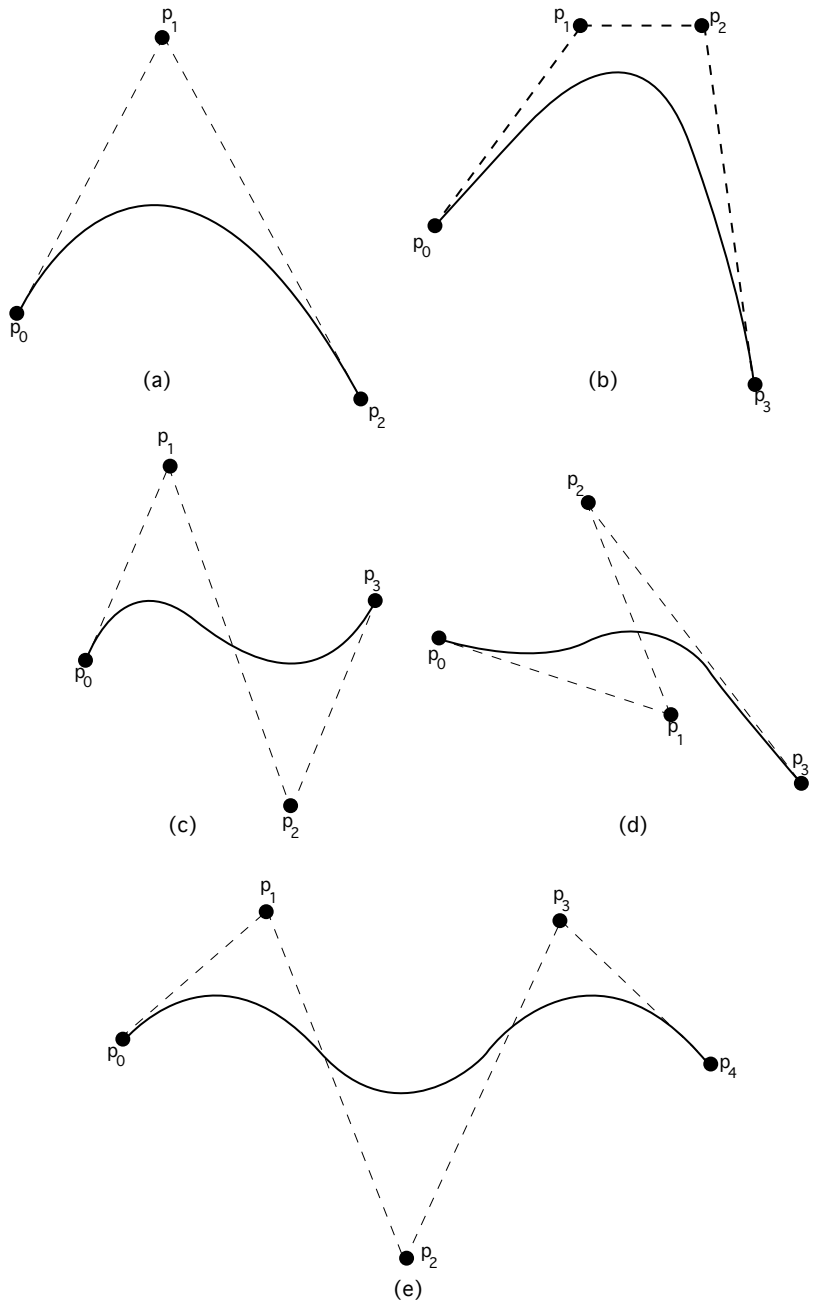


Fig. 3.61.Ejemplos de dos curvas de Bezier bidimensionales que se generaron a partir de tres, cuatro y cinco posiciones de punto de control.

Las curvas de Bézier se encuentran de manera regular en paquetes de pintura y dibujo, así como en sistemas de CAD, puesto que su implementación es fácil y su capacidad es considerable en el diseño de curvas. Se pueden establecer método eficientes para determinar las posiciones de coordenadas a lo largo de una curva de Bézier al utilizar

cálculos recursivos. Por ejemplo, los coeficientes del binomio sucesivos se pueden calcular como

$$C(n,k) = \frac{n-k+1}{k} C(n,k-1) \tag{3.50.}$$

paran n≥k.

Propiedades de las curvas de Bézier

Una propiedad muy útil de una curva de Bézier es que siempre pasa a través del primero y el Último puntos de control. Es decir, las condiciones de frontera en los dos extremos de la curva son

$$\begin{aligned} P(0) &= p_0 \\ P(1) &= p_n \end{aligned} \tag{3.51.}$$

Los valores de las primeras derivadas paramétricas de una curva de Bézier los extremos se pueden calcular a partir de las coordenadas del punto de control como

$$\begin{aligned} P'(0) &= -np_0 + np_1 \\ P'(1) &= -np_{n-1} + np_n \end{aligned} \tag{3.52.}$$

Por tanto, la pendiente en el principio de la curva es a lo largo de la línea que une los dos últimos extremos. De modo similar, las segundas derivadas paramétrica de una curva de Bézier en los extremos se calculan como

$$\begin{aligned} P''(0) &= n(n-1)\left[(p_2 - p_1) - (p_1 + p_0)\right] \\ P''(1) &= n(n-1)\left[(p_{n-2} - p_{n-1}) - (p_{n-1} + p_n)\right] \end{aligned} \tag{3.53.}$$

Otra propiedad importante de cualquier curva de Bézier es que cae adentro del casco convexo (frontera de polígono convexo) de los puntos de control. Esto se desprende de las propiedades de las funciones de combinación de Bézier: todas son positivas y su suma siempre es 1,

$$P(0) = p_0$$
$$P(1) = p_n \qquad 3.54.$$

de manera que cualquier posición de la curva sólo es la suma ponderada de las posiciones de punto de control. La propiedad de casco convexo para una curva de Bézier garantiza que el polinomio siga con suavidad los puntos de control sin oscilaciones erráticas.

Técnicas de diseño utilizando curvas de Bézier

Las curvas cerradas de Bézier se generan al especificar el primero y el último punto de control en la misma posición, como en el ejemplo que aparece en la figura 3.62.

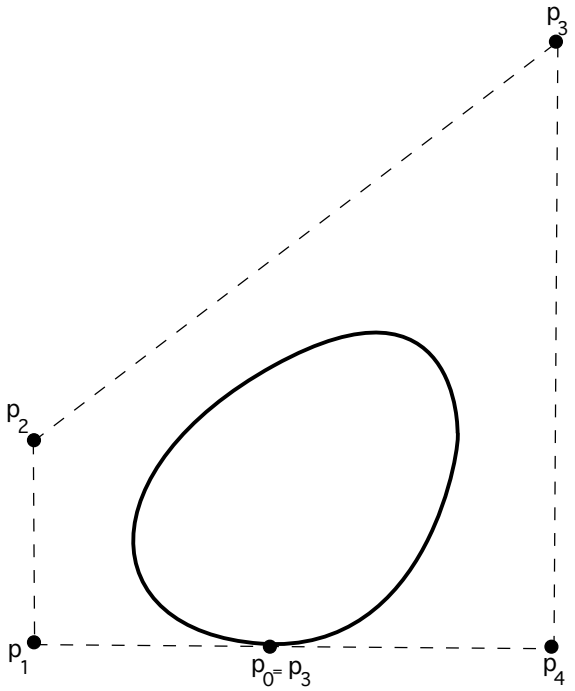


Fig. 3.62.Curva de Bézier cerrada que se genera al especificar el primero y el último punto de control en la misma posición.

Asimismo, al especificar múltiples puntos de control en una sola posición de coordenadas se obtiene más peso para esa posición.

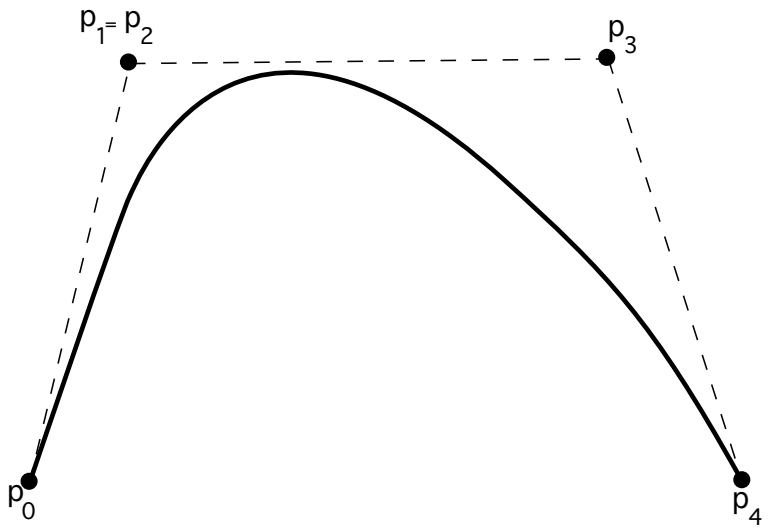


Fig. 3.63.Se puede lograr que una curva de Bézier pase más cerca de una posición de coordenadas determinada al asignar múltiples puntos de control a esa posición.

En la figura 3.63 se da entrada a una sola posición de coordenadas como dos puntos de control y se jala la curva resultante más cerca de esta posición.

Podemos ajustar una curva de Bézier para cualquier número de puntos de control, pero esta operación requiere el cálculo de funciones polinómicas de grado superior. Cuando se deben generar curvas complicadas, se pueden formar al unir varias secciones de Bézier de grado inferior. Unir secciones más pequeñas también nos ofrece un mejor control sobre la forma de la curva en regiones pequeñas.

Dado que las curvas de Bézier pasan a través de los extremos, es fácil comparar secciones curvas (continuidad de orden cero). Asimismo, las curvas de Bézier presentan la propiedad importante de que la tangente para la curva en el extremo está a lo largo de la línea que une ese extremo al punto de control adyacente. Por tanto, a fin de obtener una continuidad de primer orden entre secciones curvas, podemos elegir que los puntos de control p'_0 y p'_1 de una sección nueva está a lo largo de la misma línea recta que los puntos de control p_{n-1} y p_1 de la sección anterior (figura 3.64).

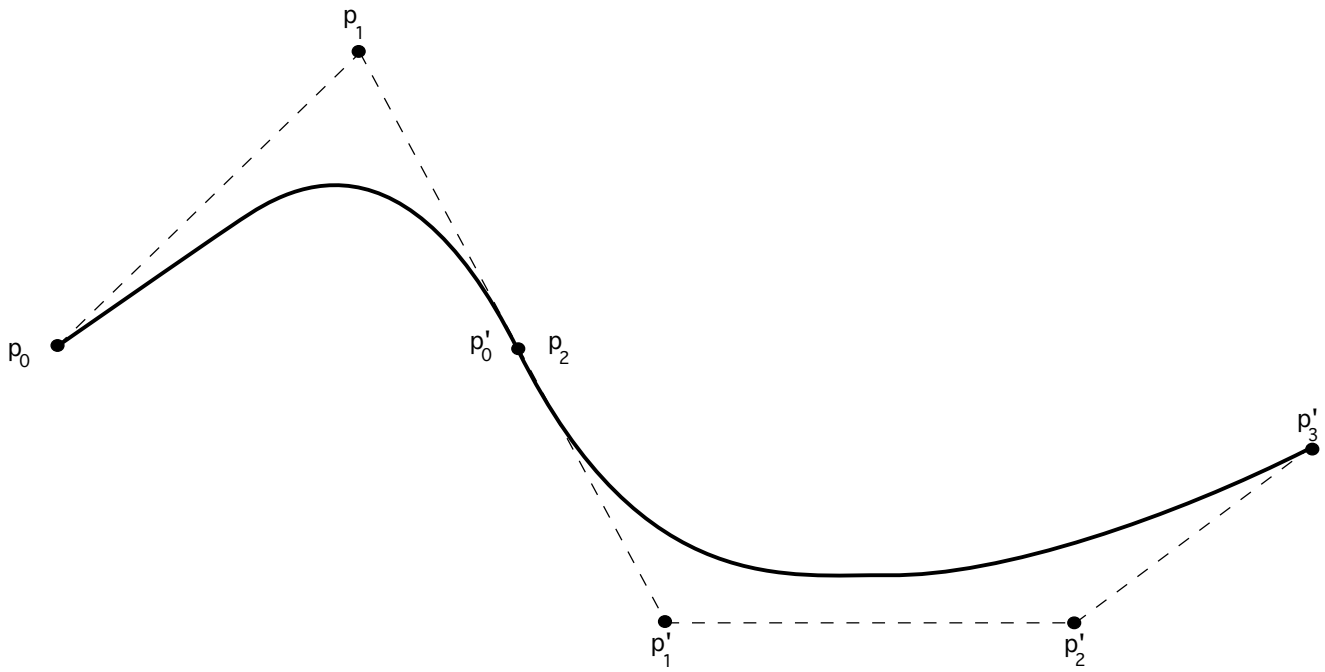


Fig. 3.64.Curva con continuidad de primer orden

Cuando las dos secciones curvas tienen el mismo número de puntos de control, obtenemos una continuidad C^1 al seleccionar el primer punto de control de la nueva sección como el último punto de control de la sección previa al colocar el segundo punto de la sección nueva en la posición

$$p_n + (p_n - p_{n-1})$$

Así los tres puntos de control son colineales y tienen el mismo espaciado.

Obtenemos una continuidad C^2 entre dos secciones de Bézier al calcular la posición del tercer punto de control en una sección nueva en términos de las posiciones de los tres últimos puntos de control de la sección anterior como

$$p_{n-2} + 4(p_n - p_{n-1})$$

Requerir una continuidad de segundo orden entre dos secciones curvas de Bézier puede ser una restricción innecesaria. Esto es cierto, en especial, con curvas cúbicas que sólo tienen cuatro puntos de control por sección. En este caso, la continuidad de segundo orden fija la posición de los tres primeros puntos de control y nos deja sólo un punto que podemos utilizar para ajustar la forma del segmento de curva.

Curvas cúbicas de Bézier

Muchos paquetes de gráficas sólo ofrecen funciones de spline cúbica Esto da una flexibilidad de diseño considerable en tanto que se evitan los demás cálculos necesarios con polinomios de orden superior. Las curvas cúbicas de Bézier se generan con cuatro puntos de control. Las cuatro funciones de combinación para curvas cúbicas de Bezier que se obtienen al sustituir n = 3 en la ecuación de curvas Bezier con polinomios Bernstein, son

$$BEZ_{0,3}(u) = (1-u)^3$$
$$BEZ_{1,3}(u) = 3u(1-u)^2$$
$$BEZ_{2,3}(u) = 3u^2(1-u)$$
$$BEZ_{3,3}(u) = u^3$$

3.55.

En la figura 3.65 se ilustran los trazos de las cuatro funciones cúbicas de combinación de Bézier. La forma de las funciones de combinación determina la manera en que los puntos de control influyen sobre la forma de la curva para los valores del parámetro u en el rango de 0 a 1.En u =O, la única función de combinación no cero es $BEZ_{0,3}$ que tiene el valor 1. En u = 1, la única función no cero es $BEZ_{3,3}$, con un valor de 1 en ese punto. De esta manera la curva cúbica de Bézier siempre pasará a través de los puntos de control p_1, y p_2.

Las otras funciones, $BEZ_{1,3}$ y $BEZ_{2,3}$, influyen en la forma de la curva en valores intermedios del parámetro u, de modo que la curva resultante se inclina hacia los puntos p_1, y p_2. La función de combinación $BEZ_{1,3}$ es el máximo en u = 1/3 y $BEZ_{2,3}$ es el máximo en u = 2/3.

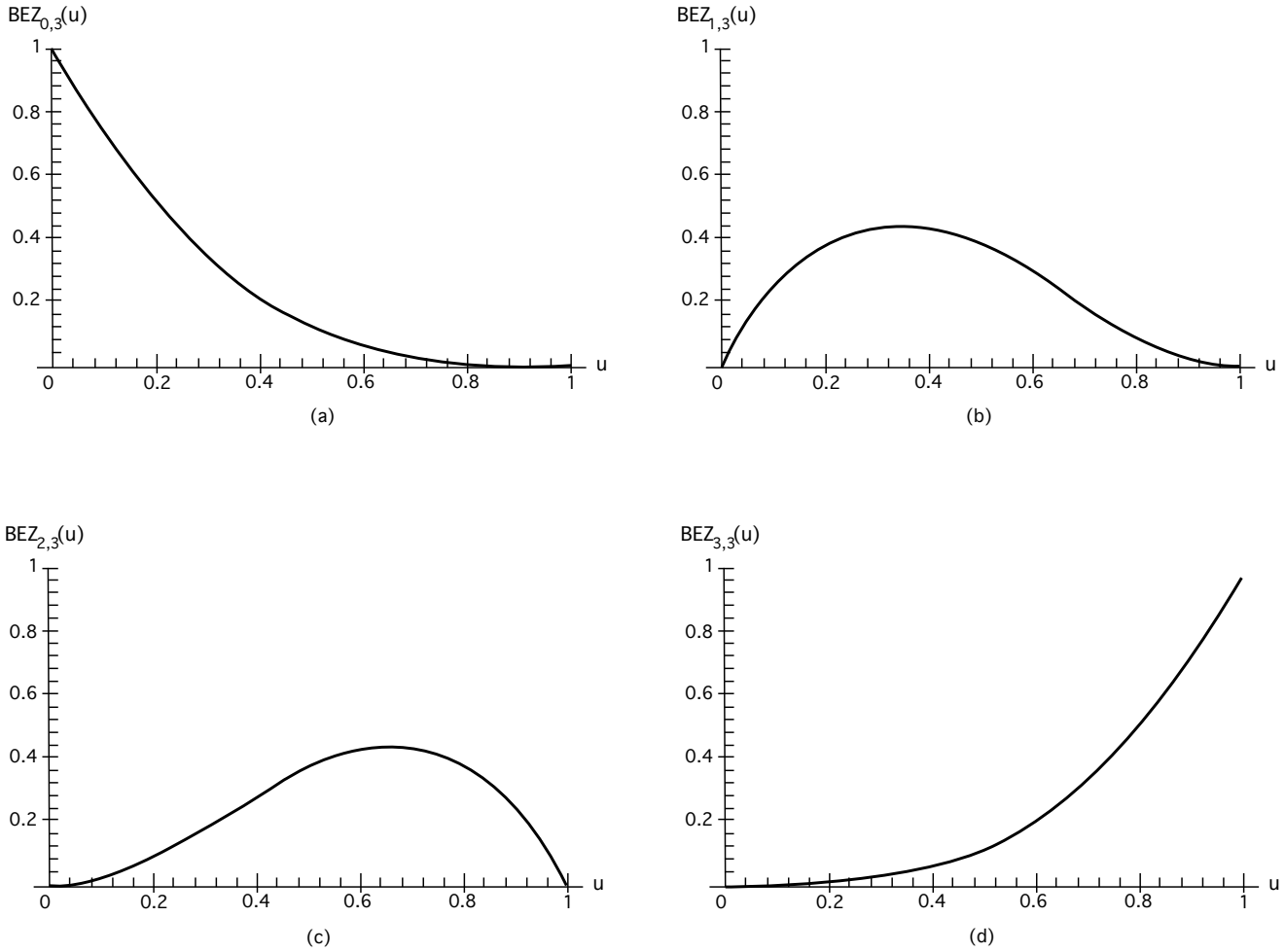


Fig. 3.65.Las cuatro funciones de combinación de Bézier para las curvas cúbicas (n = 3)

Nótese en la figura 3.65, que cada una de las cuatro funciones de combinación es no cero en el rango entero del parámetro u. Por tanto, las curvas de Bézier no permiten un control local de la forma de la curva. Si decidimos cambiar la posición de cualquiera de los puntos de control, se afectará la curva completa.

En las posiciones finales de la curva cúbica de Bezier, las primeras derivadas paramétrica (pendientes) son

$$P'(0) = 3(p_1 - 2p_0),$$
$$P'(1) = (3p_3 - p_2)$$

Y las segundas derivadas paramétricas son

$$p_{n-2} + 4(p_n - p_{n-1})$$

Al extender las expresiones polinómicas para la s funciones de combinación, podemos escribir la función de punto cúbico de Bézier en la forma matricial

$$P(u) = \begin{bmatrix} u^3 u^2 u 1 \end{bmatrix} \bullet M_{BEZ} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

donde la matriz de Bézier es

$$M_{BEZ} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

También podríamos introducir parámetros adicionales con el propósito de permitir el ajuste de la “tensión” y el “sesgo” de la curva, como lo realizamos con las splines de interpolación.

Superficies de Bézier

Se pueden utilizar dos conjuntos de curvas de Bézier ortogonales para diseñar la superficie de un objeto al especificar un entrelazado de entrada de los puntos de control. La función del vector paramétrico para la superficie de Bézier se forma como el producto cartesiano de las funciones de combinación de Bézier :

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$$

con $p_{j,k}$ especificando la ubicación de los puntos de control ($m + 1$) por ($u + 1$).

La figura 3.66 ilustra dos trazos de superficie de Bézier. Los puntos de control se conectan por medio de líneas de rayas y las líneas sólidas muestran curvas de constante u y constante v. Cada curva de constante u se traza al variar v sobre el intervalo de 0 a 1, con

u fija en uno de los valores en su intervalo de unidad. Las curvas de la constante v se trazan en forma similar.

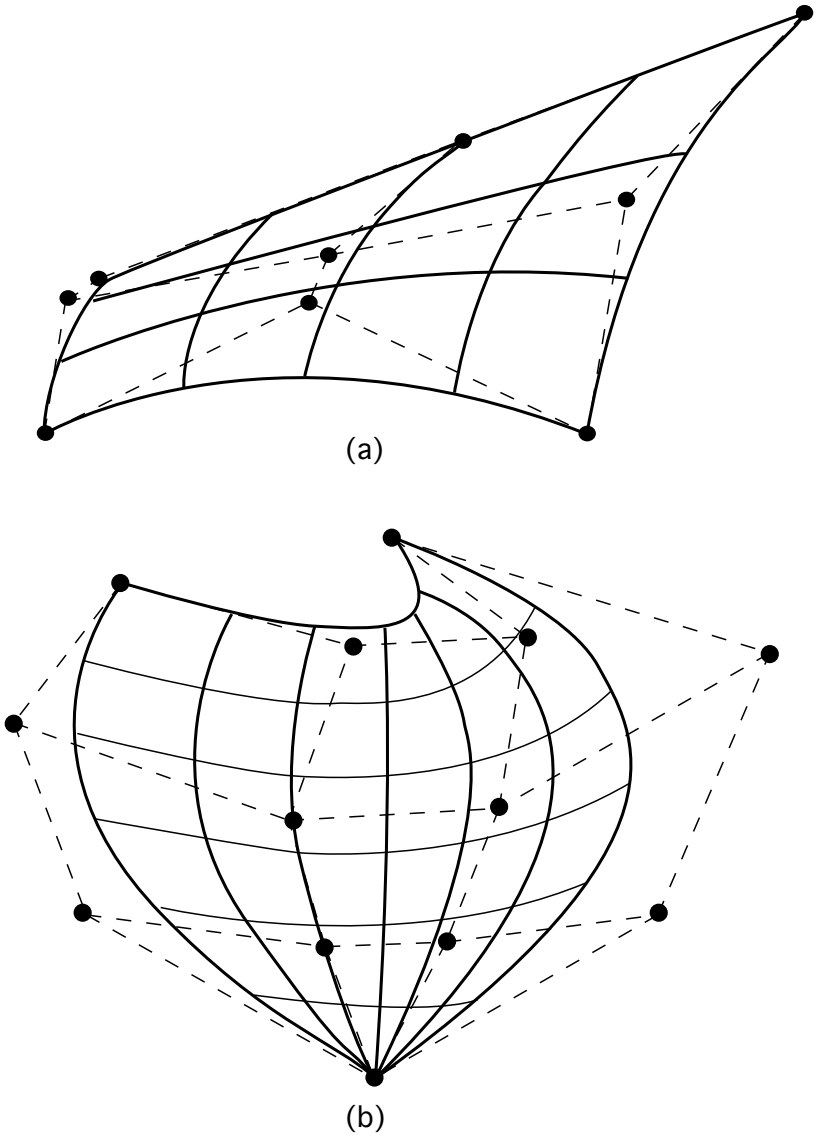


Fig. 3.66.Las superficies de Bézier que se construyen para la parte (a) m = 3, n = 3 y en la parte (b) m = 4. Los puntos de control se conectan con líneas de rayas.

Las superficies de Bézier tienen las mismas propiedades que las curvas de Bézier y proporcionan un método conveniente para las aplicaciones de diseño interactivo. Para cada parche de superficie, podemos seleccionar un entrelazado de puntos de control en el plano de terreno *xy*, así elegimos elevaciones sobre el plano de terreno para los valores de las coordenadas *z* de los puntos de control. De esta manera los parches se pueden unir al utilizar las restricciones de frontera.

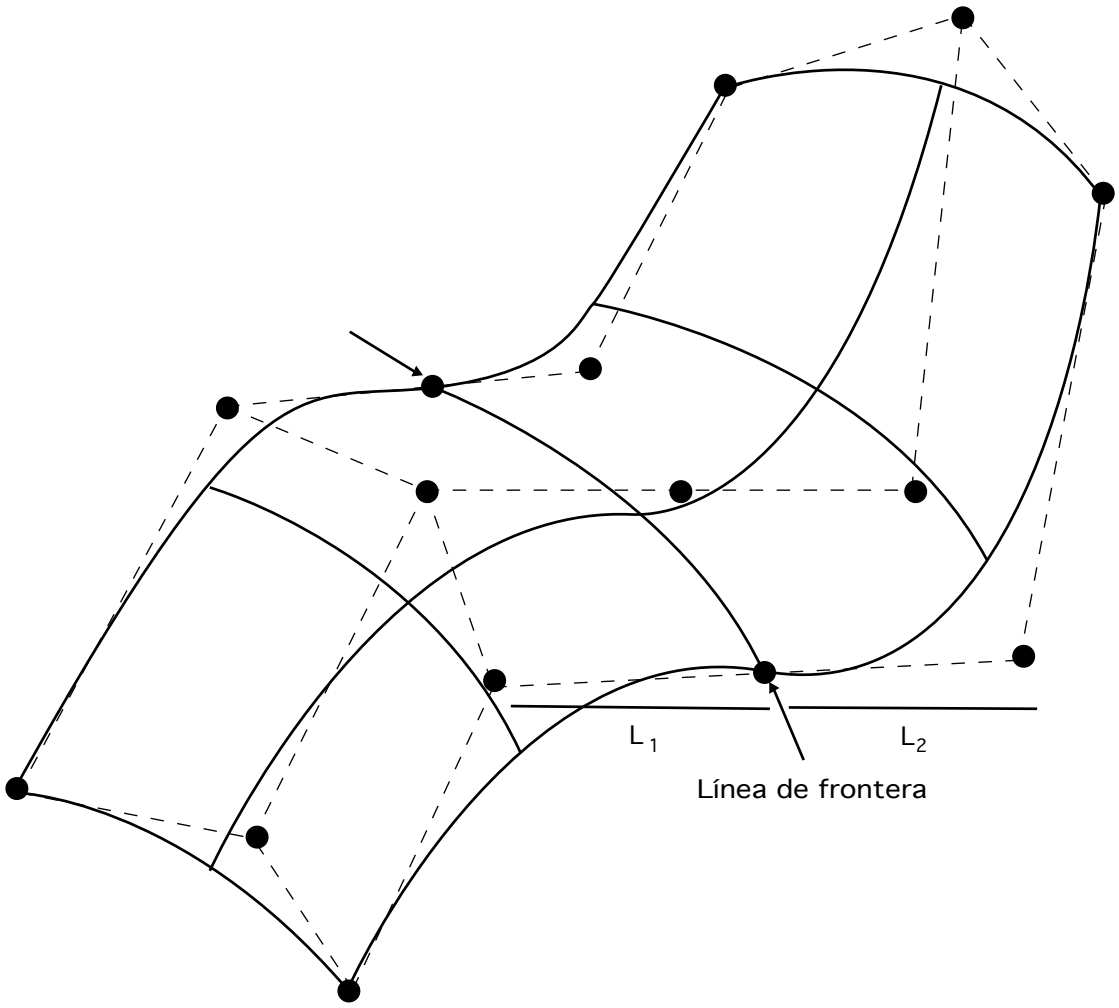


Fig. 3.67.Superficie de Bézier compuesta que se creó con dos secciones de Bézier que se unen en la línea de frontera que se indica. Las líneas de rayas conectan puntos de control específicos. Se establece la continuidad de primer orden al determinar la razón de longitud L_1 con respecto de la longitud L_2 constante para cada línea colineal de puntos de control a lo largo de la frontera entre las secciones de la superficie

La figura 3.67 ilustra una superficie que se forma con dos secciones de Bézier. Al igual que con las curvas, se asegura una transición suave de una sección a la otra al establecer tanto la continuidad de orden cero como la de primer orden en la frontera. La continuidad de orden cero se obtiene al elegir puntos de control a lo largo de una línea recta a través de la frontera y al mantener una relación constante de segmentos de línea colineales para cada conjunto de puntos de control específico a través de las fronteras de la sección.

Resumen

Se han desarrollado mucha representaciones para modelar la amplia variedad de objetos y materiales, que podríamos querer visualizar en una escena de gráficas por computadora. En la mayoría de los casos, una representación tridimensional de un objeto se crea mediante un paquete de software como un objeto gráfico estándar cuyas superficies se muestran como una malla poligonal.

La representación de frontera que más se utiliza para un objeto gráfico tridimensional es un conjunto de polígonos de superficie que encierran el interior del objeto. Muchos sistemas gráficos almacenan todas las descripciones de objetos como conjuntos de polígonos de superficie.

Una clase de objetos que se utiliza con frecuencia son las superficies cuádricas, que se describen con ecuaciones de segundo grado (cuadráticas). Incluyen esferas, elipsoides, toros, paraboloides e hiperboloides.

Una spline es una banda flexible que se utiliza para producir una curva suave a través de un conjunto de puntos designados. Varios pesos pequeños se distribuyen a lo largo de la banda para mantenerla en posición sobre la mesa de dibujo mientras se traza la curva.

En las gráficas por computadora, el término curva de spline ahora se refiere a cualquier curva compuesta que se forma con secciones polinómicas que satisfacen condiciones específicas de continuidad en la frontera de las piezas.

Pierre Bézier, ingeniero francés desarrolló el método de aproximación de splines ahora conocido como Curvas Bezier. Las splines de Bézier tienen varias propiedades que hacen que sean muy útiles y convenientes para el diseño de curvas y superficies.

Las superficies spline son una extensión bidimensional de las curvas spline. Las superficies spline se pueden entender como una curva spline en la que el lugar cada punto de control es sustituido por una curva spline del mismo tipo de la curva inicial.

El modelado sólido insiste en crear solamente modelos completos de sólidos, modelos que son adecuados para responder algoritmicamente a cualquier pregunta geométrica que se formule.

Los principales esquemas de modelado de sólido son el de representación de fronteras (Boundary representation o B-Rep) y el de geometría constructiva de sólidos (Constructive Solid Geometry o CSG).

La representación de fronteras describe el objeto en función de sus fronteras superficiales: vértices, áreas y caras. En la geometría sólida constructiva un objeto se almacena como un árbol con operadores en los nodos internos y primitivas simples en las hojas.

En general, las proyecciones transforman puntos en un sistema de coordenadas de dimensión “n” a puntos en un sistema de coordenadas con dimensión menos que n.

La proyección de objetos tridimensionales es definida por rayos de proyección rectos, llamados proyectores que emanan de un centro de proyección, pasan por cada punto del objeto e intersecan un plano de proyección para formar la proyección.

Por lo general, el centro de proyección se encuentra a una distancia finita del plano de proyección. En algunos tipos de proyecciones es conveniente pensar en función de un centro de proyección que tienda a estar infinitamente lejos.

Las proyecciones de perspectiva de cualquier conjunto de líneas paralelas que no sean paralelas al plano de conversión convergen en un punto de fuga. En el espacio tridimensional elpunto de fuga se puede considerar como la proyección de un punto en el infinito.

Las proyecciones de perspectiva se clasifican de acuerdo con el número de puntos de fuga principales y por ende, con respecto al numero de ejes que corta el plano de proyección.

Las proyecciones paralelas se clasifican en dos tipos, dependiendo de la relación etre la dirección de la proyección y la normal al plano de proyección. En las proyecciones paralelas ortográficas, estas direcciones son las mismas (o en sentido contrario): de manera que la dirección de la proyección es normal al plano de proyección.

Los tipos más comunes de proyecciones ortográficas son la de relación frontal, elevación superior o elevación de plano y la de elevación lateral. En todas ellas, el plano de proyección es perpendicular al eje principal, que por lo tanto es la dirección de la proyección

Las proyecciones ortográficas axonométricas usan planos de proyección que no son normales a un eje principal y que por ende muestran varias caras de un objeto al mismo tiempo.

Las proyecciones oblicuas, la segunda clase de proyecciones paralelas, difieren de las proyecciones ortográficas en que la normal al plano de proyección y la dirección de la proyección son diferentes. Las proyecciones oblicuas combinan las propiedades de las proyecciones ortográficas frontal, superior y lateral con las de una proyección axonométrica: el plano de proyección es normal a un eje principal, de manera que la proyección de la cara del objeto paralela a este plano permite medir ángulos y distancias.

La proyección isométrica es una proyección axonométrica de uso común. La normal al plano de proyección (y por consiguiente la dirección de la proyección) forma ángulos iguales con respecto a cada eje principal.

Bibliografía

Buss, Samuel R. 3-D computer Graphics. California, USA. Cambridge University Press.

Demel , John T./ Michael J. Miller. Gráficas por computadora. México, D.F. Ed. Mc. Graw Hill.

Foley , James D./ Andries Van Dam. Introducción a la graficación por computadora. México, D.F. Ed. Addison Wesley Iberoamericana.

Hearn, Donald / Baker, M. Pauline. Graficas por computadora 2a edición. México, D.F. Ed. Prentice Hall Hispanoamericana.

Hill, F. S. Jr. Computer Graphics Using Open Gl. USA. Ed. Prentice-Hall.

Paluszny Marco et. al. Métodos de Bézier y B-Splines. USA. Ed. Springer-Verlag.

Parent, Richard. Computer Animation: Algorithms and Techniques. Ed. Morgan Kauffman.

Pavlidis, Theo. Algorithms for Graphics and Image processing. USA. Computer Science Press.

Preparata, Franco P. Computational Geometry. USA. Ed. Springer-Verlag

Ramos, Ricardo. Informática Gráfica. Universidad de Oviedo. Oviedo, España.

Rogers , David .F. Procedural Elements of Computer Graphics. 2a edición. USA. Ed. Mc Graw Hill.

Shneider, Philip / Eberly, David H. Geometric tools for computer graphics. California, USA., Morgan Kaufmann Publishers.

Torres, J.C., Diseño asistido por ordenador. Granada, España. Universidad de Granada.

Universidad de Valencia. Informática Gráfica. Valencia, España.

Universidad de Vigo. Introducción a las gráficas por computadora. Vigo, España.

Universitat Jaume I de Castelló. Diseño y Fabricación Asistido por Ordenador. Valencia, España.

Universidad Politécnica de Valencia. Tecnicas Gráficas. Valencia, España.

Vince, John. Mathematics for computer graphics. 2a edición. USA. Springer - Verlag.

Wikipedia. Formatos Gráficos. http://es.wikipedia.org. Enero de 2007.

Wright, Richard S. OpenGLOrg. OpenGL SuperBible. Estados Unidos.

Apéndice A
Introducción a OpenGL

En OpenGL se proporciona una biblioteca básica de funciones para especificar primitivas gráficas, atributos, transformaciones geométricas, transformaciones de visualización y muchas otras operaciones. Está diseñada para ser independiente del hardware, por lo tanto, muchas operaciones como las subrutinas de entrada y salida, no están incluidas en la biblioteca básica.

Sin embargo, las subrutinas de entrada y salida y muchas funciones adicionales están disponibles en bibliotecas auxiliares que se han desarrollado para programas OpenGL.

A.1. Sintaxis básica de OpenGL

Los nombres de las funciones de la biblioteca básica de OpenGL (también llamada de biblioteca del núcleo de OpenGL utilizan como prefijo GL y cada palabra que forma parte del nombre de una función tiene su primera letra en mayúscula. Los siguientes ejemplos ilustran este convenio de denominación.

```
glBegin, glClear, glCopyPixels, glPolygonMode
```

Algunas funciones requieren que a uno (o más) de sus argumentos se les asigne una constante simbólica al especificar por ejemplo. un nombre de parámetro, un valor para un parámetro o un modo particular. Todas estas constantes comienzan con las letras GL en mayúsculas. Además, las palabras que forman parte de una constante con nombres se escriben en mayúsculas y el guión bajo (_) se utiliza como separador entre todas estas palabras del nombre. Los siguientes son unos pocos ejemplos de los varios cientos de constantes simbólicas disponibles para uso con las funciones de OpenGL.

`GL_2D, GL_RGB, GL_CCW, GL_POLYGON, GL_AMBIENT AND DIFFUSE`

Las funciones de Openl también esperan tipos de datos específicos. Por ejemplo, un parametro de una función de OpenGL podría esperar un valor que se especifica como un entero de 32 bits. Pero el tamaño de la especificación de un entero puede ser diferente en las distintas máquinas. OpenGL tiene incorporada una serie de nombres para tipos de datos específicos tales como

`Glbyte, GLshort, GLint, GLfloat, GLdouble, GLboolean`

Cada nombre de tipo de datos comienza con las letras mayúsculas GL y, a continuación, un identificador de tipo de datos estándar, escrito con letras minúsculas.

A algunos argumentos de funciones de OpenGL se les puede asignar valores empleando una matriz que enumera un conjunto de valores de datos, Esta opción se utiliza para especificar una lista de valores como un puntero a una matriz, en lugar de especificar cada elemento de la lista explícitamente como un argumento, Un ejemplo típico del uso de esta opción es la especificación de los valores de las coordenadas xyz.

A.2. Bibliotecas relacionadas

Existe un gran número de bibliotecas relacionadas para la realización de operaciones especiales, además de la biblioteca básica de OpenGL. La utilidad GLU (OpenGL Utility) proporciona subrutinas para la configuración de las matrices de visualización y proyección, descripción de objetos complejos mediante líneas y aproximaciones poligonales, visualización de cuádricas y splines B empleando aproximaciones lineales, procesamiento de operaciones de representación de superficies y otras tareas complejas. Toda implementación de OpenGL incluye la biblioteca GLU. Todos los nombres de las funciones GLU comienzan con el prefijo `glu.`

También existe un conjunto de herramientas orientadas a objetos basado en OpenGL llamado Open Inventor que proporciona subrutinas y formas de objetos predefinidos para su uso en aplicaciones tridimensionales interactivas. Este conjunto de herramientas está escrito en C++.

Para crear gráficos utilizando OpenGL, necesitamos en primer lugar configurar una ventana de visualización en nuestra pantalla de vídeo. Se trata simplemente de la zona rectangular de la pantalla en la nuestra imagen se mostrará. No podemos crear directamente la ventana de visualización con las funciones de OpenGL básicas, ya que esta biblioteca contiene unicamente funciones gráficas independientes del dispositivo, y las operaciones de gestión de ventanas dependen de la computadora que estemos utilizando. Sin embargo, existen varias bilbiotecas de sistema de ventanas que soportan las funciones de OpenGL en una gran variedad de máquinas.

La ampliación de OpenGL al sistema de ventanas X (GLX, OpenGL Extension to the X Window System) proporciona un conjunto de subrutinas que utilizan como prefijo las letras `glx`. Los sistemas Apple pueden utilizar la interfaz para operaciones de gestión de ventanas Apple GL (AGL). Los nombres de las funciones de esta biblioteca utilizan como prefijo `agl`.

En los sistemas que utilizan Microsoft Windows, las subrutinas WGL proporcionan una interfaz de Windows a OpenGL. Estas subrutinas utilizan como prefijo las letras wgl. El gestor PGL (Presentation Manager to OpenGL) es una interfaz para el sistema operativo OS/2 de IBM, que utiliza el prefijo `pgl` en las subrutinas de la biblioteca. Y el kit de herramientas GLUT (OpenGL Utility Toolkit) proporciona una biblioteca de funciones para interactuar con cualquier sistema de ventanas. Las funciones de la biblioteca GLUT utilizan como prefijo `glut`. Esta biblioteca también contiene métodos para describir y representar superficies y curvas cuádricas.

Ya que GLUT es una interfaz con otros sistemas de ventanas dependientes del dispositivo, podemos utilizar GLUT para que nuestros programas sean independientes del dispositivo.

A.3. Archivos de cabecera

En todos nuestros programas gráficos, necesitaremos incluir el archivo de cabecera para la biblioteca del núcleo de OpenGL. En la mayoría e las aplicaciones también necesitaremos GLU. Y necesitaremos incluir el archivo de cabecera para el sistema de ventanas. Por ejemplo, en Microsoft Windows, el archivo de cabecera para acceder a las subrutinas de WGL es windows.h. Este archivo de cabecera se debe indicar antes de los

archivos de cabecera de OpenGL y GLU, ya que contiene macros que se necesitan en la versión de la biblioteca de OpenGL para Microsoft Windows. Por tanto, el archivo fuente en este caso debería comenzar con

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
```

Sin embargo si utilizamos GLUT para gestionar las operaciones de gestión de ventanas, no necesitaremos incluir `gl.h` y `glu.h`, porque GLUT garantiza que estos archivos se incluirán correctamente. Por tanto, podemos reemplazar los archivos de cabecera de OpenGL y GLU por

```
#include <GL/glut.h>
```

También podríamos incluir `gl.h` y `glu.h`, pero al hacerlo seríamos redundantes y se podría ver afectada la portabilidad del programa.

Además, a menudo necesitaremos incluir archivos de cabecera que eel código C++ requiere. Por ejemplo,

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

Con el nuevo estándar ISO/ANSI para C++, estos archivos de cabecera se denominan `cstudio`, `cstdlib` y `cmath`.

A.4. Gestión de la ventana de visualización empleando GLUT

Para comenzar, podemos considerar un número mínimo y simplificado de operaciones para mostrar una imagen. Ya que estamos empleando la utilidad GLUT (OpenGL Utility Toolkit), nuestro primer paso consiste en inicializar GLUT. Esta función de inicialización podría también procesar argumentos cualesquiera de la línea de comandos, pero no necesitaremos utilizar estos parámetros en nuestro primer ejemplo de programa.

Realizamos la inicialización de GLUT con la siguiente línea:

```
glutInit(&argc, &argv);
```

A continuación, podemos indicar que se cree una ventana de visualización en la pantalla con un título en su barra de título. Esto se realiza con la función

```
glutCreateWindow("Un programa de ejemplo con OpenGL");
```

Donde el único argumento de esta función puede ser cualquier cadena de caracteres que queramos utilizar como título de la ventana de visualización.

Ahora hay que especificar que va a contener la ventana de visualización. Para ello, creamos una imagen empleando las funciones de OpenGL y se pasa la definición de la imagen a la subrutina llamada `glutDisplayFunc`, que asigna nuestra imagen a la ventana de visualización. Como ejemplo, suponga que disponemos del código de OpenGL para describir un segmento en un procedimiento llamado `lineSegment`. Entonces la siguiente llama a función pasa la descripción del segmento a la ventana de visualización.

```
glutDisplayFunc(lineSegment);
```

Pero la ventana de visualización no está aún en la pantalla. Necesita una función más de GLUT para completar las operaciones de procesamiento de ventana. Después de la ejecución de la siguiente línea, todas las ventanas de visualización que hayas creado, incluyendo su contenido gráfico, se activarán.

```
glutMainLoop();
```

Esta función debe ser la última en nuestro programa. Ésta muestra los gráficos iniciales y pone el programa en un bucle infinito que comprueba la entrada procedente de dispositivos, tales como un ratón o un teclado. Nuestro primer ejemplo no será interactivo, por lo que el programa únicamente continuará mostrando la imagen hasta que cerremos la ventana de visualización.

Aunque la ventana de visualización que creamos tendrá una posición y un tamaño predeterminados, podemos establecer estos parámetros empleando funciones adicionales del kit de herramientas GLUT. Utilizamos la función glutInitWindowPosition para porporcionar una posición inicial para la esquina superior izquierda de la ventana de visualización. Esta

posición se especifica en coordenadas enteras de pantalla, cuyo origen está en la esquina superior izquierda de la pantalla.

Por ejemplo, la siguiente línea especifica que la esquina superior izquierda de la ventana de visualización se debería colocar 50 pixeles a la derecha del borde izquierdo de la pantalla y 100 pixeles hacia abajo desde el borde superior de la pantalla.

```
glutInitWindwPosition(50,100);
```

Análogamente, la función glutInitWindowsSize se utiliza para establecer la anchura y la altura en pixeles de la ventana de visualización. Por tanto, especificamos una ventana de visualización con una anchura inicial de 400 pixeles y una altura de 300 pixeles con la línea siguiente,

```
glutInitWindowsSize(400,300);
```

Después de que la ventana de visualización esté en pantalla, podemos volver a cambiar tanto su posición como su tamaño.

También podemos establecer otras opciones de la ventana de visualización, tales como los búferes y una opción de los modos de color, con la función glutInitDisplayMode. Los argumentos de esta subrutina se asignan mediante constantes simbólicas GLUT. Por ejemplo, la siguiente orden especifica que se utilice un único buffer de refresco en la ventana de visualización y que se utilice el modo de color RGB (rojo, verde, azul) para seleccionar los valores de los colores.

```
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

Los valores de las constantes que se pasan a esta función se combinan empleando la operación lógica *or*. En realidad, el buffer único y el modo de color RGB son opciones predeterminadas. Pero usaremos esta función para recordar que se han establecido para nuestra visualización.

A.5. Un programa OpenGL completo

Todavía hay que realizar algunas tareas antes de que tengamos todas las partes necesarias para un programa completo. En la ventana de visualización podemos elegir un

color de fondo. Y necesitamos construir un procedimiento que contenga las funciones apropiadas de OpenGL para la imagen que queremos mostrar en pantalla.

Mediante el empleo de valores de color RGB, establecemos que el color de la ventana de visualización sea blanco con la función de OpenGL

```
glClearColor(1.0, 1.0, 1.0, 1.0, 0.0);
```

Los tres primeros argumentos de esta función establecen cada una de las componentes de color roja, verde y azul en el valor de 1.0. Por tanto, obtenemos el color blanco en la ventana de visualización. Si, en lugar de 1.0, establecemos cada una de las componentes de color en 0.0, obtendríamos el color negro como color de fondo. Y si a cada una de las componentes rojs, verde y azul se les asigna un valor intermedio entre 0.0 y 1.0 obtendríamos algún nivel de gris.

El cuarto parámetro de la función `glClearColor` se denomina valor alfa del color especificado. Un uso del valor alfa es el parámetro de << fundido >> (*blending*). Cuando activamos las operaciones de fundido de OpenGL, los valores alfa se pueden utilizar para calcular el valor resultante de dos objetos que se superponen. Un valor alfa de 0.0 indica que el objeto es totalmente transparente, mientras que el valor alfa de 1.0 indica que el objeto es opaco. Por ahora, establecemos simplemente el valor alfa en 0.0.

Aunque la orden `glClearColor` asigna un color a la ventana de visualización, ésta no muestra la ventana de visualización en la pantalla. Para conseguir que el color asignado a la ventana se visualice, necesitamos invocar la siguiente función de OpenGL.

```
glClear(GL_COLOR_BUFFER_BIT);
```

El argumento `GL_COLOR_BUFFER_BIT` es una constante simbólica que especifica que son los valore de los bits del buffer de color (buffer de refresco) los que se deben asignar a los valores indicados en la función `glClearColor`.

Además de establecer el color de fondo de la ventana de visualización, podemos elegir entre una gran variedad de esquemas de color para los objetos que queremos mostrar en una escena. En nuestro ejemplo inicial de programación, establecemos simplemente el color del objeto en rojo.

```
glColor3f(1.0, 0.0, 0.0);
```

El sufijo `3f` de la función glColor indica que especificamos las tes componentes de color RGB mediante el empleo de valores de punto flotante (f). Estos valores se deben encontrar dentro del rango comprendido entre 0.0 y 1.0, y hemos establecido la componente roja en 1.0 y las componentes verde y azul en 0.0

En nuestro primer programa, simplemente mostramos un segmento bidimensional. Para ello, necesitamos decir a OpenGL como queremos "proyectar" nuestra imagen en la ventana de visualización, porque la generación de una imagen bidimensional se trata en OpenGL como un caso especial de la visualización tridimensional. Por lo que, aunque solo queramos producir una línea muy simple bidimensional, OpenGL procesa a imagen mediante todas las operaciones de visualización tridimensional. Podemos establecer que el tipo de proyección (modo) y otros parámetros de visualización que necesitemos con las dos funciones siguientes:

```
glMatrizMode(GL_PROJECTION);
gluORTHO2D(0.0, 200.0, 0.0, 150.0);
```

Esto especifica que se debe utilizar una proyección ortogonal para mapear los contenidos de una zona rectangular bidimensional (2D) de las coordenadas universales a la pantalla, y que los valores de la coordenada *x* dentro de este rectángulo varían desde 0.0 hasta 200.0 y que los valores de la coordenada *y* varían desde 0.0 hasta 150.0.

Objetos cualesquiera que definamos dentro de este rectángulo de coordenadas universales se mostrarán dentro de la pantalla de visualización. Cualquier cosa fuera de este rango de coordenadas universales se mostrarán dentro de la ventana de visualización. Cualquier cosa fuera de este rango de coordenadas no se visualizará. Por tanto, la función de GLU `gluOrtho2D` establece que el sistema de coordenadas de referencia dentro de la ventana de visualización deberá tener las coordenadas (0.0 , 0.0) en la esquina inferior izquierda de la ventana de visualización y (200.0 y 150.0) en la esquina superior izquierda de la ventana.

Ya que solo describimos un objeto bidimensional, el único efecto que tiene la proyección ortogonal es "pegar" nuestra imagen en la ventana de visualización definida anteriormente. Por ahora utilizaremos un rectángulo de coordenadas universales con la

misma relación de aspecto que la ventana de visualización, para que no haya distorsión en nuestra imagen.

Finalmente, necesitamos llamar las subrutinas apropiadas de OpenGL para crear nuestro segmento. El código siguiente define un segmento bidimensional definiendo sus extremos con coordenadas cartesianas enteras de valores (180, 15) y (10, 145).

```
glBegin (GL_LINES)
    glVertex2i(180,15);
    glVertex2i(10, 145);
glEnd();
```

Ahora es el momento e reunir todas las piezas. El siguiente programa de OpenGl está organizado en tres procedimientos. Colocamos todas las inicializaciones y los parámetros de configuración relacionados en el procedimiento `init`. Nuestra descripción geométrica de la "imagen" que queremos visualizar está en el procedimiento `lineSegment`, que es el procedimiento que será referenciado por la función de GLUT `glutDisplayFunc`. Y el procedimiento `main` contiene las funciones de GLUT que configuran la ventana de visualización y que muestran nuestro segmento en la pantalla.

A continuación se muestra la ventana de visualización y el segmento generado por este programa.

```
#include<GL.glut.h> //u otras líneas dependiendo del sistema que usemos

void init(void){
    glClearColor(1.0, 1.0, 1.0, 0.0); //Establece el color de la ventana
                                      // de visualización en blanco
    glMatrixMode(GL_PROJECTION);//Establece los parámetros de proyección
    gluOrtho2D(0.0, 200.0, 0.0, 15.0);
}

void lineSegment(void){
    glClear(GL_COLOR_BUFFER_BIT); //Borra la ventana de visualización
    glColor3f(1.0, 0.0, 0.0); //Establece el color del segmento de línea
                              //en rojo
    glBegin(GL_LINES)
        glVertex2i(180, 15); //Especifica la geometría del segmento
```

```
                                // de línea
        glVertex2i(10, 145);
    glEnd();
    glFlush();   //Procesa todas subrutinas de OpenGL tan rápidamente como
                 //sea posible
}

void main(int argc, char ** argv){
    glitInit(&argc, argv); //Inicializa GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);   //Establece el modo de
                                                   // visualización
    glutInitWindowPosition(50,100); // Establece la posición de la esquina
                    //superior izquierda de la ventana de visualización
    glutInitWindowSize(400,300);  //Establece el ancho y la altura de la
                                  //ventana de visualización
    glutCreateWindow("Ejemplo de programa en OpenGL") //Crea la ventana
    init();      //Ejecuta el procedimiento de inicialización
    glutDisplayFunc(lineSegment); //Envía los gráficos a la ventana de
                                  //visualización
    glutMainLoop();               //Muestra todo y espera
}
```

Al final del procedimiento `lineSegment` hay una función, `glFlush`, que todavía no se ha mencionado. Es simplemente una subrutina que fuerza la ejecución de nuestras funciones de OpenGL, las cuales almacenan las computadoreas en buffers en diferente posiciones, dependiendo de cómo esté implementada OpenGL. En una red ocupada, por ejemplo, podría haber retrasos en el procesamiento de algunos buffers. Pero la llamada a `glFlush` fuerza a que todos estos buffers se vacíen y que las funciones de OpenGL se procesen.

El procedimiento lineSegment que hemos creado para descrbiir nuestra imagen se denomina *función de respuesta a la visualización (display callback function)*. Este procedimiento lo "registra" `glutDisplayFunc` como la subrutina que se invoca siempre que sea preciso mostrar la ventana de visualización de nuevo. Esto puede ocurrir, por ejemplo, si se mueve la ventana de visualización. Por lo general, los programas que utilizan OpenGL se organizan como un conjunto de funciones de respuesta que se invocan cuando ocurren determinadas acciones.

A.6. Instalación de OpenGL

Es recomendable que una vez terminada la práctica se examine el Apéndice A, el cual contiene una introducción a la programación en OpenGL.

OpengGL en Windows

Las librerías de OpenGL para Windows fueron implementadas en un inicio para Windows NT y posteriormente en otras versiones del sistema operativo. Existen problemas de seguridad, errores de compilación, de tiempo de ejecución y fallos en el sistema los cuales no han sido documentados, por lo tanto el usuario es responsable del uso que se le de al software. Las siguientes instrucciones no funcionaran en Windows Vista.

Es conveniente señalar que los ejemplos, ejercicios y prácticas señaladas en este texto han sido probadas únicamente en sistemas UNIX, LINUX, BSD y Mac OS X.

Para comenzar la programación es necesario instalar GLUT (véase Apéndice A). GLUT puede usarse con Visual C++ así como con algunos productos de Borland.

Paso 1: Instalar GLUT

El primer paso es descargar GLUT, puede hacerse desde la siguiente página:

http://www.opengl.org/resources/libraries/glut/glut_downloads.php#windows

Los detalles de la instalación y compilación pueden observarse en el Léame o Readme disponible en la siguiente página:

http://www.xmission.com/~nate/glut.html

Después de instalar, se deben mover algunos archivos como a continuación se indica:

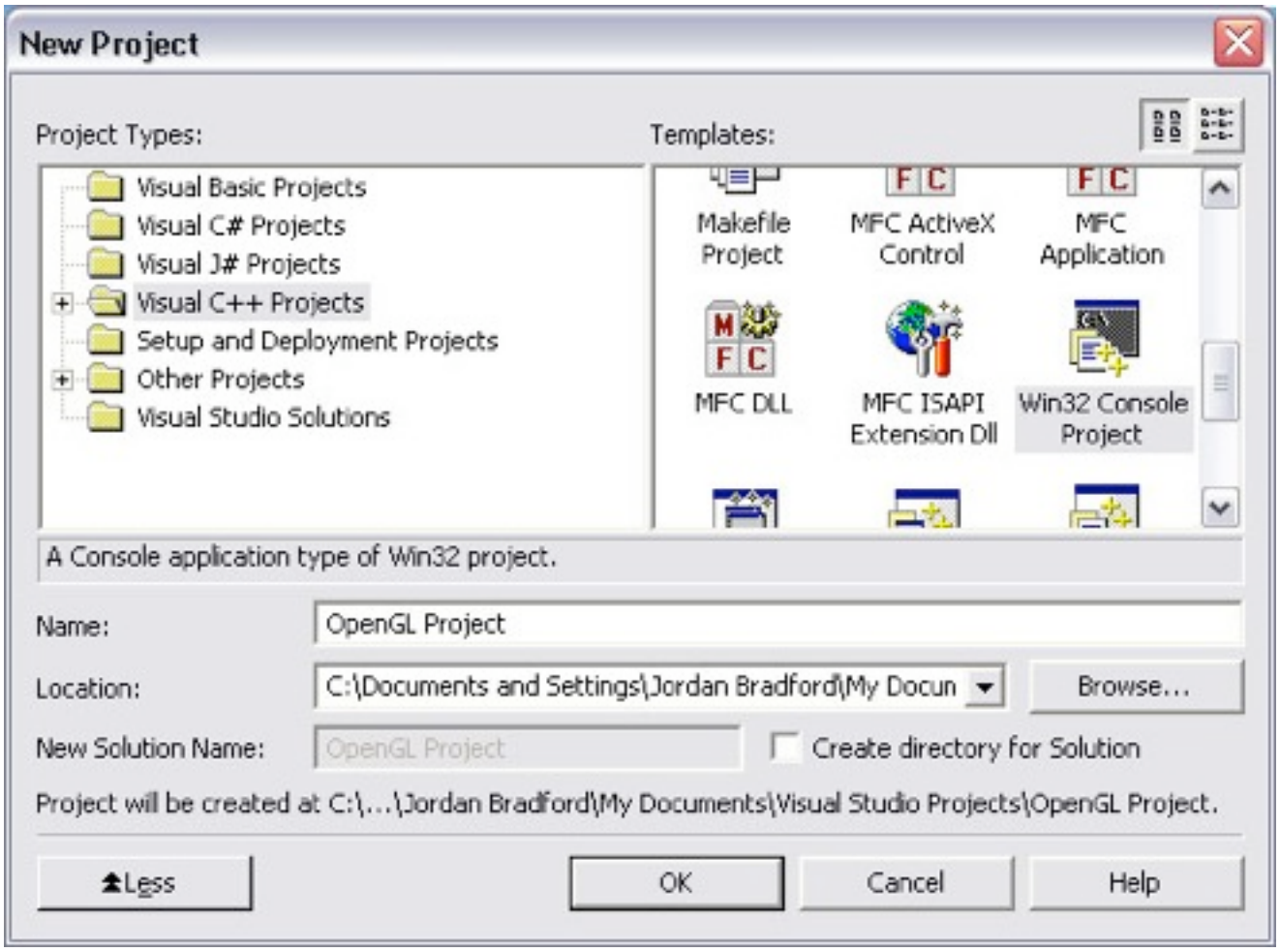
Archivo	Ubicación
glut32.dll	Windows XP	Server 2003: C:\WINDOWS\system\
	Windows 2000: C:\WINNT\system\
glut32.lib	C:\Program Files\Microsoft Visual Studio NET 2003\Vc7 \PlatformSDK\Lib
glut.h	C:\Program Files\Microsoft Visual Studio NET 2003\Vc7 \PlatformSDK\Include\gl

Si se desea utilizar cualquier aplicación o programa creado con OpenGL en una computadora diferente a la que fue compilado, deberá incluirse el archivo *glut32.dll* en la ruta indicada arriba, de lo contrario no funcionará.

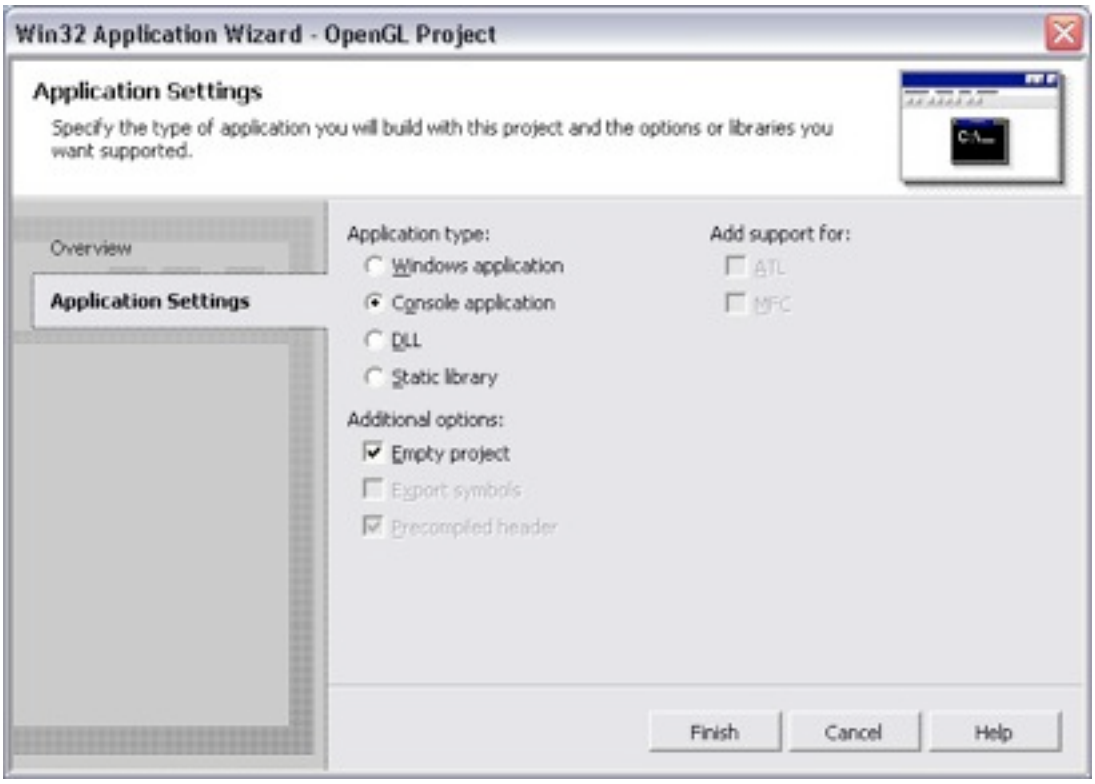
Paso 2: Crear un proyecto de Visual Studio

Debido a que GLUT está diseñado para ser independiente del sistema de ventanas, GLUT debe correr como una aplicación de consola en vez de correr como una aplicación nativa de Windows.

4. Crea un proyecto nuevo (Archivo -> Nuevo -> Proyecto). Aparecerá un cuadro de diálogo.
5. En el panel *Tipos de Proyecto* (Project Types) selecciona *Proyectos Visual C++* (Visual C++ Projects). Luego en el panel de plantillas selecciona *Proyecto de Consola Win32* (Win32 Console Project). Dale nombre al proyecto y dale clic en Aceptar. Aparecerá el *Wizard* de aplicaciones Win32.



En la parte izquierda de la pantalla da clic en el tab *Propiedades de la aplicación* (Application Settings), y marca el checkbox *Proyecto vacío* y da clic en Terminar.



Paso 3. Añadir código fuente

Se asume que ya se poseen conocimientos previos de Visual Studio, así que se no es necesario dar una explicación detallada sobre la adición de código fuente.

Cuando se incluye GLUT en un programa, automáticamente se incluyen otros archivos cabecera de Open GL. Escribir explícitamente:

```
#include <GL/gl.h>
#include <GL/glu.h>
```

No es necesario, solamente se debe escribir:

```
#include <GL/glut.h>
```

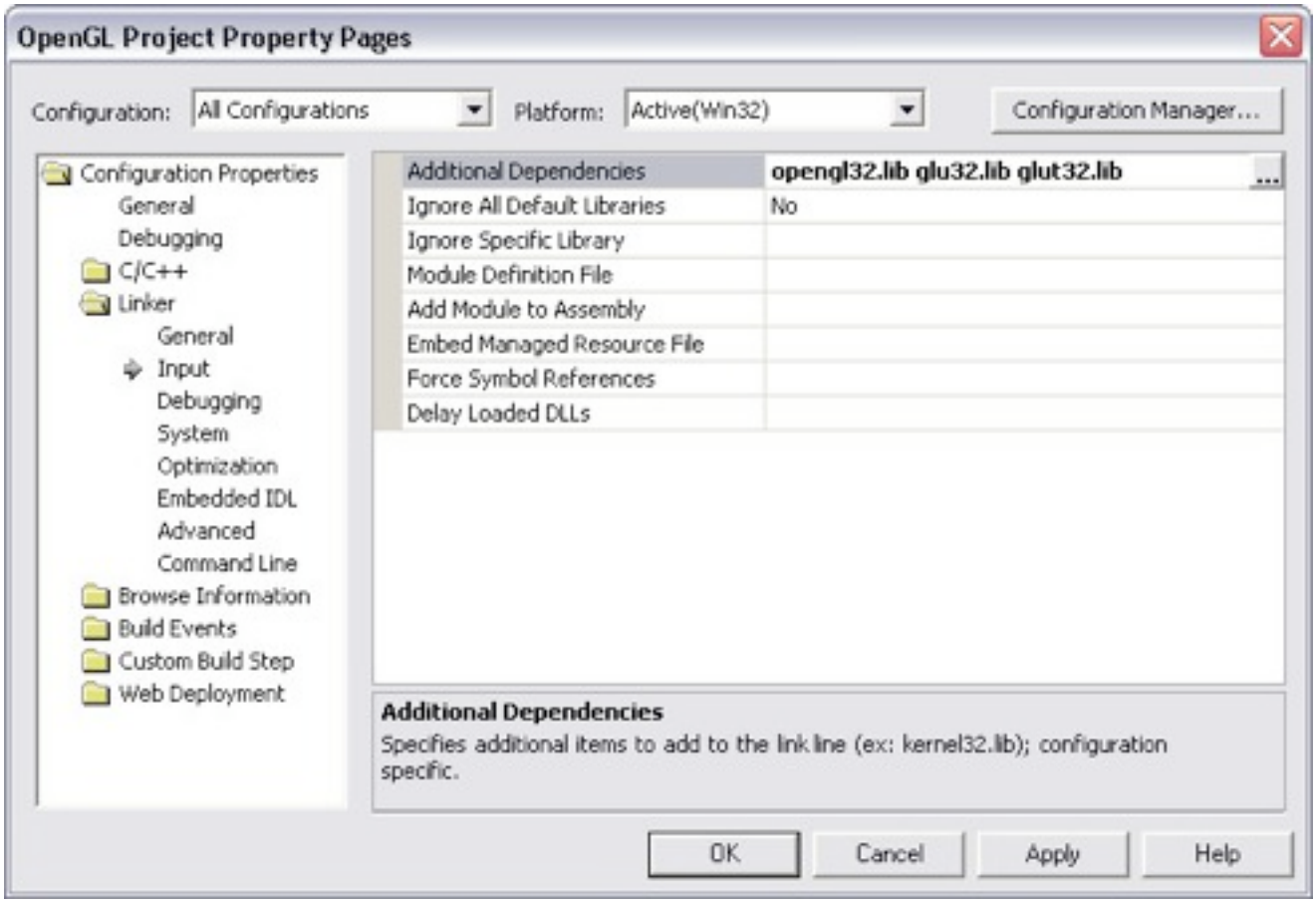
No debe utilizarse la diagonal invertida (\). Visual Studio compilara el programa pero los

compiladores estándar no lo harán.

Paso 4: Modificar las propiedades del proyecto

Antes de compilar el proyecto, se le debe hacer un link en visual Studio para que sepa donde encontrar GLUT. Para hacer esto, se debe abrir el diálogo de *propiedades del proyecto* (Proyecto -> Propiedades)

De la lista desplegable *Configuración* selecciona *Todas las configuraciones*. En el panel izquierdo, selecciona *Linker* y luego la opción *Entrada* (Input). Añade el siguiente código al cuadro de texto Dependencias Adicionales (Additional Dependencies). Copia y pega: opengl32.lib glu32.lib glut32.lib



Después de dar clic en Aceptar puedes empezar a programar.

OpenGL en Linux

Para instalar la librería GLUT en Linux deberás entrar a la consola y teclear lo siguiente:

Ubuntu	`$sudo apt-get install libglut3 libglut3-dev`
Debian	`#apt-get install libglut3 libglut3-dev`
Fedora, Mandriva, RedHat	`yum install libglut3 libglut3-dev`

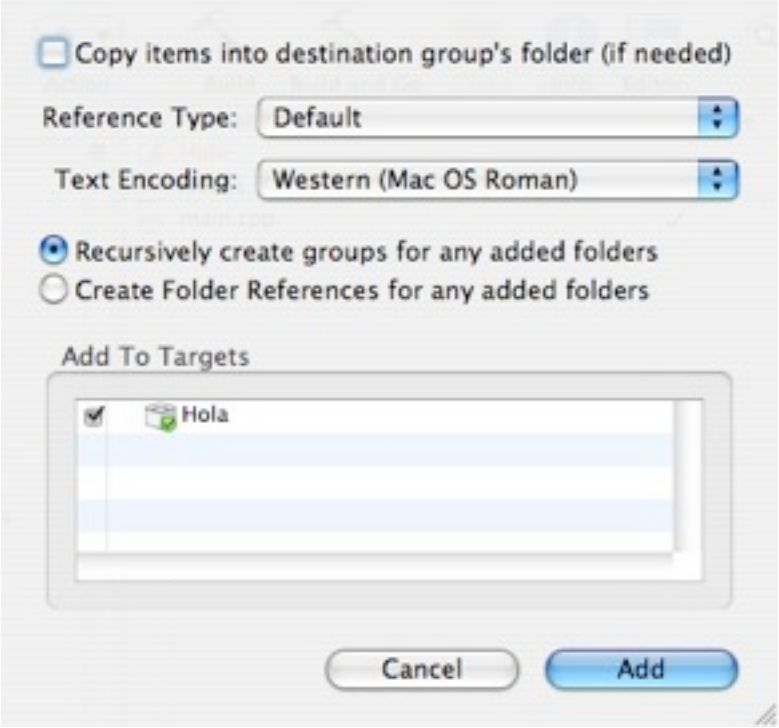
Hecho esto, se puede comenzar a programar.

OpenGL en Mac

La herramienta para Mac Xcode ya incluye la librería GLUT. Xcode viene incluido de forma gratuita dentro de los CD de instalación del sistema operativo Mac. Para utilizar OpenGL unicamentese tiene que añadir el *Framework* al proyecto.

Primero se crea una aplicación *C++ Tool*, luego se abre la ventana para añadir *Frameworks* (Project ->Add to Project) y se selecciona el framework GLUT (Se encuentra en HD -> Developer-> SDKs -> MacSDK ->System -> Library -> Frameworks -> GLUT.framework).

Aparecerá una ventana como la siguiente



Se selecciona la opción *Copy items...* si se desea que se copie el framework a la carpeta del proyecto. Dar clic en *Add*.

En el archivo main.cpp añadir al inicio:

```
#include <GLUT/glut.h>
```

Ahora se puede comenzar a programar.

Apéndice B
Matemáticas para Graficación

B.1. Sistemas de coordenadas

Tanto los sistemas de referencias cartesianos como los no cartesianos resultan útiles en las aplicaciones gráficas. Normalmente, especificamos las coordenadas en un programa gráfico utilizando un sistema de referencia cartesiano, pero la especificación inicial de una escena podría proporcionarse en un sistema de referencia no cartesiano. A menudo, las simetrías escénicas, cilíndricas o de otros tipos pueden aprovecharse para simplificar las expresiones relativas a las descripciones o manipulaciones de los objetos.

Coordenadas de pantalla bidimensionales

Para los comandos independientes del dispositivo incluidos dentro de un paquete gráfico, las coordenadas de pantalla se referencian dentro del primer cuadrante de un sistema cartesiano bidimensional en posición estándar (figura B.1. (a)). El origen de las coordenadas de este sistema de referencia se encuentra situado en la esquina inferior izquierda de la pantalla, por lo que las posiciones de pantalla están representadas internamente con respecto a la esquina superior izquierda de la misma.

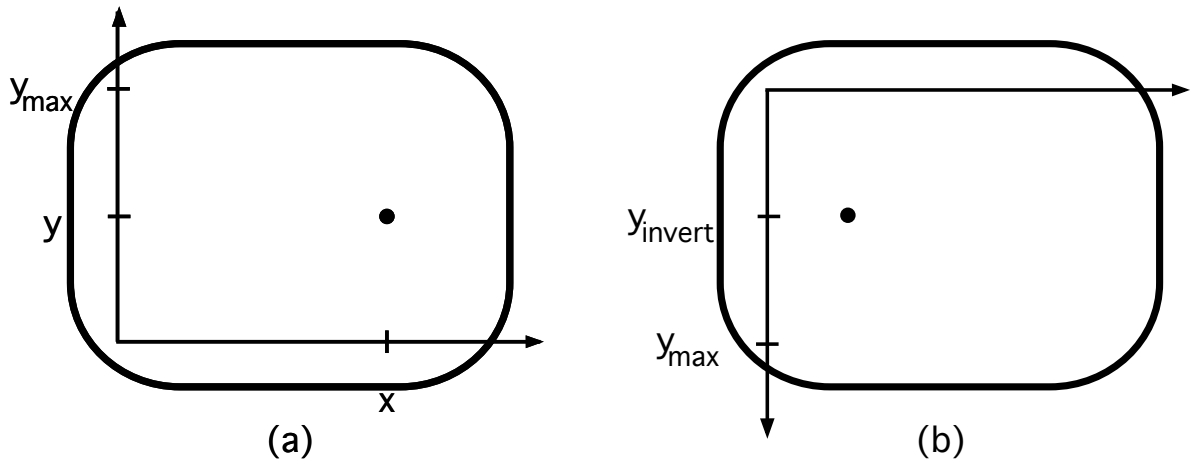


Fig. B.1.Coordenadas de pantalla cartesianas referenciadas con respecto a la esquina infeior izquierda de la pantalla (a) o a la esquina superior izquierda de la pantalla (b)

Por tanto, los comandos dependientes del dispositivo, como por ejemplo los relativos a la entrada interactiva y a las manipulaciones de las ventanas de visualización, suelen hacer referencia a las coordenadas de pantalla utilizando el sistema cartesiano invertido (figura B. 1.(b)). Los valores de las coordenadas horizontales en los dos sistemas son iguales y, un valor *y* invertido se convierte en in valor *y* medido desde la parte inferior de la pantalla mediante el cálculo:

B.1
$$y = y_{\max} - y_{invert}$$

En algunos paquetes de aplicación, el origen de las coordenadas de pantalla puede situarse en una posición arbitraria, como por ejemplo el centro de la pantalla.

Sistemas de referencia cartesianos bidimensionales estándar

Utilizamos sistemas cartesianos en posición estándar para las especificaciones en coordenadas universales, en coordenadas de visualización y para otras referencias dentro de la pipeline de visualización bidimensional. Las coordenadas en estos sistemas de referencias pueden ser positivas o negativas, con cualquier rango de valores. Para mostrar una vista de una imagen bidimensional, designamos una ventana de de recorte y un visor con el fin de mapear la sección de la imagen sobre las coordenadas de pantalla.

Coordenadas polares en el plano *xy*

Un sistema cartesiano bidimensional frecuentemente es el sistema de referencia en coordenadas polares (figura B.2), en el que las coordenadas se especifican mediante una distancia radial *r* con respecto a un eje de coordenadas y un desplazamiento angular θ con respecto a la horizontal. Los desplazamientos angulares positivos se definen en el sentido contrario a las agujas del reloj, mientras que los desplazamientos angulares negativos se definen en el sentido de las agujas del reloj.

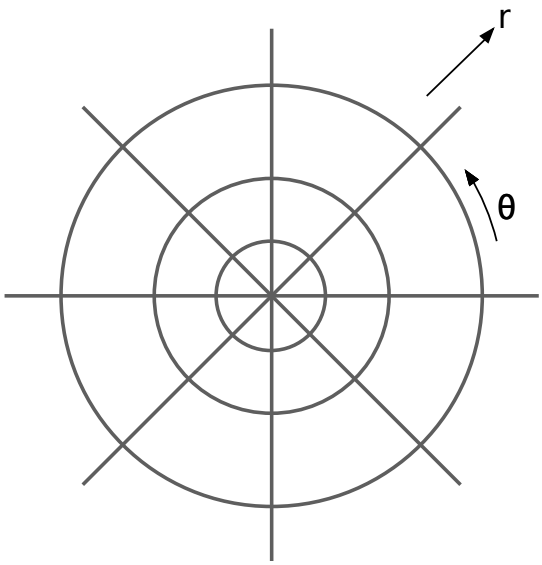


Fig. B.2. Sistema de referencia en coordenadas polares, formado mediante círculos concentricos y líneas radiales.

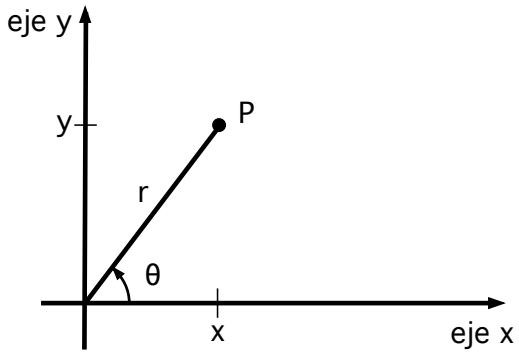


Fig. B.3.Relación entre coordenadas polares y cartesianas

La relación entre las coordenadas cartesianas y polares se muestra en la figura B.3. Considerando el triángulo recto de la figura B.4 y utilizando la definición de las funciones trigonométricas, podemos realizar la transformación de coordenadas polares a coordenadas cartesianas mediante las expresiones:

B.2. $$x = r\cos\theta, \quad y = r\sin\theta$$

La transformación inversa, de coordenadas cartesianas y polares es:

B.3. $$r = \sqrt{x^2 + y^2}$$

Los valores angulares pueden medirse en grados o unidades adimensionales (radianes). Un radian se define como el ángulo subtendido por un arco circular que tenga una longitud igual al radio del círculo. Esta definición se ilustra en la figura B.5, que muestra dos líneas que se intersectan en un plano y un círculo centrado en el punto de intersección P. Para cualquier círculo centrado en P, el valor del ángulo θ en radianes está dado por el cociente:

B.4. $$\theta = \frac{s}{r}(radianes)$$

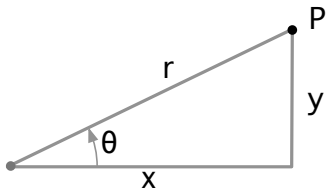


Fig. B.4. Triángulo recto con hipotenusa *r*, lados *x* e *y* y un ángulo interior θ

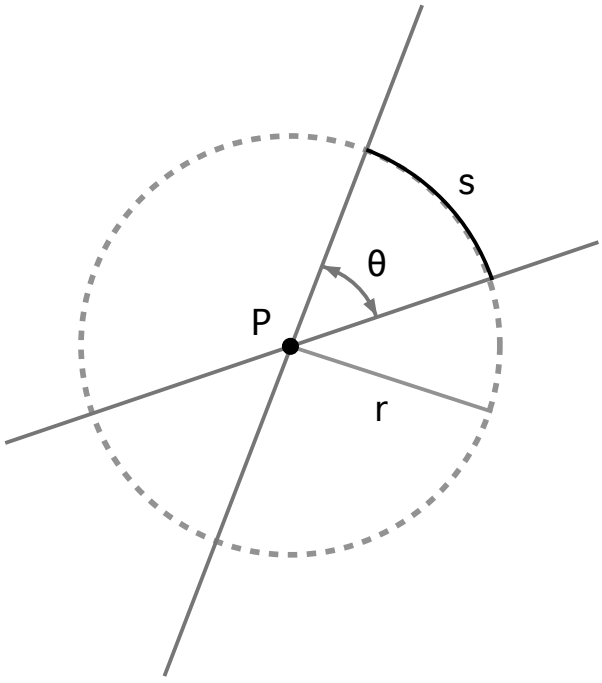


Fig. B.5. Un ángulo θ subtendido por un arco circular de longitud s y radio r.

Donde *s* es la longitud del arco circular que subtienden a θ y *r* es el radio del círculo. La distancia angular total alrededor del punto P es la longitud del perímetro del circulo (2pr) dividida por *r*, lo que es igual a 2p radianes. Si hablamos en grados, una circunferencia se divide en 360 arcos de igual longitud, por lo que cada arco subtiende un ángulo de 1 grado. Por tanto, 360° = 2π radianes.

Pueden utilizarse otras cónicas, además de los círculos, podemos especificar las coordenadas. Por ejemplo, utilizando elipses concéntricas en lugar de círculos, podemos especificar los puntos en coordenadas elípticas, de forma similar, pueden aprovecharse otros tipos de simetrías para definir coordenadas planas hiperbólicas o parabólicas.

Sistema de referencia cartesianos tridimensionales estándar

La figura B.6 muestra la orientación convencional para los ejes de coordenadas en un sistema de referencia cartesiano tridimensional. Decimos que este tipo de sistema cumple la regla de la mano derecha, porque el pulgar de la mano derecha apunta en la dirección *z* positiva si nos imaginamos encerrando el eje z al curvar los dedos desde el eje *x* positivo hacia el eje *y* positivo (abarcando 90°) como se ilustra en la figura B.6.

En la mayoría de los programas infográficos, las descripciones de los objetos y otros tipos de coordenadas se especifican mediante coordenadas cartesianas que cumplen la regla de la mano derecha.

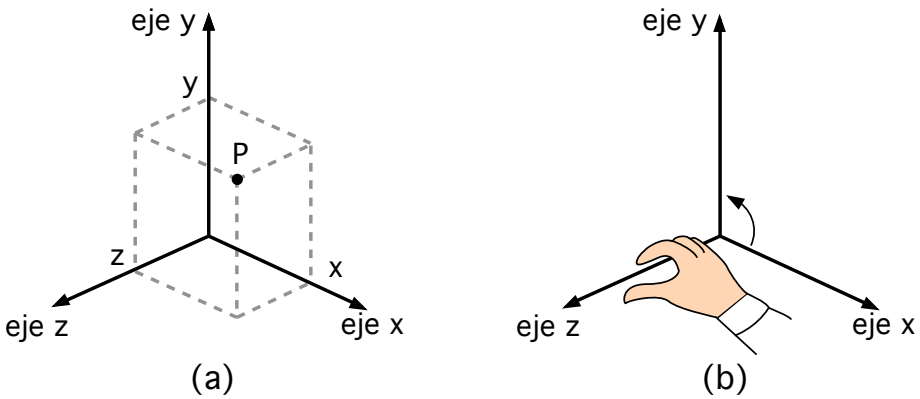


Fig. B.6. Coordenadas de un punto P en la posición (x, y, z) en un sistema de referencia cartesiano estándar que cumple con la

regla dela mano derecha.

Los sistemas de referencia cartesianos son sistemas de coordenadas ortogonales, lo que simplemente significa que los ejes de coordenadas son perpendiculares entre sí. Asimismo, en los sistemas de referencia cartesianos, los ejes son líneas rectas. Aunque también los sistemas de coordenadas con ejes resultan útiles en muchas aplicaciones. La mayoría de dichos sistemas son también ortogonales, en el sentido de que las direcciones de los ejes en cualquier punto del espacio son mutuamente perpendiculares.

Coordenadas de pantalla cartesianas tridimensionales

Cuando se muestra una vista de una escena tridimensional sobre un monitor de vídeo, se almacena información de profundidad para cada posición de pantalla. La posición tridimensional que corresponde a cada punto de pantalla, suele estar referenciada mediante un sistema que cumple la regla de la mano izquierda como se muestra en la figura B.7.

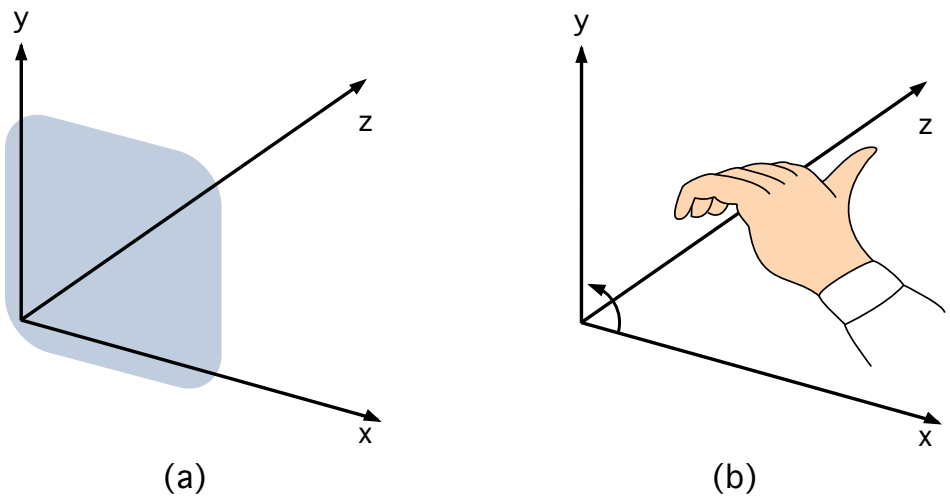


Fig. B.7. Sistema de coordenadas cartesianas que cumple con la regla de la mano izquierda, superpuesto sobre la superficie de un monitor de vídeo.

En este caso, el pulgar de la mano izquierda apunta en la dirección *z* positiva si nos imaginamos rodeando el eje z d modo que los dedos de la mano izquierda vayan desde el eje x positivo hasta el eje y positivo abarcando 90°. Los valores de z positivos indican posiciones situadas detrás de la pantalla para cada punto en el plano *xy,* y el valor de z se incrementará a medida que los objetos se alejen del observador.

B.2. Puntos y vectores

Existe una diferencia fundamental entre el concepto de punto geométrico y el concepto de vector. Un punto es una posición especificada mediante sus coordenadas en algún sistema de referencia. Un vector, por el contrario, tiene propiedades que son independientes del sistema de referencia concreto que elijamos.

Propiedades de los puntos

La figura B.8 ilustra la especificación de un punto bidimensional P mediante sus coordenadas en dos sistemas de referencia distintos. En el sistema A, el punto tiene unas coordenadas que están dadas por el par coordenado (*x, y*) y su distancia con respecto al origen es $\sqrt{x^2 + y^2}$. En el sistema B, el mismo punto tiene coordenadas (0, 0) y la distancia hasta el origen de coordenadas del sistema es 0.

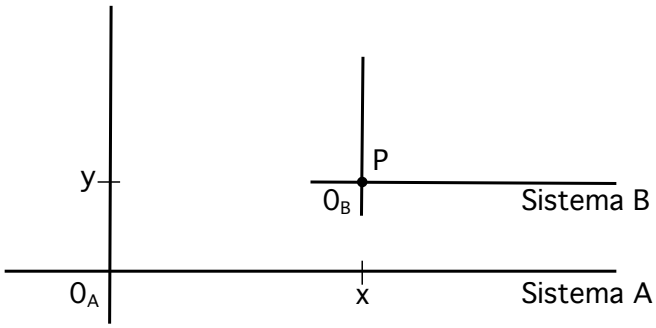


Fig. B.8. Coordenadas de un punto P en dos sistemas de referencia cartesianos distintos

Propiedades de los vectores

En un determinado sistema de coordenadas, podemos definir un vector como la diferencia entre dos puntos. Así, para os puntos bidimensionales P_1 y P_2 en la figura B.9, podemos especificar un vector como:

B.5.
$$\begin{aligned} V &= P_2 - P_1 \\ &= (x_2 - x_1, y_2 - y_1) \\ &= (V_x, V_y) \end{aligned}$$

Donde las componentes cartesianas (o elementos cartesianos) V_x y V_y son las proyecciones de V sobre los ejes *x* e *y*. También podríamos obtener estas mismas componentes del vector utilizando otros dos puntos dentro del sistema de coordenadas. De hecho, existe un número infinito de parejas de puntos que nos dan las mismas componentes de vector, y los vectores se suelen definir mediante un único punto relativo al sistema de referencia actual. Por tanto, un vector no tiene una posición fija dentro de un sistema de coordenadas, asimismo, si transformamos la representación de V a otro sistema de referencia, las coordenadas de las posiciones P_1 y P_2 cambiarán, pero las propiedades básicas del vector no sufrirán modificación.

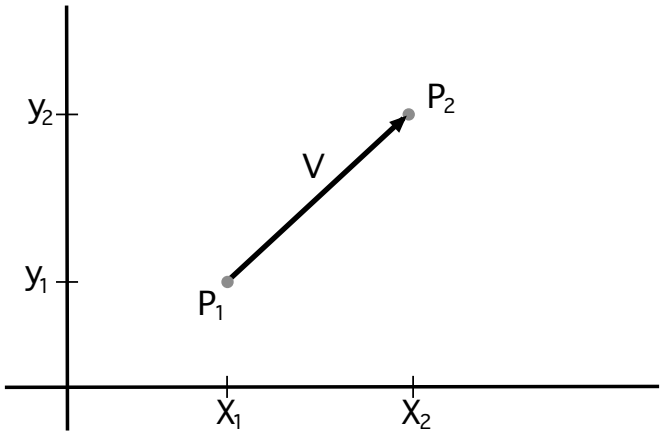


Fig. B.9. Un vector bidimensional V definido en un sistema de referencia cartesiano como la diferencia entre dos puntos.

Podemos describir un vector como un segmento de línea dirigido que tiene dos propiedades fundamentales: modulo y dirección. Para el vector bidimensional de la figura B. 9, calculamos el módulo del vector utilizando el teorema de Pitágoras, que nos da la distancia entre sus dos extremos según la dirección del vector:

B.6.
$$|V| = \sqrt{V_x^2 + V_y^2}$$

Podemos especificar la dirección del vector de diversas formas. Por ejemplo, podemos proporcionar la dirección en términos del desplazamiento angular con respecto a la horizontal de forma siguiente:

B.7.

$$\alpha = \tan^{-1}\left(\frac{V_y}{V_x}\right)$$

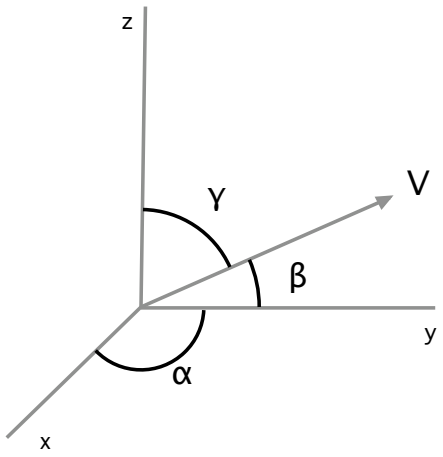


Fig. B.10. Ángulos directores de α, β, γ.

Un vector tiene el mismo módulo y dirección independientemente de dónde situemos el vector dentro de un cierto sistema de coordenadas. Asimismo, el módulo del vector es independiente del sistema de coordenadas que elijamos. Sin embargo, si transformamos el vector a otro sistema de referencia, los calores de esos componentes y si dirección dentro de ese sistema de referencia cartesiano rotado, de modo que la dirección del vector esté definida ahora según la nueva dirección *y*.

Para una representación cartesiana tridimensional de un vector, V = (V_x, V_y, V_z), el módulo del vector será

B.8.

$$|V| = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

Y podemos dar la dirección del vector en términos de los angulares directores, α, β, y γ, que el vector forma con cada uno de los ejes de coordenadas positivos. Podemos calcular estos ángulos de la forma siguiente:

B.9.

$$\cos\alpha = \frac{V_x}{|V|}, \qquad \cos\beta = \frac{V_y}{|V|}, \qquad \cos\gamma = \frac{V_z}{|V|}$$

Los calores de cos α, cos β, y cos γ se denominan cosenos directores del vector. En realidad solo hace falta especificar dos e los cosenos directores para proporcionar la dirección de V, ya que:

B.10. $$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1$$

Los vectores se utilizan para representar cualquier tipo de magnitud que tenga como propiedades un módulo y una dirección. Dos ejemplos comunes son la fuerza y la velocidad (figura B.11.). Una fuerza puede considerarse como la intensidad con que se tira o empuja en una dirección concreta. Un vector de velocidad especifica la rapidez con la que un objeto se mueve en cierta dirección.

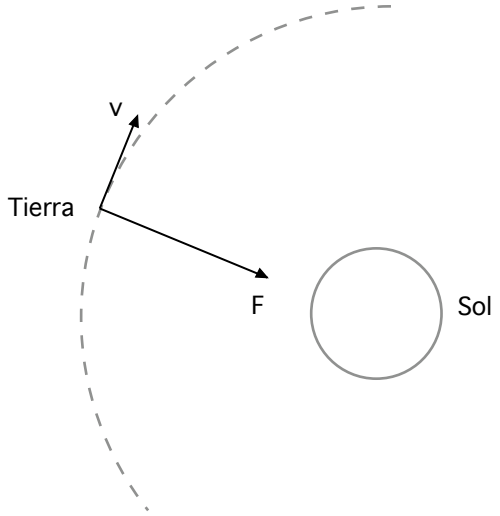


Fig. B.11. Un vector de fuerza gravitatoria F y un vector de velocidad v.

Suma de vectores y multiplicación escalar

Por definición, la suma de vectores se obtiene sumando las componentes correspondientes:

B.11. $$V_1 + V_2 = (V_{1x} + V_{2x}, V_{1y} + V_{2y}, V_{1z}, V_{2z})$$

La figura B.12 ilustra geométricamente la suma bidimensional de vectores. Obtenemos la suma de vectores colocando el extremo inicial sobre el extremo final del otro vector y dibujando la representación del vector suma desde el extremo inicial del primer vector hasta el extremo final del segundo la suma de un vector escalar no está definida, ya que un

escalar sólo tiene un valor numérico, mientras que un vector tiene *n* componentes numéricas en un espacio *n*-dimensional.

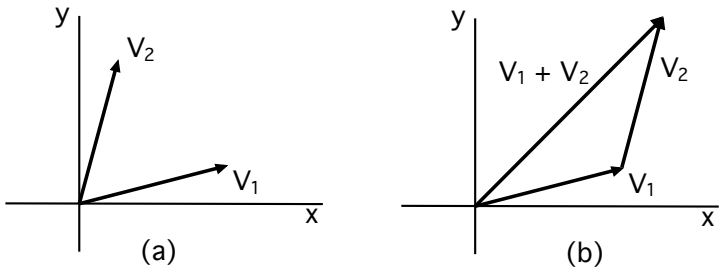


Fig. B.12. Dos vectores (a) pueden sumarse geométricamente situando los dos vectores uno a continuación de otro (b) y dibujando el vectores resultante desde el extremo inicial del primer vector hasta el extremo final del segundo vector.

La multiplicación de un vector escalar *s* se define como

B.12. $$sV = (sV_x, sV_y, sV_z)$$

Por ejemplo, si el parámetro escalar *s* tiene el valor 2, cada componente de V se dobla y el módulo se dobla también.

También podemos combinar vectores utilizando procesos multiplicativos, de diversas formas. Un método muy útil consiste en multiplicar los módulos de los dos vectores, de modo que este producto se utilice para formar otro vector o una magnitud escalar.

Producto escalar de dos vectores

Podemos obtener un valor escalar a partir de dos vectores mediante el cálculo

B.13. $$V_1 \bullet V_2 = |V_1||V_2|\cos\theta, \qquad 0 \le \theta \le \pi$$

Donde θ es el más pequeño de los dos ángulos que pueden definirse entre las direcciones de ambos vectores (figura B.13.). Este esquema de multiplicación se denomina producto escalar de dos vectores. También se denomina producto interno, particularmente al hablar de productos escalares en el análisis tensorial.

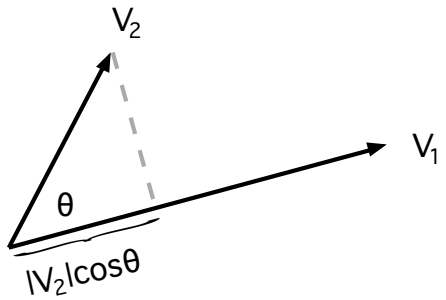


Fig. B.13. El producto escalar de dos vectores se obtiene multiplicando las componentes paralelas.

La ecuación B.13. es válida en cualquier sistema de coordenadas y puede interpretarse como el producto de las componentes paralelas de los dos vectores, donde $|v_2|\cos\theta$ es la proyección del vector V_2 en la dirección de V_1.

Además de expresar el producto escalar en forma independiente del sistema de coordenadas, podemos también expresar este cálculo en un sistema de coordenadas especifico. Para un sistema de referencia cartesiano, el producto escalar se calcula como:

B.14. $$V_1 \bullet V_2 = V_{1x}V_{2x} + V_{1y}V_{2y} + V_{1z}V_{2z}$$

El producto escalar es una generalización del teorema de Pitágoras y el producto escalar de un vector por sí mismo da como resultado el cuadrado del módulo del mismo vector.de igual manera, el producto escalar de dos vectores es cero y si y solo si los dos vectores son perpendiculares (ortogonales).

El producto escalar es conmutativo

B.15. $$V_1 \bullet V_2 = V_2 \bullet V_1$$

porque esta operación produce un valor escalar. Asimismo, el producto escalar es distributivo con respecto a la suma de vectores.

B.16. $$V_1 \bullet (V_2 + V_3) = V_1 \bullet V_2 + V_1 \bullet V_3$$

Producto vectorial de dos vectores

Podemos utilizar la siguiente fórmula con el fin de combinar dos vectores para producir otro vector.

B.17. $$V_1 \times V_2 = u|V_1||V_2|\sin\theta, \qquad 0 \le \theta \le \pi$$

El parámetro u en esta expresión es vector unitario perpendicular tanto a V_1 como a V_2 (figura B.14.). La dirección de u esta determinada por la regla de la mano de la mano derecha: rodeamos un eje perpendicular al plano que contiene a V_1 y a V_2 de modo que los dedos de la mano derecha se curven de V_1 a V_2.

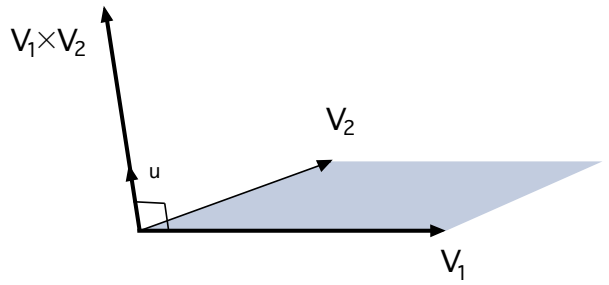


Fig. B.14. El producto vectorial de dos vectores es un vector que apunta en una dirección perpendicular a los dos vectores originales y con un módulo igual al área del paralelogramo sombreado.

El vector u estará entonces en la dirección a la que apunta el dedo pulgar. Este cálculo se denomina producto vectorial de do vectores y la ecuación a.20 es válida en cualquier sistema de coordenadas. El producto vectorial de dos vectores es otro vector perpendicular al plano de esos dos vectores, y el módulo del producto vectorial es igual al área del paralelogramo formado por dos vectores.

También podemos expresar el producto vectorial en términos de las componentes de los vectores dentro de un sistema de referencia específico. En un sistema de coordenadas cartesianas, calculamos las componentes del producto vectorial como:

B.18. $$V_1 \times V_2 = (V_{1y}V_{2z} - V_{1z}V_{2y}, V_{1z}V_{2x} - V_{1x}V_{2z}, V_{1x}V_{2y} - V_{1y}V_{2x})$$

Si designamos los vectores unitarios según los ejes *x*, *y*, *z* como u_x, u_y, u_z, podemos escribir el producto vectorial en términos de las componentes cartesianas utilizando una notación de determinantes:

B.19.
$$V_1 \times V_2 = \begin{vmatrix} u_x & u_y & u_z \\ V_{1x} & V_{1y} & V_{1z} \\ V_{2x} & V_{2y} & V_{2z} \end{vmatrix}$$

El producto vectorial de dos vectores paralelos cualesquiera es cero. Por tanto, el producto vectorial de un vector por sí mismo es cero. De igual manera, el producto vectorial no es conmutativo sino anticonmutativo,

B.20.
$$V_1 \times V_2 = u|V_1||V_2|\sin\theta, \qquad 0 \le \theta \le \pi$$

También se verifica que el producto vectorial no es asociativo, es decir,

B.21.
$$V_1 \times (V_2 \times V_3) \neq (V_1 \times V_2) \times V_3$$

Sin embargo, el producto vectorial es distributivo con respecto a ala suma y resta de vectores:

B.22.
$$V_1 \times (V_2 + V_3) = (V_1 \times V_2) + (V_1 \times V_3)$$

B.3. Matrices

Una matriz es una disposición rectangular de magnitudes (valores numéricos, expresiones o funciones), denominados elementos de la matriz. Algunos ejemplos de matrices son:

B.23.
$$\begin{bmatrix} 3.60 & -0.01 & 2.00 \\ -5.46 & 0.00 & 1.63 \end{bmatrix}, \quad \begin{bmatrix} e^x & x \\ e_{2x} & x^2 \end{bmatrix}, \quad \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}, \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Las matrices se identifican de acuerdo con el número de filas y el número de columnas, para los ejemplos anteriores, las matrices de izquierda a derecha son 2 por 3, 3 por 2, 1 por 3 y 3 por 1. Cuando el número de filas es igual al número de columnas, como en el segundo

ejemplo, la matriz se denomina matriz cuadrada. En general, podemos escribir una matriz *r* por *c* como:

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1c} \\ m_{21} & m_{22} & & m_{2c} \\ \vdots & \vdots & & \vdots \\ m_{r1} & m_{r2} & & m_{rc} \end{bmatrix}$$

B.24.

Donde m_{jk} representa los elementos de la matriz M. El primer subindice de cada elemento proporciona el número de fila y el segundo subíndice proporciona el número de columna.

Una matriz con una única fila o una única columna. En general una matriz puede considerarse como una colección de vectores fila o como una colección de vectores columna.

Cuando expresamos diversas operaciones en forma matricial el convenio matemático estándar consiste en representar un vector mediante una matriz columna. De acuerdo con este convenio, escribiremos la representación matricial de un vector tridimensional en coordenadas cartesianas como:

$$V = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

B.25.

Aunque utilicemos esta representación matricial estándar tanto para los puntos como para los vectores, existe una distinción importante entre ambos conceptos. La representación vectorial de un punto siempre asume que el vector está definido desde el origen hasta dicho punto y la distancia del punto al origen no erá invariante cuando cambiemos de un sistema de coordenada a otro. Asimismo, no podemos “sumar” puntos y tampoco podemos aplicar a los puntos operaciones vectoriales, como el producto escalar y el producto vectorial.

Multiplicación por un escalar y suma de matrices

Para multiplicar una matriz M por un valor escalar *s,* multiplicamos cada elemento m_{jk} por dicho escalar. Como ejemplo, si

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Entonces,

$$3M = \begin{bmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \end{bmatrix}$$

La suma de matrices solo está definida para aquellas matrices que tengan el mismo número de filas *r* y el mismo número de columnas *c*. Para dos matrices cualesquiera *r* por *c*, la suma se obtiene sumando los correspondientes elementos. Por ejemplo,

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 0.0 & 1.5 & 0.2 \\ 6.0 & 1.1 & -10.0 \end{bmatrix} = \begin{bmatrix} 1.0 & 3.5 & 3.2 \\ -2.0 & 6.1 & -4.0 \end{bmatrix}$$

Multiplicación de matrices

El producto de dos matrices se define como una generalización del producto escalar de vectores. Podemos multiplicar una matriz *m* por *n* A por una matriz *p* por *q* B para formar la matriz producto AB, supuesto que el número de columnas de A sea igual al número de filas de B. En otras palabras, debe cumplirse que n = p. Entonces se obtiene la matriz producto formando las sumas de los productos de los elementos de los vectores fila de A por los elementos correspondientes de los vectores columna de B. Así, para que el siguiente producto:

B.26. $$C = AB$$

Obtenemos una matriz *m* por *q* C cuyos elementos se calculan como

B.27.
$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

En el siguiente ejemplo, una matriz 3 por 2 se postmultiplica por una matriz 2 por 2, para generar una matriz producto 3 por 2

$$\begin{bmatrix} 0 & -1 \\ 5 & 7 \\ -2 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 0\cdot 1 + (-1)\cdot 3 & 0\cdot 2 + (-1)4 \\ 5\cdot 1 + 7\cdot 3 & 5\cdot 2 + 7\cdot 4 \\ -2\cdot 1 + 8\cdot 3 & -2\cdot 2 + 8\cdot 4 \end{bmatrix} = \begin{bmatrix} -3 & -4 \\ 26 & 38 \\ 22 & 28 \end{bmatrix}$$

La multiplicación de vectores en notación matricial produce el mismo resultado que el producto escalar, supuesto que el primer vector se exprese como un vector fila y el segundo vector se exprese como un vector columna. Por ejemplo:

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 32 \end{bmatrix}$$

Este producto de vectores da como resultado una matriz con un único elemento (una matriz 1 por 1). Sin embargo, si multiplicamos los vectores en orden inverso, obtendremos la siguiente matriz 3 por 3:

$$\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 8 & 12 \\ 5 & 10 & 15 \\ 6 & 12 & 18 \end{bmatrix}$$

Como ilustran los dos productos de vectores anteriores, la multplicación de matrices no es conmutativa por regla general. Es decir

B.28.
$$AB \neq BA$$

Pero la multiplicación de matrices es distributiva con respecto a la suma de matrices:

B.29.
$$A(B + C) = AB + AC$$

Transpuesta de una matriz

La transpuesta M^T de una matriz se obtiene intercambiando las filas y columnas de la matriz. Por ejemplo,

B.30.
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \begin{bmatrix} a & b & c \end{bmatrix}^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Para un producto de matrices la transpuesta es:

B.31.
$$(M_1 M_2)^T = M_2^T M_1^T$$

Determinante de una matriz

Si tenemos una matriz cuadrada, podemos combinar los elementos de una matriz para generar un único número denominado determinante de la matriz. Las evaluaciones de determinantes resultan muy útiles a la hora de analizar y resolver un amplio rango de problemas. Para una matriz 2 por 2 A, el determinante de segundo orden se define como

B.32.
$$\det A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

Los determinantes de orden superior se obtienen recursivamente a partir de los valores de los determinantes de orden inferior para calcular un determinante de orden 2 o superior, podemos seleccionar cualquier columna *k* de una matriz de *n* por *n* y calcular el determinante como:

B.33.
$$\det A = \sum_{j=1}^{n} (-1)^{j+k} a_{jk} \det A_{jk}$$

Donde $\det A_{jk}$ es el determinante (n-1)por(n-1) de la submatriz que se obtiene a partir de A borrando la fila j-ésima y la columna k-ésima. Alternativamente podemos seleccionar cualquier fila *j* y calcular el determinante como:

B.34. $$\det A = \sum_{k=1}^{n} (-1)^{j+k} a_{jk} \det A_{jk}$$

La evaluación de determinantes para matrices de gran tamaño (por ejemplo, n>4) puede realizarse de manera más eficiente utilizando métodos numéricos. Una forma de calcular un determinante consiste en descomponer una matriz en dos factores: A=LU, donde todos los elementos de la matriz L por encima de la diagonal son cero.

Entonces, podemos calcular el producto de las diagonales tanto para L como para U y obtener detA multiplicando los dos productos diagonales. Este método se basa en la siguiente propiedad de los determinantes:

B.35. $$\det(AB) = (\det A)(\det B)$$

Inversa de una matriz

Con las matrices cuadradas, podemos obtener una matriz inversa si y solo si el determinante de la matriz es distinto de cero. Si tiene una inversa, se dice que la matriz es una matriz no singular. En caso contrario, la matriz se denomina matriz singular. En la mayoría de las aplicaciones prácticas en las que las matrices representan operaciones físicas, lo normal es que exista la inversa.

La inversa de una matriz (cuadrada) n por n M se designa M^{-1}, cumpliéndose que:

B.36. $$MM^{-1} = M^{-1}M = I$$

Donde I es la matriz identidad. Todos los elementos diagonales de I tiene el valor 1 y todos los demás elementos (los no situados en la diagonal) son cero.

Los elementos de la matriz inversa pueden calcularse a partir de los elementos de M mediante la fórmula:

B.37. $$m_{jk}^{-1} = \frac{(-1)^{j+k} \det M_{kj}}{\det M}$$

Donde m^{-1}_{jk} es el elemento es el elemento en la fila j-ésima y la columna k-ésima de M^{-1} y M_{jk} es la submatriz (n-1) por (n-1) que se obtiene al borrar la fila k-ésima y la columna j-ésima de la matriz M. Para valores de *n*, podemos calcular de manera más eficiente los valores de los determinantes y los elementos de la matriz inversa utilizando métodos numéricos.

Representaciones no paramétricas

Cuando escribimos las descripciones de los objetos directamente en términos de las coordenadas correspondientes al sistema de referencias que estamos utilizando, la representación se denomina no paramétrica. Por ejemplo, podemos describir una superficie, con cualquiera de las siguientes funciones cartesianas:

B.38. $$f_1(x,y,z)=0, \quad \text{o} \quad z=f_2(x,y)$$

La primera forma de la ecuación B.38. se denomina fórmula implicita de la superficie, mientras que la segunda forma se denomina representación explícita. En la representación explícita, *x* e *y* se denominan variables independientes y *z* se denomina variable dependiente.

De forma similar, podemos representar una línea curva tridimensional en forma no paramétrica como la intersección de dos funciones de superficies, o bien podemos representar la curva con la pareja de funciones:

B.39. $$y=f(x) \quad \text{y} \quad z=g(x)$$

con la coordenada *x* como variable independiente. Los valores de las variables dependientes *y* y *z* se determinan entonces a partir de las ecuaciones B.39 asignando valores a x para algún número prescrito de intervalos.

Las representaciones no paramétricas resultan útiles para describir los objetos de un cierto sistema de referencia, pero presentan algunas desventajas a la hora de utilizarlas en algoritmos gráficos.

Si queremos obtener una gráfica suave, debemos cambiar la variable independiente cuando la primera derivada (pendiente) de f(x) o g(x) sea superior a 1. Esto requiere controlar de manera continua los valores de las derivadas para determinar cuándo es necesario cambiar los roles de las variables dependientes e independientes. Asimismo, las ecuaciones B.39. proporcionan un formato muy engorroso para representar funciones mutivaluadas, por ejemplo, la ecuación implícita de un círculo centrado en el origen del plano *xy* es:

$$x^2 + y^2 - r^2 = 0$$

Y la ecuación explícita de *y* es la función multivaluada:

$$y = \pm\sqrt{r^2 - x^2}$$

En general, una representación mas conveniente para describir los objetos en los algoritmos gráficos es en términos de ecuaciones paramétricas

Representaciones paramétricas

Podemos clasificar los objetos según el número de parámetros necesarios para describir las coordenadas de los mismos. Una curva, por ejemplo, en un sistema de referencia cartesiano, se clasifica como un objeto euclídeo unidimensional, mientras que una superficie es un objeto euclídeo bidimensional. Cuando se proporciona la descripción de un objeto en términos de su parámetro de dimensionalidad, la descripción se denomina representación paramétrica.

La descripción cartesiana de los puntos situados a lo largo de la trayectoria de una curva puede proporcionarse en forma paramétrica utilizando la siguiente función vectorial

B.40. $$P(u) = (x(u), y(u), z(u))$$

donde cada una de las coordenadas cartesianas es una función del parámetro *u*. En la mayoría de los casos, podemos normalizar las tres funciones de coordenadas de modo que el parámetro y varíe en el rango comprendido entre 0 y 1.0. Por ejemplo, un círculo en el plano *xy* con un radio *r* y con centro en el origen de coordenadas puede definirse en forma paramétrica mediante las siguientes tres funciones:

B.41. $$x(u) = r\cos(2\pi u), \quad y(u) = r\sin(2\pi u), \quad z(u) = 0, \quad 0 \le u \le 1$$

Puesto que esta curva está definida en el plano xy, podemos eliminar la función z(u), que tiene el valor constante de 0.

De forma similar, podemos representar las coordenadas de una superficie utilizando la siguiente función vectorial cartesiana:

B.42. $$P(u,v) = (x(u,v), y(u,v), z(u,v))$$

Cada una de las coordenadas cartesianas es ahora función de los dos parámetros de la superficie *u* y *v*. Una superficie esférica con radio *r* y con centro en el origen de coordenadas, por ejemplo, puede describirse mediante las ecuaciones:

B.43.
$$\begin{aligned} x(u,v) &= r\cos(2\pi u)\sin(\pi v) \\ y(u,v) &= r\sin(2\pi u)\sin(\pi v) \\ z(u,v) &= r\cos(\pi v) \end{aligned}$$

El parámetro *u* define líneas de longitud constante sobre la superficie, mientras que el parámetro *v* describe líneas de latitud constante. Las ecuaciones paramétricas, de nuevo, se suelen normalizar para asignar a *u* y a *v* valores en el rango comprendido entre 0 y 1.0.

Apéndice C
Manual de Prácticas

Práctica 1 Algoritmos aplicados a gráficos

Material

- Previamente el alumno deberá haber leído el tema 1.1 de este texto
- Acceso a internet y biblioteca

Objetivo

El alumno conocerá los algoritmos gráficos y su funcionamiento.

Desarrollo

El profesor dará una explicación breve acerca los algoritmos para gráficos creados a partir de los 1960s, tales como el algoritmo z-buffer. Los alumnos buscarán en internet y/o libros una descripción de cada uno de los algoritmos y escribirán un resumen.

Reporte:

1. Portada
2. Resumen de los algoritmos
3. Conclusiones

Recursos:

http://arantxa.ii.uam.es/~pedro/graficos/teoria/
http://es.wikipedia.org/wiki/Gr%C3%A1ficos_por_computadora
Watt, A., Policarpo, F., The Computer Image, Addison Wesley.
Joan Trias Pairo. Geometría para la informática gráfica y CAD. Ed. Alfaomega

Policarpio, Fabio. The Computer image. Ed. Addison-Wesley

Práctica 2 Dispositivos

Material

- Acceso a internet y biblioteca

Objetivo

El alumno conocerá los principales dispositivos utilizados para la graficación.

Desarrollo

El alumno realizará una investigación de los dispositivos que se benefician y/o utilizan a los gráficos por computadora, tales como las tabletas digitalizadoras.

Reporte

1. Portada
2. Dispositivos de entrada
3. Dispositivos de salida
4. Conclusiones

Resumen

http://es.wikipedia.org/wiki/Entrada/Salida

http://dblinux.sis.epn.edu.ec/~elascano/sistemasmultimediales/mediosytc/output.html

Martín, Javier. Informática Básica. Editorial Ra-Ma.

Hearn, Donald y M. Pauline Baker. Gráficas por Computadora. Prentice Hall.

Foley, J. D. y A. Van Dam. Gráficos por computadora. Principios y práctica. Segunda edición. Addison - Wesley

Práctica 3 Compresión de archivos gráficos

Material

- Acceso a internet y biblioteca

Objetivo

Conocer los métodos actuales utilizados para la compresión de archivos gráficos.

Desarrollo

El alumno realizará un trabajo de investigación sobre las técnicas de compresión de archivos gráficos que incluya los siguientes temas:

- Codificación de longitud de recorrido
- Codificación LZW
- Métodos de compresión mediante reconocimiento de patrones
- Codificación Huffman
- Codificación aritmética

Reporte

1. Portada
2. Definición de codificación
3. Objetivos de la codificación
4. Tipos de codificación
5. Codificación de longitud de recorrido
6. Codificación LZW
7. Métodos de compresión mediante reconocimiento de patrones
8. Codificación Huffman
9. Codificación aritmética
10. Conclusiones

Recursos

http://www.nationalarchives.gov.uk/documents/es5_image_compression.pdf

http://es.wikipedia.org/wiki/Compresi%C3%B3n_de_datos

Gonzalez, R.C. Tratamiento digital de imágenes. Ediciones Díaz de Santos.

Brown, C. W., Shepherd B. J. Graphics File Formats, reference and guide, Prentice Hall.

Murray, J. D., van Ryper, W. Encyclopedia of Graphics File Formats, O'Reilly Associates.

Práctica 4 OpenGL

Material

- Acceso a internet y biblioteca

Objetivo

A través de esta práctica el alumno conocerá qué es OpenGL y cuales son sus aplicaciones.

Desarrollo

El alumno realizará un trabajo de investigación de carácter teórico.

Reporte

1. Portada
2. Breve historia de OpenGL
3. Diseño
4. Extensiones
5. Bibliotecas
6. Conclusiones

Recursos

http://es.wikipedia.org/wiki/OpenGL

http://www.opengl.org/

Wirght, Richard. Programación en OpenGL. Anaya Multimedia

Angel, E. Interactive Computer Graphics: A top-down approach with OpenGL, AddisonWesley.

Ribelles ,J. y Lluch J. Opengl en Fichas: Una introducción practica. Treballs d' informatica i Tec

Práctica 5 Instalación de OpenGL

Material

- Acceso a internet

Objetivo

El alumno aprenderá a instalar las librerías de OpenGL

Desarrollo

El profesor explicará la forma de instalar OpenGL en distintas plataformas, posteriormente el alumno instalará de forma personal el software indicado.

Reporte

1. Portada
2. Pasos de instalación para OpenGL
3. Problemas en la instalación
4. Conclusiones

Recursos

Http://www.opengl.org

http://www.opengl.org/resources/libraries/glut/glut_downloads.php

http://www.xmission.com/~nate/glut.html

Anexo 1 de este texto

Wirght, Richard. Programación en OpenGL. Anaya Multimedia

Práctica 6 Traslación Bidimensional

Material

- Tener instalado OpenGL

Objetivo

Desarrollar un programa en OpenGL que realice una traslación bidimensional

Desarrollo

El profesor explicará en qué consiste la traslación bidimensional. El alumno creará un programa que realice la operación mencionada.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del método o función de traslación
4. Conclusiones

Recursos:

http://itmgraficacion.googlepages.com/

http://www.ieev.uma.es/tdi/www_netscape/TEMAS/Tdi_22/index4.php

Demel, John T. y Michael J. Miller. Gráficas por computadora. Ed. Mc. Graw Hill.

Govil-Pai, S., Pai, R.Learning Computer Graphics, Springer Verlag.

Wright, Richard. Programación en OpenGL. Anaya Multimedia

Práctica 7 Rotación Bidimensional

Material

- Tener instalado OpenGL

Objetivo

Desarrollar un programa en OpenGL que realice una rotación bidimensional

Desarrollo

El profesor explicará el procedimiento para realizar una rotación bidimensional. El alumno programará una rutina en OpenGL que rote un polígono sobre un punto de pívote.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del método o función de rotación
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://www.ieev.uma.es/tdi/www_netscape/TEMAS/Tdi_22/index5.php

Demel, John T. y Michael J. Miller. Gráficas por computadora. Ed. Mc. Graw Hill.

Plastock, R. A. y Kaslley. Gráficas por computadora. McGraw-Hill.

Wright, Richard. Programación en OpenGL. Anaya Multimedia

Práctica 8 Escalación Bidimensional

Material

- Tener instalado OpenGL

Objetivo

Desarrollar un programa en OpenGL que realice una escalación bidimensional

Desarrollo

El profesor explicará el procedimiento para realizar una escalación bidimensional, posteriormente el alumno programará una rutina que realice cálculos para cambios de escala en un polígono. El programa deberá mostrar el polígono escalado.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del método o función de escalación
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://www.ieev.uma.es/tdi/www_netscape/TEMAS/Tdi_22/index6.php

Donald Hearn / M. Pauline Baker. Graficas por computadora. Ed. Prentice Hall Hispanoamericana.

Woo, M., Neider, J., Davis, T., OpenGL Programming Guide, 2nd. Ed., Addison Wesley

Wright, Richard. Programación en OpenGL. Anaya Multimedia

Práctica 9 Matrices Bidimensionales Compuestas

Material

- Tener instalado OpenGL

Objetivo

Implementar una rutina en OpenGL que realice tres operaciones a través de matrices bidimensionales compuestas.

Desarrollo

El profesor explicará el procedimiento para realizar matrices bidimensionales compuestas así como su utilidad. El alumno desarrollara un programa que realice una escalación, rotación y traslación de un triángulo.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente de los métodos o funciones en los cuales se crea y modifica la matriz bidimensional
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://gsii.usal.es/~corchado/igrafica/descargas/temas/Tema06.pdf

Mortenson , Michael E. Mathematics for Computer Graphics Applications. Ed. Industrial Press Inc.

Donald Hearn / M. Pauline Baker. Graficas por computadora. Ed. Prentice Hall Hispanoamericana.

Wright, Richard. Programación en OpenGL. Anaya Multimedia.

Práctica 10 Traslación tridimensional

Material

- Tener instalado OpenGL

Objetivo

Conocer el procedimiento para realizar la traslación tridimensional de un objeto.

Desarrollo

El profesor mostrará el procedimiento matemático para realizar una traslación trisimensional. El alumno creará una rutina que construirá una matriz de traslación tridimensional.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del método en el cual se crea la matriz de rotación
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://gsii.usal.es/~corchado/igrafica/descargas/temas/Tema06.pdf

Foley, James D. y Andries Van Dam. Introducción a la graficación por computadora. Ed. Addison Wesley Iberoamericana.

Buss, R. Samuel. Computer Graphics: An introduction with OpenGL. Cambridge University Press.

Wright, Richard. Programación en OpenGL. Anaya Multimedia

Práctica 11 Escalación tridimensional

Material

- Tener instalado OpenGL

Objetivo

El alumno desarrollará un programa que realizará la construcción de una matriz de traslación tridimensional.

Desarrollo

Previamente a la práctica el profesor explicará el procedimiento para llevar a cabo la escalación tridimensional. El alumno desarrollará un programa que cree una matriz de traslación usando la incorporación directa de las coordenadas del punto fijo

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del método en el cual se crea la matriz de escalación
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://gsii.usal.es/~corchado/igrafica/descargas/temas/Tema06.pdf

McConell, Jeffrey. Computer Graphics, theory into practice. Jones and Bartlett Publishers.

SIGGRAPH '97. Computer graphics: proceedings. Addison-Wesley

Wright, Richard. Programación en OpenGL. Anaya Multimedia

Práctica 12 Matrices Tridimensionales Compuestas

Material

- Tener instalado OpenGL

Objetivo

Implementar una rutina en OpenGL que realice tres operaciones a través de matrices tridimensionales compuestas.

Desarrollo

El profesor explicará el procedimiento para realizar matrices tridimensionales compuestas así como su utilidad. El alumno desarrollara un programa que realice una escalación, rotación y traslación de una matriz.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente de los métodos o funciones en los cuales se crea y modifica la matriz tridimensional
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

Demel, John T. y Michael J. Miller. Gráficas por computadora. Ed. Mc. Graw Hill.

Lewell, John. Computer Graphics: A Survey of Current Techniques and Applications. Van Nostrand Reinhold.

Hearn, Donald. Gráficos por computadora con OpenGl. Pearson.

Práctica 13 Funciones para poliedros regulares de GLUT

Material

- Tener instalado OpenGL

Objetivo

Conocer e implementar las funciones de creación de poliedros regulares de GLUT.

Desarrollo

El profesor explicará los métodos o funciones de la blbilioteca GLUT para la creación de poliedros regulares. El alumno posteriormente desarrollará un programa que muestre los cinco tipos diferentes de poliedros en una visualización de malla de alambre.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del método o función en que se crean y muestran en pantalla los poliedros
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

Foley, J. D. y A. Van Dam. Gráficos por computadora. Principios y práctica. Segunda edición. Addison - Wesley

Hill, Francis. Computer Graphics: Using OpenGL. Prentice Hall.

Hearn, Donald. Gráficos por computadora con OpenGl. Pearson.

Práctica 14 Superficies Cuádricas con GLUT y GLU

Material

- Tener instalado OpenGL

Objetivo

Implementar la visualización de objetos a través del uso de superficies cuádricas.

Desarrollo

El profesor explicará los conceptos relacionados con las superficies cuádricas y la utilización de los métodos que lo implementan en las librerías GLU Y GLUT. El alumno utilizará las bibliotecas mencionadas para crear un programa en el que se visualizarán tres objetos con superficies cuádricas (esfera, cono y cilindro).

Reporte

1. Portada
2. Descripción del programa creado

3. Código fuente del método o función en que se crean y muestran en pantalla los objetos de superficies cuádricas
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://www.cvc.uab.es/shared/teach/a21306/doc/Apuntes%20de%20OpenGL.pdf

Watt, A., Watt, M., Advanced Animation and Rendering Techniques: Theory and Practice, Addison-Wesley.

Watt, A., 3D Computer Graphics, Addison-Wesley,

Steed A., Slater y Chrysanthou. Computer graphics and virtual environments. Addison-Wesley

Práctica 15 Curvas de Bézier

Material

- Tener instalado OpenGL

Objetivo

El alumno conocerá e implementará el cálculo de las funciones de combinación de Bézier y la generación de curvas bidimensionales con splines de Bézier

Desarrollo

El profesor explicará el método para realizar el cálculo de las funciones de combinación de Bézier así como el procedimiento para la generación de curvas bidimensionales con splines de Bézier. El alumno desarrollará un programa que cree una curva de Bézier con cuatro puntos de control en el plano *xy.*

Reporte

1. Portada
2. Descripción del programa creado

3. Código fuente del programa
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://lsi.ugr.es/~jctorres/cad/teoria/sesion7.html

Bartels, R., Beatty, J., Barsky, B. An Introduction to Splines for Use in Computer Graphics and Geometric Modelling. Springer Verlag,

Harrington, Steve. Computer Graphics: A Programming Approach. Ed. Mc Graw Hill.

Hearn, Donald. Gráficos por computadora con OpenGl. Pearson.

Práctica 16 Superficies de Bézier

Material

- Tener instalado OpenGL

Objetivo

El alumno conocerá e implementará el método de generación de Superficies de Bézier.

Desarrollo

El profesor explicará el método para generar superficies de Bézier. El alumno creará un programa que genere una superficie de Bézier.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del programa
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://www.cs.buap.mx/~asanchez/Graphics.html

Farin, G., Curves and Surfaces for Computer Aided Geometric Design, Academic Press.

Glassner, A., Graphics GEMS V, Academic Press

Parejo, José. Juan Manuel Cordero Valle. Curvas y Superficies para modelado geométricoEditorial Ra-ma.

Práctica 17 Superficies B-spline

Material

- Tener instalado OpenGL

Objetivo

El alumno conocerá e implementará el método de generación de Superficies B-spline.

Desarrollo

El profesor explicará el método para generar superficies de B-Spline El alumno creará un programa que genere una superficie de B-Spline.

Reporte

1. Portada
2. Descripción del programa creado
3. Código fuente del programa
4. Conclusiones

Recursos

http://itmgraficacion.googlepages.com/

http://lsi.ugr.es/~jctorres/cad/teoria/sesion9.html

Piegl, L., Tiller, W., The NURBS Book, Springer Verlag

Parslow, R. D. Computer Graphics: Techniques and Applications. Brunel University Computer Science.

Hearn, Donald. Gráficos por computadora con OpenGl. Pearson.