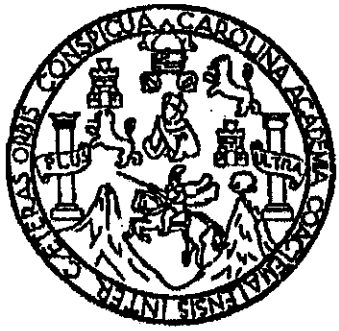
# UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



## FACULTAD DE INGENIERIA

# ANIMACION POR COMPUTADORA EN 3-D, ALGORITMOS Y ESTADO ACTUAL

TESIS

Presentada a la Junta Directiva de la
Facultad de Ingeniería

POR

HERMAN IGOR VELIZ LINARES

Al conferirle el título de

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MARZO DE 1,997

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de tesis titulado:

# ANIMACION POR COMPUTADORA EN 3-D, ALGORITMOS Y ESTADO ACTUAL

tema que me fuera asignado por la coordinación de la carrera de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería.

Herman Igor Véliz Linares

Guatemala, marzo de 1,997


## MIEMBROS DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO | Ing. Herbert René Miranda Barrios. |
| Vocal 1o. | Ing. Jack Douglas Ibarra Solórzano. |
| Vocal 2o. | Ing. Miguel Angel Sánchez Guerra. |
| Vocal 3o. | Ing. Juan Echeverría Méndez. |
| Vocal 4o. | Br. Victor Rafael Lobos Aldana. |
| Vocal 5o. | Br. Wagner Gustavo López Cáceres. |
| Secretario | Licda. Gilda Marina Castellanos de Illescas. |

## TIRBUNAL QUE PRACTICO EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. Julio Ismael González Podszueck. |
| EXAMINADOR | Ing. José Christian Bradna. |
| EXAMINADOR | Ing. Luis Barrundia. |
| EXAMINADOR | Ing. Carlos Ruíz Blau. |
| SECRETARIO | Ing. Francisco Javier González López |

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Escuelas de Ingeniería Civil, Ingeniería Mecánica Industrial, Ingeniería Química, Ingeniería Mecánica Eléctrica, Técnica y Regional de Post-grado de Ingeniería Sanitaria.

Ciudad Universitaria, zona 12
Guatemala, Centroamérica

Guatemala, 22 de octubre de 1,996

Ing.
Jorge Luis Alvarez
Asesor de Públicos y Privados
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Alvarez:

Me permito informarle que he procedido a revisar el trabajo de tesis titulado "Animación por Computadora en 3-D, algoritmos y estado actual", elaborada por el estudiante Herman Igor Véliz Linares, la cual considero que cumple con los objetivos propuestos para su desarrollo.

Sin otro particular, me suscribo de usted,

Atentamente,

Ing. Calixto Calvillo Pérez
Asesor de Tesis
Col. 3656

UNIVERSIDAD DE SAN CARLOS
CIENCIAS DE LA COMPUTACION Y SISTEMAS
FACULTAD DE INGENIERIA

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Escuelas de Ingeniería Civil, Ingeniería Mecánica Industrial, Ingeniería Química, Ingeniería Mecánica Eléctrica, Técnica y Regional de Post-grado de Ingeniería Sanitaria.

Ciudad Universitaria, zona 12
Guatemala, Centroamérica

El coordinador de la Carrera de Ingeniería en Ciencias y Sistemas, después de conocer el dictamen del asesor del trabajo de tesis del estudiante Herman Igor Véliz Linares, titulado "**Animación por Computadora en 3-D, algoritmos y estado actual**", procede a la autorización del mismo.

Ing. Calixto Raúl Menzón Pérez
Coordinador de la Carrera
Ingeniería en Ciencias y Sistemas

UNIVERSIDAD DE SAN CARLOS
CIENCIAS DE LA COMPUTACION Y SISTEMAS
FACULTAD DE INGENIERIA

Guatemala, 30 de octubre de 1996.

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Escuelas de Ingeniería Civil, Ingeniería Mecánica Industrial, Ingeniería Química, Ingeniería Mecánica Eléctrica, Técnica y Regional de Post-grado de Ingeniería Sanitaria.

Ciudad Universitaria, zona 12
Guatemala, Centroamérica

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la autorización por parte del Coordinador de la Carrera de Ingeniería en Ciencias y Sistemas, al trabajo de tesis titulado **Animación por Computadora en 3-D, Algoritmos y estado actual**, presentado por el estudiante Herman Igor Véliz Linares, procede a la autorización para la impresión de la misma.

IMPRIMASE

Ing. Herbert Miranda Barrios
DECANO

Guatemala, marzo de 1,997

UNIVERSIDAD DE SAN CARLOS
DECANO
FACULTAD DE INGENIERIA
DE GUATEMALA

# DEDICATORIA

## A Dios:

Esencia misma del amor, la verdad y el poder.
'Guías mi camino'
'Iluminas mi vida'
'Fortaleces mi alma y espíritu'
'Donde Dios nos plantó es necesario florecer'

## A mi Padre:

Por ser guía y apoyo en los primeros veinte años de mi vida.
Gracias por tú sabiduría y enseñanza.
'Siempre en mi corazón'
'Siempre en mi pensamiento'
Dios te tenga en sus brazos.

## A mi Madre:

Ejemplo del amor más grande en este mundo.
Gracias por tus palabras de aliento, dedicación, sacrificio y apoyo.
'Amar es la vida misma y el secreto de todo lo hermoso en este mundo'

## A mis Hermanos:

Heidy, Boris y José David.
Por ser parte integral e importante en mi familia
'Mantengamos muy en alto el optimismo,
no importando las situaciones de la vida,
que por más difícil que parezcan,
siempre habrá una salida'

## A mis amigos y compañeros:

Con quienes he compartido estos años como en familia.
Gracias a Dios, porque ustedes estuvieron en cada avance en mi camino. Gracias por su ayuda y apoyo, especialmente: Sara Herrera de González, Floriza Avila, Rolando Gándara, Bayron López, Everest Medinilla y Antonio López.
'Es bueno consolar a los amigos,
pero es mejor animarlos'

## A alguien que siempre está en nuestra vida:

'Mientras el sentimiento más grande y fuerte esté con nosotros y sea único, no existirán barreras que puedan acabar con él'
Gracias por todo.

# AGRADECIMIENTO

Al Ing. Calixto Monzón
Por su colaboración y asesoría.

A Heidy Véliz
Por tú apoyo y ayuda.

Al personal del Centro de Cálculo de la Facultad de Ingeniería y a todo el personal de la Facultad de Ingeniería, por su colaboración desinteresada. Muchas Gracias.

A todas las personas que me prestaron su ayuda, de forma desinteresada, Gracias.

# TABLA DE CONTENIDO

# INTRODUCCION

Las gráficas por computadora es un tema interesante que proporciona un mecanismo para construir y observar objetos en tres dimensiones sin el uso de materiales físicos. Los objetos pueden ser teteras, logotipos, carros, edificios, cuerpos humanos, moléculas, puentes, nubes, montañas o el clima; de hecho no existen límites para poder simular cualquier objeto por computadora. La Animación por Computadora introduce la dimensión del tiempo dentro del mundo virtual y permite manipular estos objetos y crear la ilusión de que tienen un movimiento animado.

Los logotipos en los programas de televisión se puede hacer que vuelen de un lugar a otro, realizar acrobacias y volar hacia el infinito. El esqueleto humano puede ser hecho para caminar y explorar dentro de un conjunto de bolas y descomponerlas en distintos objetos y formas. Incluso, las lámparas de mesa pueden ser animadas con vida con tal realismo, que nadie se podrá imaginar cómo se logra tal realismo.

Sin embargo, la Animación por Computadora ha encontrado un excelente campo en la publicidad, efectos especiales de películas y televisión, además es usada para visualizar de una mejor forma una amplia variedad de bases de datos de 3-D y conjuntos de datos numéricos. Además, la técnica CAD (computer aided design) es usada para la práctica en la arquitectura, el cual utiliza la Animación por Computadora para visualizar algún proyecto de arquitectura. Esto se puede hacer con una secuencia animada que muestre una toma aérea o interior del proyecto diseñado. Algunos sistemas existentes que utilizan técnicas de realidad virtual (VR) que permiten visiones estereoscópicas de un ambiente desplegado en tiempo real. Una tecnología similar a la VR es utilizada para evaluar el diseño del uso de luz dentro de edificios.

Las aplicaciones industriales de la Animación por Computadora incluyen productos de visualización de carros, ingeniería de componentes y construcción de proyectos tales como puentes, túneles y esquemas de hidro-eléctricas. En simulación de vuelo, la imagen generada en tiempo real crea escenas sofisticadas en 3-D de aeropuertos internacionales, y dentro de la seguridad de este ambiente virtual los pilotos pueden ser entrenados para desarrollar sus habilidades y procedimientos prácticos de emergencia.

La Animación por Computadora ha sido usada para el desarrollo de gráficas educativas para describir problemas matemáticos, físicos, de astronomía, ciencias, biología, química en dos y tres dimensiones. Las técnicas de visualización son muy útiles para trasladar conjunto de datos multi-dimensionales a imagenes animadas.

El desarrollo de los proyectos de animación no es lo mismo, aunque involucran la modelación, representación de movimientos. En proyectos comerciales, sería imposible proceder con cualquiera de estas fases sin antes tener un bosquejo detallado, el cual describe una historia detallada en la forma de una secuencia de imagenes, que es utilizado para definir los tiempos y control de movimientos, fondos, esquemas de color, luz, requerimientos de post-producción y costos.

Aunque el bosquejo conste de una o dos docenas de esquemas o imagenes, este proporciona una primera idea de cómo será la animación final. Cuando se empieza algún proyecto, lo primero que se hace es identificar cualquier objeto físico que pueda ser necesario para ser la base del objeto virtual en animación. Esto puede incluir algunos objetos tales como muebles, botellas de leche, teteras, y productos de consumo tales como detergentes, relojes, bebidas, etc. La historia también puede incluir puentes, aviones, planetas, naves espaciales, animales y

ciudades enteras, el trasladar dichos objetos a datos geométricos es a veces complicado, ya que involucran medidas y algunas dimensiones importantes que tienen que ser tomadas en cuenta.

Los sistemas comerciales de Animación por Computadora proporcionan un ambiente interactivo en donde el usuario puede construir modelos en 3-D usando diferentes tipos de software. Una interface de manejo de menú proporciona al usuario diferentes características que incluyen: una librería de gráficas básicas tales como cajas, esferas, cilindros, pirámides y conos; algunas facilidades para esculpir, unir, mezclar, extraer; y algunas otras herramientas para formar superficies blandas.

La siguiente etapa depende de la complejidad del proyecto, que puede ser el despliegue de un logotipo, o el despliegue de muchas botellas en movimiento, por lo que el trabajo del animador es resolver los problemas que surgen cuando se desarrollan proyectos como los anteriores. Pero los sistemas de animación proporcionan una variedad de herramientas para la animación de objetos, los cuales son muy útiles para resolver cualquier problema que se presente, pero cuando son problemas demasiado complejos, se pueden desarrollar programas especiales para resolver dichos problemas.

Otro ejemplo muy común puede ser el movimiento de una bola balanceándose, pero en el mundo virtual las gráficas por computadora dicho movimiento debe ser especificado explícitamente. Esto puede ser en la forma de ecuación en donde la dinámica de la bola está expresada usando los conocimientos de balística, o se puede usar una técnica gráfica, donde una curva controla la posición de la bola en cualquier punto en el tiempo. Con otra aproximación, el animador deberá desarrollar la curva hasta crear el efecto visual deseado.

Sin embargo, la animación no siempre se refiere o se relaciona con el movimiento de objetos; involucra el movimiento de una cámara imaginaria, la forma en que se mueven las luces, la forma de los rangos de colores. El controlar este amplio rango de atributos da como resultado una infinidad de técnicas que son referencidas a lo largo de este trabajo de tesis. Consecuentemente, los algoritmos para generar curvas son muy importantes en la Animación por Computadora, y además, se pueden desarrollar técnicas especiales para desarrollar pequeños segmentos de curvas que pueden ser unidos entre sí para formar una curva mayor.

En alguna etapa, el animador deberá establecer los números para los colores y niveles de realismo, y el sistema de animación por computadora proporciona las herramientas para asignar a los objetos con un amplio rango de tributos físicos en la forma de color y textura. La superficie de un objeto puede ser ajustada para cualquier color, y hacer que parezca de colores mate, cromados, brillosos. La técnica para mapeo de textura permite que una imagen real sea mapeada sobre cualquier superficie. Existen procesos similares que permiten simular los efectos de transparencia, reflexiones y efectos de abombamiento.

El animador debe tener la capacidad de crear los efectos de iluminación del mundo imaginario, decidir en donde se colocará la luz fuente, la intensidad y color. Después realizar pruebas para garantizar que la secuencia total, se tiene un nivel aceptable de continuidad.

La secuencia de animación por computadora puede estar compuesta con acción o involucrar el uso de control de movimiento de cámaras; en esta parte, es donde se usa lo que son las técnicas de post-producción. Las imagenes son transferidas a un sistema de edición digital donde pueden ser mezcladas cualquier número de imagenes sin disminuir su calidad.

Aunque la imagenes por computadora parecen irreales a su manera, a como se está acostumbrado a ver, no existe nada fuera de lo normal en la creación de las mismas; éstas son producidas con una variedad de técnicas que han sido desarrolladas en el transcurso de los últimos años. Existe una complejidad en la creación de árboles en crecimiento hasta el desarrollo de los movimientos faciales de un rostro humano, sus gestos y animación.

La mayoría de las técnicas que desarrollan proyectos como los anteriores están basados en simples conceptos, que no se requiere de conocimientos complejos de computación, pero si de un mínimo nivel de ellos. Una forma lógica de resolver los problemas es el uso de la matemática, sobre todo, trabajar en el diseño, prueba de los programas de computadora.

Es muy útil el conocimiento sobre matemática, ya que se deberá trabajar la mayor parte del tiempo con técnicas matemáticas para crear una simulación dinámica. Algunas veces no es posible obtener un nivel de entendimiento matemático, tal y como se requiere para desarrollar algunos objetos complejos. La mayoría de las técnicas expuestas en el presente trabajo de tesis solamente se requiere de algunas herramientas matemáticas que se utilizan para resolver diferentes problemas propuestos durante el transcurso del presente trabajo.

La animación por computadora, como se verá, es una disciplina bastante desarrollada en diferentes lenguajes, y es usada por animadores para comunicarse con ella. Esto es una mezcla de los términos de animación tradicional con algunas técnicas presentadas en capítulos posteriores tales como la técnicas de sombreado de Phong, superficies de Bézier, NURBS y fractales, con términos matemáticos como rotación de matrices, vectores y quaternion.

Mientras se avance en la lectura del presente trabajo de tesis, hay que recordar que el tema de la Animación por Computadora es bastante amplio y que está en su proceso de evolución; en un principio se utilizaron técnicas sencillas ya que eran menos costosas. Actualmente se hace uso de polígonos, cortes, partículas y fractales para modelar el mundo virtual deseado. Se usa la técnica de sombreado de Gouraud, trayectorias de reflejos y radiosidad para representar escenas en color, y animar objetos y cámaras utilizando curvas de Bézier y matrices. En el futuro, se utilizarán nuevas técnicas y se desecharán las actuales. El hecho de que la Animación por Computadora involucra el uso de la computadora hace que puedan existir cambios, nuevas ideas esperan por ser desarrolladas, y sólo serán descubiertas por aquellas personas que manejan el concepto de mantener una mente abierta a lo nuevo.

# CAPITULO 1

# GRÁFICAS POR COMPUTADORAS

## INTRODUCCIÓN

Es cualquier secuencia de tiempo de cambios visuales en una escena. Además de cambiar las posiciones de los objetos con traslaciones y rotaciones, una animación por computadora podría desplegar variaciones de tiempo, en el tamaño, el color, la transparencia o la textura de la superficie de los objetos.

Por ejemplo, las animaciones publicitarias realizan la transición de la forma de un objeto a otra; por ejemplo, la transformación de una lata de aceite para motor en un motor de automóvil.

Además, la Animación por Computadora se refiere a la aplicación de sistemas de computación en la producción de imágenes animadas. La tecnología de la computación y el arte de la animación están definidas como tópicos independientes y son fáciles de entender.

Para entender esta relación de técnicas, debe conocerse cómo el sistema numérico de las computadoras es usado para procesar imágenes -este tema es conocido en el área como 'gráficas por computadora'. En un inicio, no se está familiarizado con los principios de las gráficas por computadora; quizá se esté preparado para una introducción sobre el tema y estudiar algunos conceptos que transforman una computadora en una herramienta para el diseño de gráficas. Este capítulo será de introducción, el cual servirá para familiarizarse con el tema de la Animación por Computadora.

## 1.1 ORIGEN DE LAS GRÁFICAS POR COMPUTADORA

El uso de gráficas por computadora no es un tema nuevo; se usan para resolver y definir problemas dibujando gráficas utilizando algunas técnicas ya establecidas de la geometría, álgebra, óptica y fisiología humana. La geometría es utilizada para describir un espacio de dos y tres dimensiones, mientras las técnicas algebraicas son usadas para definir y evaluar ecuaciones asociadas con el espacio. La ciencia de la óptica proporciona modelos para describir el comportamiento de la luz, y la fisiología humana ofrece modelos para una visión humana y una percepción del color.

Aunque este rango de tópicos sugiere un cierto aire de complejidad, las gráficas por computadora es, de hecho, relativamente fácil de entender -para los tópicos individuales usados para simular un mundo virtual de forma, tamaño, color y movimiento pueden ser usados algunos principios técnicos básicos. Así como se puede esperar, cuando las imágenes por computadora son usadas para explorar temas tales como modelos moleculares o diagnósticos médicos, estas gráficas básicas pueden ser realizadas por medio de otras técnicas. Existen aspectos de la Animación por Computadoras que requieren de algunos conocimientos de matemática, que ayuda a las personas a resolver un problema más complejo.

# 1.1.1 GRÁFICAS POR COMPUTADORA DE 2-DIMENSIONES

El centro de las gráficas por computadora es el concepto de las coordenadas cartesianas. En el caso de las 2-Dimensiones (2-D), éste permite un punto sobre una superficie plana que será direccionado con la ayuda de un par de ejes horizontal y vertical. Un punto es localizado midiendo dos distancias paralelas a los ejes desde su punto de intersección llamado el origen. Las medidas vertical y horizontal para cualquier punto son únicos y son llamados coordenada *x* y coordenada *y* respectivamente. La Figura 1.1 muestra este esquema y también permite la convención para direcciones positivas y negativas.

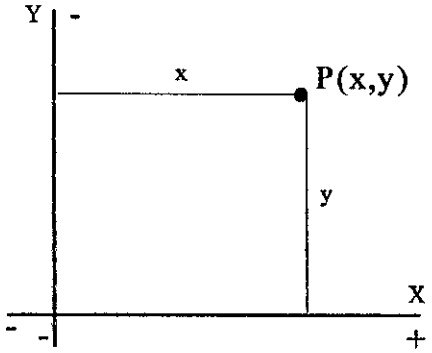



**Figura 1.1:** las coordenadas *x*, *y* del punto **P** son especificados por las distancias horizontal y vertical de **P** desde el origen. Note los signos (+,-).

Cualquier forma de 2-D puede ser representada por una secuencia de puntos o vértices como se muestra en la Figura 1.2 y, como cada vértice consiste de un par ordenado de números (coordenada-x seguido por la coordenada-y), pueden ser almacenados fácilmente dentro de una computadora. Sin embargo, nuestro conocimiento personal es visual y es establecido siguiendo las manecillas del reloj o lo contrario. Esta dirección límite debe permanecer en la computadora ordenando las coordenadas verticales en una o dos secuencias como se muestra en la Figura 1.3

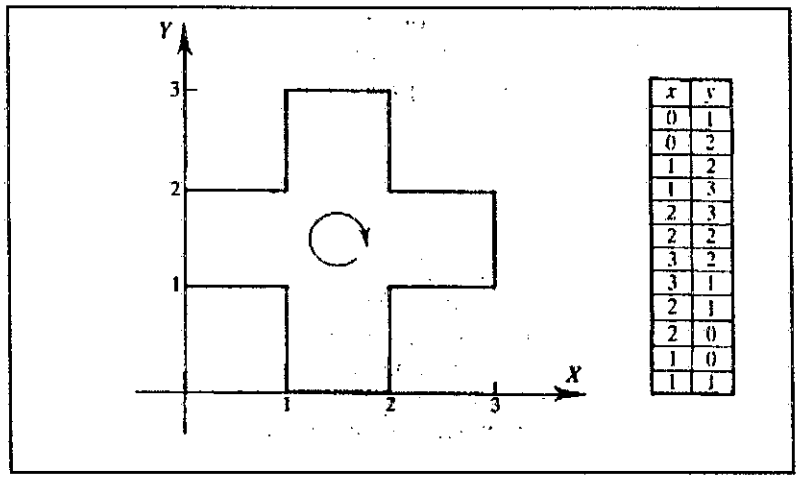



**Figura 1.2: la figura mostrada consiste de 12 vértices; sus coordenadas se muestran en la tabla. Note que los vértices fueron trazados en el sentido de las agujas del reloj.**

Las coordenadas de 2-D se agrupan entre paréntesis, p.e. (-2.5,1.5), donde 2.5 y 1.5 se refieren a la coordenada-x y coordenada-y respectivamente. Existen muchas consecuencias útiles para la notación de estas coordenadas: primero, si el valor de la coordenada es doble, su valor equivalente para una figura es dos veces su tamaño original, por lo tanto, la escala puede ser la siguiente:

$$x' = rx$$
$$y' = ry,$$

donde *(x,y)* es el vértice que es cambiado de tamaño por *r* para crear *(x',y')*. Segundo, si las coordenadas-x son incrementadas por una unidad, las nuevas coordenadas son equivalentes a la misma figura cambiada o trasladada una unidad a la derecha de la figura original. Sin embargo, una figura puede ser trasladada en la coordenada-x y la dirección-y, por lo tanto la operación de traslado se hace de la siguiente forma:

$$x' = x + u$$
$$y' = y + v,$$

donde *(x,y)* es el vértice que ha sido trasladado por *(u,v)* hacia su nueva posición de *(x',y')*.

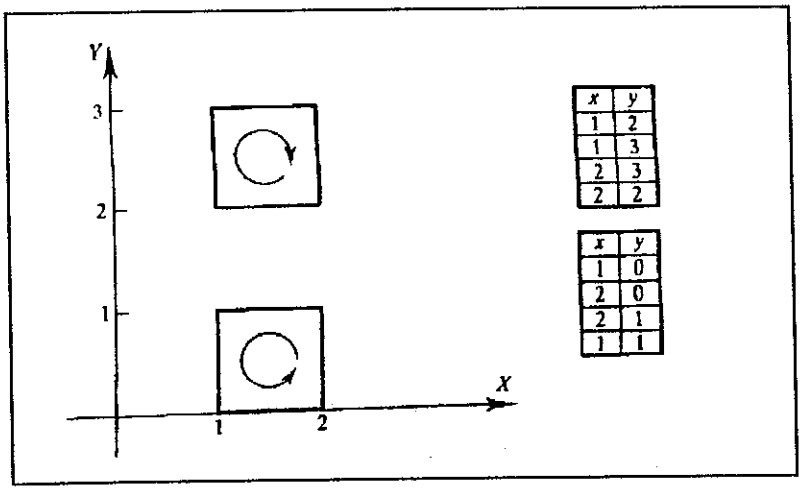



**Figura 1.3: éstos dos cuadros han sido formados en el sentido de las agujas del reloj y vice versa. Aunque ambas técnicas son válidas, cierto software acepta una de las dos formas.**

Algunas computadoras permiten ejecutar millones de operaciones aritméticas en un segundo y son, por lo tanto, capaces de cambiar y trasladar grandes conjuntos de coordenadas instantáneamente.

Aparte del traslado y cambio, la operación de rotación con respecto al origen es también importante y se logra por medio de la siguiente fórmula:

$$x' = x\cos(ß) - y\text{sen}(ß)$$
$$y' = x\text{sen}(ß) + y\cos(ß),$$

donde *(x,y)* es el punto a ser rotado por el ángulo ß hacia su nueva posición *(x',y')*. La rotación es contraria a las agujas del reloj cuando ß es positivo, y vice versa. Para rotar una figura completa por ß, las funciones coseno y seno son evaluadas primero para reducir el proceso para cuatro multiplicaciones, una suma y una resta para cada vértice. Por lo tanto, son evaluadas una vez que las coordenadas de toda la figura sean procesadas.

Cortar una figura es una transformación y se hace de la siguiente forma:

$$x' = x + y\tan(ß)$$
$$y' = y$$

donde *(x,y)* es el punto a ser cortado por el ángulo ß a su nueva posición *(x',y')*.

Estas cuatro operaciones -aumentar, trasladar, rotar y cortar- forman las operaciones fundamentales de manipulación de figuras en gráficas de 2-D y tienen una descripción mínima usando una notación matricial. Para entender esta notación, se considera la siguiente relación:

$$x' = 3x + 5y$$

Esto puede ser escrito usando una notación matricial, como:

$$x' = \begin{bmatrix} 3 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Todo lo que se ha hecho es aislar las constantes de las variables. Las constantes 3 y 5 forman un vector fila, mientras las variables *x* e *y* son llamadas un vector columna. Para evaluar *x'* debemos multiplicar el vector fila por el vector columna, el cual es obtenido multiplicando los correspondientes términos juntos y sumándolos. En este caso, multiplicamos 3 por *x* y 5 por *y*, y sumando los resultados juntos.

En general, la relación es:

$$x' = ax + by$$

es escrita como:

$$x' = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Sin embargo, la notación de las coordenadas empleadas en gráficas por computadora involucra expresiones para las coordenadas *x* e *y*, y eventualmente una coordenada-z es necesaria para un trabajo en tres dimensiones (3-D). Considere estas relaciones:

$$x' = 3x + 2y$$
$$y' = 8x + 4y$$

Pueden ser escritas en forma matricial como:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 8 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Así para evaluar el valor *x'*, multiplicamos la primera fila de la matríz por el vector columna, y para *y'* multiplicamos la segunda fila de la matríz por la columna vector. El método anterior es llamado post-multiplicación por vectores columna. Sin embargo, la operación anterior se puede hacer de la siguiente forma:

$$[x' \; y'] = [x \; y] \begin{bmatrix} 3 & 8 \\ 2 & 4 \end{bmatrix}$$

Consideremos una extensión para esta notación y veamos como la siguiente relación puede ser expresada usando matrices:

$$x' = 2x + 3y + 8$$
$$y' = 4x + 2y + 6$$

De alguna forma encontraremos un mecanismo para incluir las constantes 8 y 6. Estos términos extra pueden ser colocados de la siguiente forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 8 \\ 4 & 2 & 6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Esta notación usa coordenadas homogéneas, las cuales son usadas en geometría proyectiva; el término extra es tratado como un término para ajustar. Por ejemplo, las siguientes coordenadas homogéneas (1,2,1), (2,4,2), (3,6,3) y (5,10,5) representan el mismo punto (1,2) si el tercer término en cada vértice es dividido dentro de los dos primeros términos. En general, un punto *(X,Y)* es representado en su forma homogénea como *(HX,HY,H)*, así cuando *H=1* su valor puede ser ignorado.

Se puede fácilmente visualizar este proceso, si el espacio homogéneo es tratado como un espacio 3-D con ejes *X,Y* y *H*. Por ejemplo, se puede trazar una linea recta a través de los puntos (0,0,0), (1,2,1), (2,4,2), (3,6,3), y (4,8,4), como se muestra en la Figura 1.4. Así cualquier colección de puntos teniendo una coordenada homogénea de 1 puede ser proyectada en el plano *x-y*, con un valor homogéneo de 5, ajustando las coordenadas *x* e *y* por 5. Esto sugiere que se puede imaginar que el plano 2-D, donde la escala, traslación, rotación, y la división de operaciones, puede ser en cualquier parte del eje *H* (excepto para *H=0*). Debemos recordar dividir las coordenadas por H al final de la operación. Por conveniencia, *H = 1*.

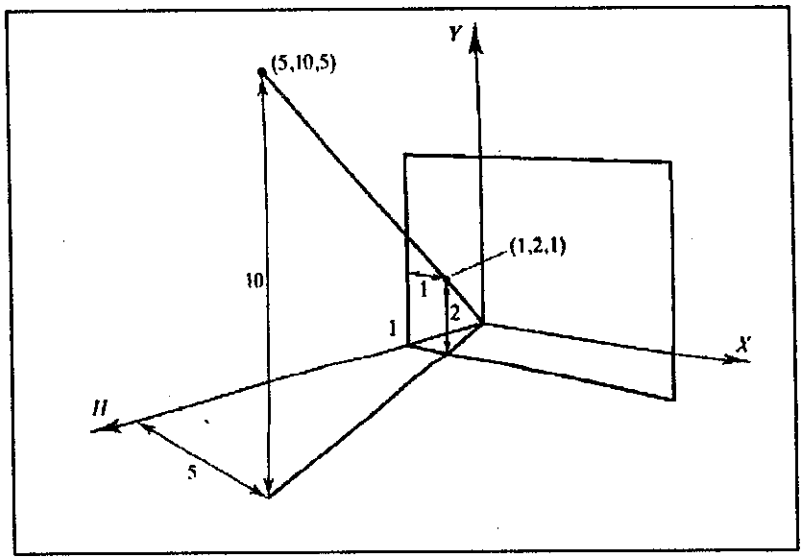



**Figura 1.4: este conjunto de ejes 3-D muestra como un punto (5,10,5) puede ser proyectado hacia atrás sobre un plano donde *H=1*. Este punto tiene coordenadas (1,2,1) y es una versión a escala del original. Forma la base de las coordenadas homogéneas.**

Regresando a la escala, traslación, y operaciones de rotación descritas anteriormente, éstas pueden ser reescritas como:

**Escala:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

**Traslación:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

**Rotación:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(ß) & -\text{sen}(ß) & 0 \\ \text{sen}(ß) & \cos(ß) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

**División:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \tan(ß) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Las matrices proporcionan una forma conveniente para describir operaciones geométricas, y también permiten dos o más operaciones para ser representadas por una simple matriz la cual es obtenida multiplicando las matrices individuales. Por ejemplo, dadas dos matrices *A* y *B*:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad B = \begin{bmatrix} r & s \\ t & u \end{bmatrix}$$

donde *AB* está definida como:

$$AB = \begin{bmatrix} ar + bt & as + bu \\ cr + dt & cs + du \end{bmatrix}$$

Este proceso de multiplicación es llamado 'concatenación', y aunque no puede ser tan obvio, *AB* no es equivalente a *BA*. Para la operación contraria tenemos:

$$BA = \begin{bmatrix} ra + sc & rb + sd \\ ta + uc & tb + ud \end{bmatrix}$$

Esta muestra claramente que los elementos de la matríz no son equivalentes.

Se muestra esto con un ejemplo. Se consideran las siguientes operaciones donde primero se rota un punto en el sentido contrario de las agujas del reloj 90° y es representado por una matríz *A*, mientras el segundo traslada el punto +1 unidad en la dirección *x* y es representada por la matríz *B*.

$$A = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Cuando se concatenan las dos matrices se debe recordar mantener la primera operación de matriz al lado derecho de la operación, lo cual significa que debe ser evaluada como *BA*, y produce:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Por lo tanto, la rotación y traslación combinada pueden ser representadas como:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Esto puede ser probado substituyento un punto arbitrario como (2,0). Una rotación de 90° lo mueve a (0,2), y la traslación lo mueve a (1,2). Haciendo uso de la matriz anterior, tenemos:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

Esta cual produce el punto (1,2) como se indicó.

Si las matrices son evaluadas en orden contrario, tenemos:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

y concatenado da:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Así el punto (2,0) será trasladado primero por (1,0) y luego rotado a 90° para producir el punto (0,3)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

Esta cual no es equivalente a la operación original.

## 1.1.2 GRÁFICAS POR COMPUTADORA EN 3-D

Ampliando las coordenadas cartesianas a tres dimensiones, se requiere que cada punto en el espacio 3-D sea localizado usando tres coordenadas. Sin embargo, ilustrándolo como un esquema con diagrama, se tiene una pequeña desventaja por lo cual es conveniente usar una imagen perspectiva para dar a entender un espacio profundo donde, en realidad, no existe una información perspectiva almacenada dentro de las coordenadas. De hecho, una técnica geométrica será usada para convertir las coordenadas de 3-D en una proyección perspectiva de 2-D.

En un conjunto 3-D, se definen ejes mutuamente ortogonales; el punto donde se cruzan en común se llama origen. La Figura 1.5 ilustra un conjunto de ejes orientadas hacia la derecha alineadas con el eje-x, luego el eje-y en medio, seguido del eje-z. Esta configuración es una notación popular dentro de las

gráficas por computadora. El sistema orientado hacia la izquierda es usado para revertir uno de los ejes. Para elegir el eje vertical, parece que no existe ninguna notación establecida, y se puede elegir como el eje-y o eje-z.

A través de todo este trabajo, se usará un sistema de 3-D orientado hacia la derecha, para describir la posición de objetos, cámaras y luces, al que se llamará Sistema Mundial de Coordenadas (WCS, World Coordinate System). Cualquier punto en el espacio 3-D es localizado por tres coordenadas *(x,y,z)* representando las distancias paralelas a lo largo de los tres ejes (Figura 1.6), y los objetos pueden ser construidos desde un conjunto de polígonos, cuyos vértices serán definidos usando esta notación de coordenadas. Si un polígono planar es creado desde una cadena arbitraria de bordes, el sentido direccional del límite depende sobre qué lado del polígono se observan los límites.

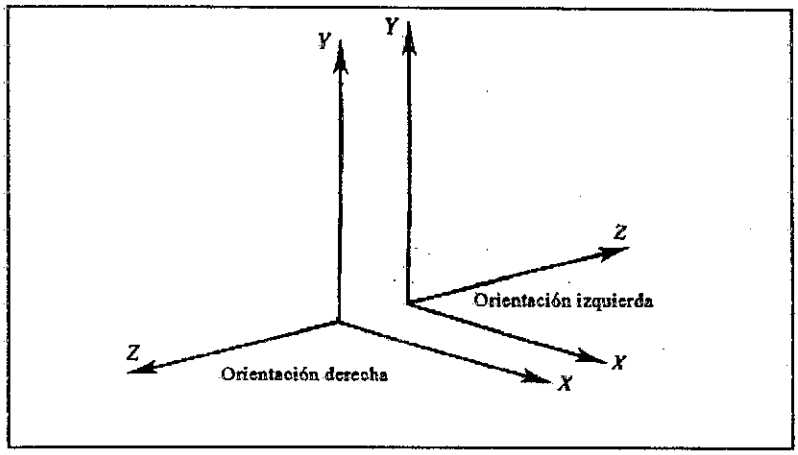



**Figura 1.5: un conjunto de ejes en 3-D puede ser descrito como orientado a la derecha o izquierda. Dependiendo de quien lo dibuja, es posible alinearlo con el eje-x, luego el eje-z como el eje central, después el eje-y. Aunque los sistemas son válidos, una notación permanente debe ser usado en el sistema de gráficas por computadora.**

Una vez más, las operaciones de escala, traslación, rotación y división pueden ser definidas usando coordenadas homogéneas. La operación de escala es:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r & 0 & 0 & 0 \\ 0 & r & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

donde *(x,y,z)* es obtenido por *r* para *(x',y',z')*.

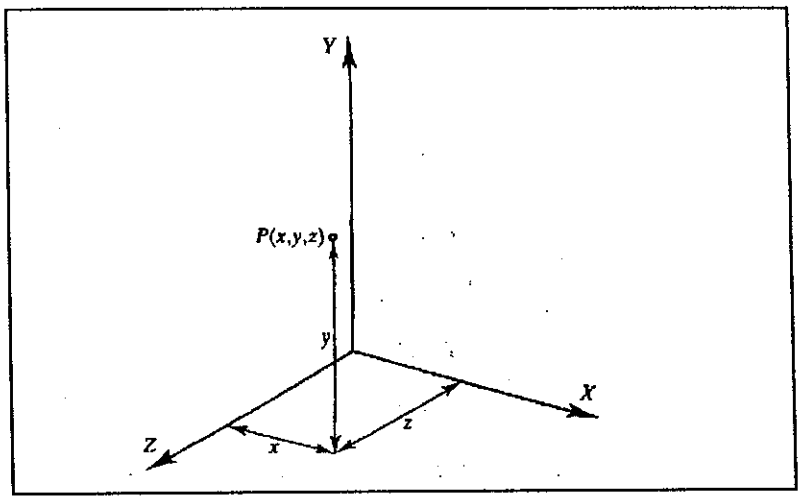



Figura 1.6: en 3-D el punto *P* es localizado por las tres coordenadas cartesianas *(x,y,z)* representando las distancias medidas paralelamente con los tres ejes ortogonales.

Otra vez, las operaciones de escala, traslación, rotación y división pueden ser definidas usando coordenadas homogéneas. La operación de escala es:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r & 0 & 0 & 0 \\ 0 & r & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Donde *(x,y,z)* es obtenido por *r* para *(x',y',z')*.

La operación de traslación es:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Donde *(x,y,z)* es obtenido por *(u,v,w)* para *(x',y',z')*.

Como un punto puede puede ser rotado alrededor de cualquier eje, se necesitan tres matrices para describir las rotaciones. Para los ejes *x*, *y* e *z*, respectivamente, éstas son:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\text{pitch}) & -\text{sen}(\text{pitch}) & 0 \\ 0 & \text{sen}(\text{pitch}) & \cos(\text{pitch}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\text{yaw}) & 0 & \text{sen}(\text{yaw}) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(\text{yaw}) & 0 & \cos(\text{yaw}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(roll) & -sen(roll) & 0 & 0 \\ sen(roll) & \cos(roll) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Donde *pitch* (ajustar), *yaw* (desviar) y *roll* (rodar) son los ángulos de rotación. El teorema de Euler establece que cualquier sistema de coordenadas *(x,y,z)* puede ser llevado en coincidencia con un segundo sistema de coordenadas *(x',y',z')* con el mismo origen por rotación a través de tres ángulos: pitch, yaw y roll.

La operación de división, al igual que la rotación, está asociada con los ejes *x*, *y* e *z*, respectivamente, así:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \tan(ß) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \tan(ß) & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \tan(ß) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Donde ß es el ángulo a tomar.

Las matrices de rotación pueden ser aplicadas individualmente para establecer los vértices para realizar una rotación por medio de los ejes. Esto crea una rotación perfectamente bien compuesta de 3-D, pero es difícil imaginar y controlar, especialmente cuando las rotaciones angulares son diferentes para cada eje. Esta rotación compuesta puede ser representada por una simple matriz, multiplicándola con matrices individuales.

Se pueden observar algunos de los problemas que resultan de la animación en 3-D cuando se describen movimientos complejos tridimensionales. Otra consecuencia de crear una rotación compuesta desde dos o tres rotaciones de ejes es que las rotaciones no son conmutativas; eso quiere decir que un conjunto de vértices rotados alrededor de los ejes *x*, *y* e *z*, no será, en general, equivalente a hacer la rotación alrededor de los ejes *x*, *z* e *y*. Las dos secuencias generalmente nos darán finalmente dos posiciones diferentes para los vértices, y la animación deberá ser consistente en especificar la secuencia de ejes sobre los que se hará la rotación.

Los objetos definidos como una colección de vértices frecuentemente necesitan ser rotados alrededor de un eje arbitrario, y como es de esperar existe una matríz que describe esta operación.

No obstante, se ha descubierto el mecanismo para describir las posiciones de los vértices en el espacio 3-D: éste será usado para construir una gran cantidad de objetos incluyendo mesas, botellas de vino, rostros humanos, agua, nubes y fuegos artificiales. Esta gran cantidad de objetos requiere igualmente de una amplia gama de herramientas de software para construirlos. Se ha visto que cuando un objeto tiene una definición de coordenadas, éstas pueden ser procesadas por matrices, las cuales efectivamente alteran su posición en el espacio. Tales técnicas serán complementadas por otros procedimientos manipulativos, los cuales

podrán doblar, distorsionar y transformar objetos para simular técnicas de la animación tradicional.

El único aspecto relevante, que queda en las gráficas de 3-D, se refiere a cómo una cámara virtual puede ser colocada en frente de los objetos de 3-D para captar su perspectiva. El dinamismo de esta cámara virtual juega un papel importante en la animación por computadora.

## 1.2 SISTEMA MUNDIAL DE COORDENADAS (WCS)

El WCS es el lugar en 3-D para que una secuencia sea animada. Es el espacio donde los modelos son colocados; las luces son colocadas para iluminar los objetos, y las cámaras son maniobradas para observar todo lo que sucede.

Algunos sistemas requerirán al usuario construir modelos en un sistema coordinado de objetos (OCS, por sus siglas en inglés), entonces con los comandos que el usuario establece las posiciones adecuadas en el WCS. En realidad, se debe recordar que los modelos no son más que listas de coordenadas, puntos de control o tal vez conjuntos de ecuaciones. Estos están en la memoria de la computadora, y para accesarlas están sujetas a las operaciones de rotación, escalar y traslación; son estas acciones las que hacen capaz de obtener las posiciones en el WCS. Cuando se desea obtener alguna copia del objeto, en lugar de replicar los datos, el objeto está sujeto a las mismas operaciones mencionadas anteriormente. Este proceso puede necesitar una gran cantidad de memoria, especialmente cuando los objetos son complejos. Una ventaja de esto, es que cuando los datos originales de las coordenadas del objeto son alteradas, todas las demás copias reflejarán también los cambios.

Como un ejemplo, se considera la situación donde un cubo ha sido creado en el OCS con origen en el centro (Figura 1.7). Si una muestra de este cubo se desea en $(x_p,y_p,z_p)$ del WCS, y tamaño duplicado, entonces la operación de matrices consiste de una escala inicial de dos, seguido de una traslación. Cualquier punto (x,y,z) está sujeto a las siguientes operaciones de matrices para transformarlo en (x',y',z'):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_p \\ 0 & 1 & 0 & y_p \\ 0 & 0 & 1 & z_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

que concatenado da:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & x_p \\ 0 & 2 & 0 & y_p \\ 0 & 0 & 2 & z_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

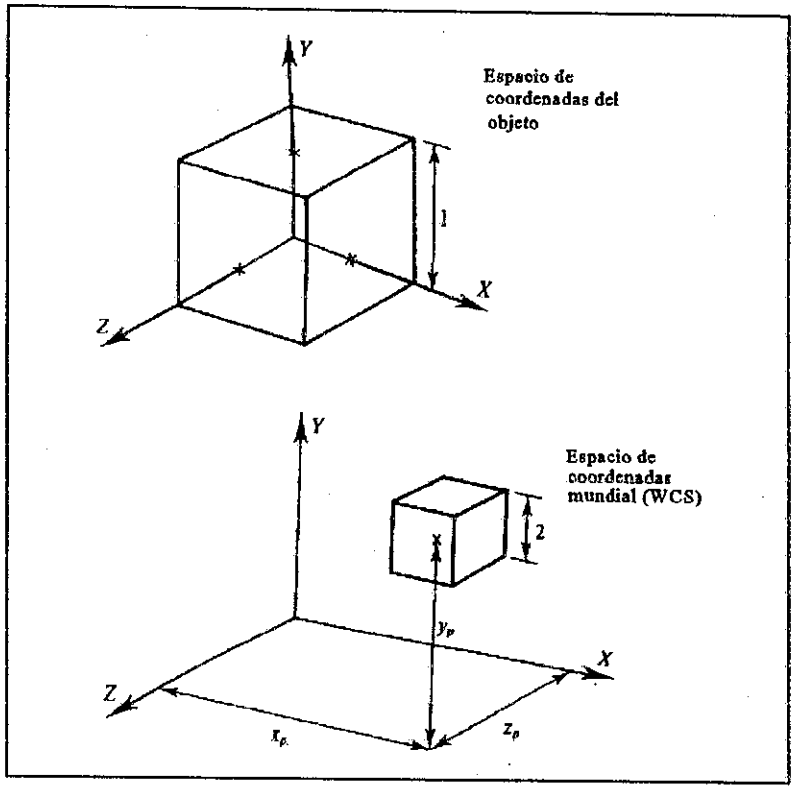



Figura 1.7: los objetos son generalmente modelados en el OCS y transformados al WCS usando alguna matriz. En este caso, una caja primero es multiplicada por 2 y luego trasladada por $(x_p, y_p, z_p)$.

## 1.2.1 EL SISTEMA DE COORDENADAS DE LA CÁMARA

En la vida real, el tomar una foto con una cámara es una simple operación; no es más que elegir una posición conveniente, apuntando la cámara en la dirección correcta, enfocando y realizando la toma. Muchas cámara modernas realizan en enfoque automáticamente. No obstante, si se usa una cámara de video, existirán problemas de como mantener la cámara firme, mantener el objetivo enfocado, mientras se mueve la cámara y el objetivo.

El concepto de una cámara en el WCS permite vistas del mundo a ser enfocado fijando su lugar y punto de enfoque, lo que proporciona suficientes datos geométricos para crear una matriz que convertirá coordenadas del WCS al CCS (sistema de coordenadas de la cámara). Fijando la posición y orientación de la cámara se puede utilizar cualquier método, como el de coordenadas cartesiandas o coordenadas polares, o una mezcla de ambos.

Moviendo la cámara a una nueva posición, se crea otro punto de vista, y uno de los requerimientos de un sistema de animación por computadora es proporcionar al animador con comandos para efectuar dichos movimientos.

La salida visual de un sistema de animación es 'ver' efectivamente por la cámara, además, una tarea importante es encontrar métodos creativos para controlar su posición y orientación. Como la cámara no tiene una existencia física, puede ser colocada en cualquier parte del espacio 3-D para describir el mundo virtual de la computadora, incluyendo el interior de los objetos. Por el momento, será conveniente colocar la cámara en el origen del sistema mundial de coordenadas (WCS) y directamente a lo largo del eje-z positivo tal y como se muestra en la Figura 1.8

Si se asume que una colección de vértices están localizados en el WCS y son

visibles para la cámara, una proyección perspectiva de estos vértices puede ser calculada asumiendo que la cámara tiene un agujero pequeño (del tamaño de un alfiler, pinhole) con una abertura localizada en el origen. Una base de vidrio imaginaria como pantalla es colocada a una distancia *d* detrás del pinhole para captar una escena invertida en perspectiva. Este fenómeno óptico de revertir la imagen no causa problemas; para estar dentro de este mundo imaginario, lo revertido puede ser corregido rotando los ejes de la pantalla como se muestra la figura 1.9.

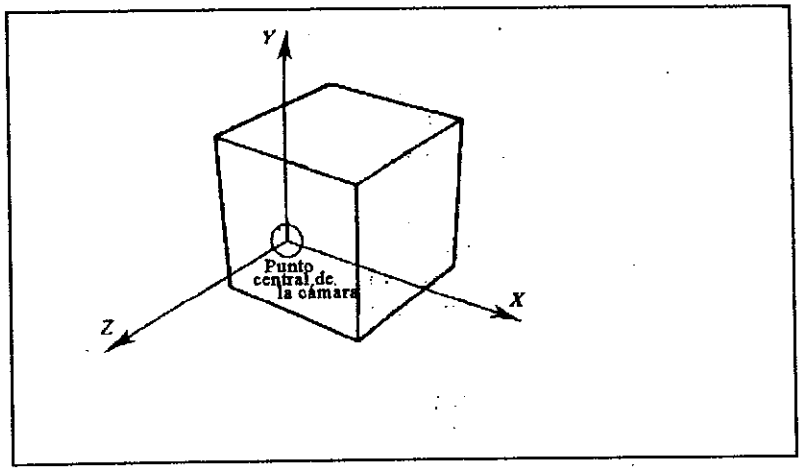



**Figura 1.8: en gráficas por computadora es conveniente usar una cámara para abarcar la acción de la imagen en perspectiva que se ha obtenido. Aquí la cámara está orientada tal que el agujero central está en el origen de los ejes y dirigido a lo largo del eje-z positivo.**

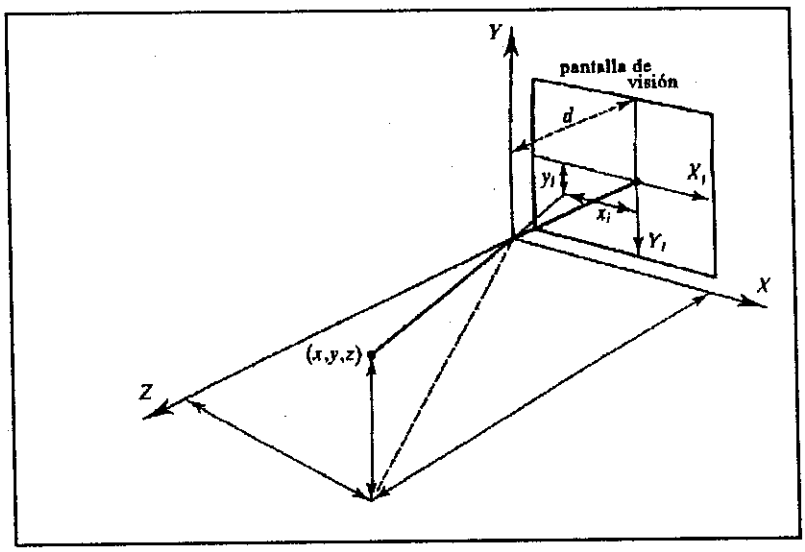



**Figura 1.9: con la pantalla posicionada a una distancia d detrás del origen, el pinhole de la cámara invierte la imagen en ambas direcciones horizontal y verticalmente. Asumiendo que la cámara virtual está posicionada al revés, la imagen tiene que ser corregida por un lado horizontal. La inversión horizontal ocurre porque el espacio coordenado está orientado hacia la derecha mientras que la imagen en el espacio está hacia la izquierda.**

Usando la geometría de triángulos semejantes, la Figura 1.10 muestra un lado y un plano de elevación del sistema de ejes WCS y el sistema de ejes de la cámara. La posición de un punto $(x,y,z)$ está localizado en $(x_i,y_i)$ sobre la pantalla usando las siguientes relaciones:

$$\frac{y}{z} = \frac{y_i}{d} \qquad \text{y} \qquad \frac{x}{z} = \frac{-x_i}{d}$$

por lo tanto:

$$x_i = \frac{-dx}{z} \qquad y_i = \frac{dy}{z}$$

Estos dos cálculos son los necesarios para una proyección perspectiva.

Hay tres observaciones sobre estos cálculos: la primera se refiere al signo negativo que aparece en el cálculo de $x_i$; esto es suficiente para el sistema orientado hacia la izquierda de los ejes usados por el espacio de la imagen de la cámara (IS siglas en inglés de image space). Si este signo cambia, no se muestra, y la imagen puede aparecer horizontalmente traspuesta. Segundo, la distancia $d$ es un término multiplicador, el cual realiza una acción escalar. Por lo tanto, si $d$ es variable, el tamaño final de la imagen se altera, y actúa igual que un control de agrandamiento (zoom) para la cámara. Tercero, las coordenadas $x$ e $y$ del punto en el WCS son divididas por su coordenada-z; esto significa que los puntos distantes tendrán un valor de $z$ muy grande y un correspondiente $x_i$ e $y_i$ pequeños sobre la pantalla. Así, como un objeto retrocede desde la cámara, su imagen se va volviendo más pequeña, lo cual sucede cuando las cosas se miran en el mundo real.

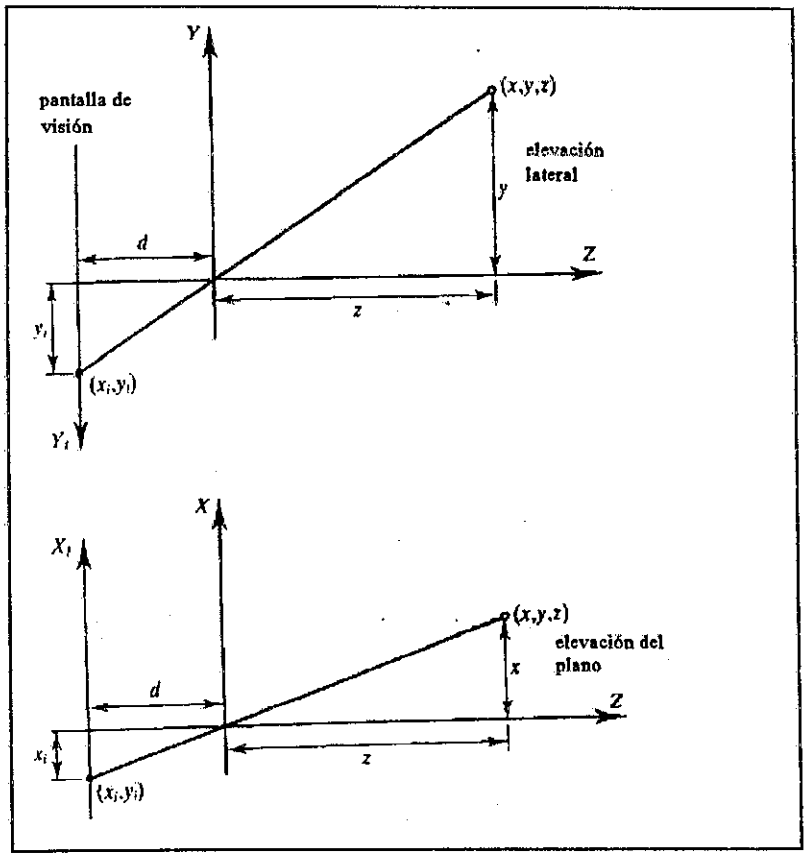



**Figura 1.10: la parte elevada (side elevation) de la pantalla muestra un triángulo relacionando $y_i$ con $d$, e $y$ con $z$. El plano de elevación (plan elevation) muestra también un triángulo que relaciona $x_i$ con $d$, y $x$ con $z$.**

Los cálculos de las perspectiva anteriores poseen una matriz, la cual es:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \begin{bmatrix} -d/z & 0 & 0 & 0 \\ 0 & d/z & 0 & 0 \\ 0 & 0 & d/z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Donde se observa que los puntos del sistema de coordenadas de la cámara (CCS sus siglas en inglés) se transforman en IS como:

$$x_i = \frac{-dx}{z}$$

$$y_i = \frac{dy}{z}$$

$$z_i = d$$

Nótese que el valor de $z_i$ es igual a la distancia en el plano.

Desafortunadamente, esta matriz contiene referencias a la coordenada-z del vértice a ser proyectado, y esto debe ser eliminado si la matríz tiene algún valor independiente. De hecho, si multiplicamos todo por $z/d$, toda la relación será simplificada. Como se ha trabajado con coordenadas homogéneas, el lado izquierdo de la operación puede ser cambiado modificando el término escalar homogéneo. La matriz es cambiada multiplicando cada término como sigue:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ z/d \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & z/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Pero aun esta matriz contiene una referencia a la coordenada-z; sin embargo, puede ser eliminada completamente y da como resultado la siguiente matríz:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ z/d \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & z/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Ahora se tiene una matriz simple para producir una proyección perspectiva que solamente depende de la distancia sobre el plano.

Alguien que conozca de cámaras sabrá que la longitud de enfoque del lente controla su campo de visión, y consecuentemente el grado de perspectiva obtenido en la imagen final. Esto también puede ser simulado dentro de la computadora. Para entender esto, hay que considerar la geometría de la pantalla de la cámara con el tamaño de la imagen, como se muestra en la Figura 1.11. Si la altura máxima de la imagen es $d$ (la cual es la distancia de la pantalla desde el pinhole) el ángulo de visión es aproximadamente 53°. Si el tamaño de la imagen es mantenido, pero la distancia de la pantalla es dividido ($d/2$), el ángulo de visión aumenta a 90°, creando un ángulo de efecto más amplio. Se puede crear un campo de visión angosto manteniendo el tamaño de la imagen constante y aumentando la distancia de la pantalla.

Este ajuste, que se puede implementar dentro del sistema de animación, significa que se puede ajustar dinámicamente la longitud de enfoque de la cámara durante una secuencia de animación.

La descripción anterior asume que la cámara está situada en el origen del WCS, y dirigida a lo largo del eje-z. Esto es conveniente para calcular la perspectiva de la imagen, pero es bastante restrictiva para realizar cualquier animación. Se puede ampliar este modelo tal que la cámara pueda ser colocada en cualquier lugar en el espacio y enfocar hacia cualquier punto. La siguiente descripción utiliza un método simple de colocar la cámara usando ángulos de roll, pitch y yaw, aunque existen otros métodos.

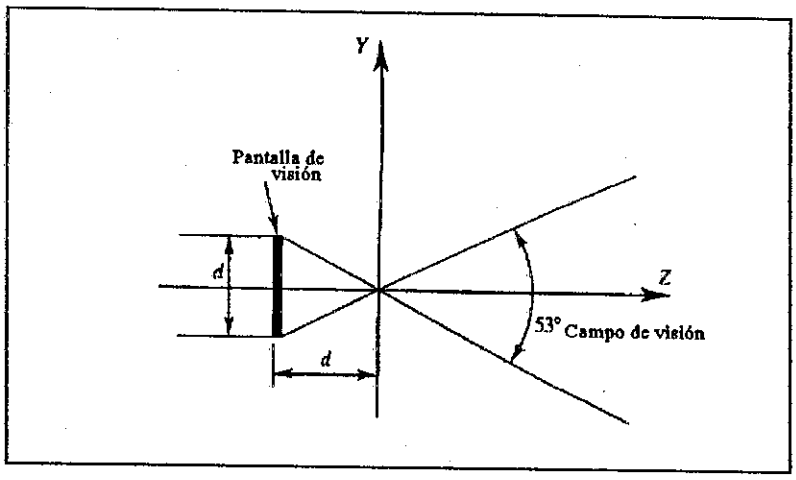



**Figura 1.11: si la pantalla de visión es colocada *d* unidades desde el origen y el tamaño de su imagen es también *d*, entonces el campo efectivo de visión es aproximadamente 53°. Como la distancia de la pantalla y el tamaño de la imagen pueden ser cambiados independientemente, el campo de visión de la cámara puede ser cambiado dinámicamente durante una secuencia de animación.**

Para empezar, hay que imaginar que el sistema de ejes de la cámara está alineado como se muestra en la Figura 1.12a. La cámara puede ser movida hacia otro punto en el espacio con una orientación arbitraria aplicando una rotación roll alrededor del eje-z (Figura 1.12b), seguido de una rotación pitch alrededor del eje-x (Figura 1.12c), y de una rotación yaw alrededor del eje-y (Figura 1.12d). Finalmente, la cámara es trasladada por $(t_x, t_y, t_z)$ (Figura 1.11e). La posición de la cámara es determinada por las cuatro operaciones: roll, pitch, yaw y traslación.

Hay que considerar, entonces, una cámara orientada hacia cualquier lugar en el espacio usando este método, y dirigida hacia algún grupo de objetos. Si se revierte la secuencia de roll, pitch, yaw y traslación, la cámara retrocederá a su alineación original con el WCS. Además, si los objetos son movidos en la misma dirección, la relación entre la cámara y objetos no cambiará. Pero lo importante es el movimiento de los objetos al retroceder a la alineación de CCS (Camera Coordinate System) con el WCS, que es fácil realizar la proyección perspectiva. Además, como la cámara no es realmente una entidad física; todo lo que se necesita realizar es respecto a las coordenadas de los objetos para revertir el conjunto de operaciones que se aplicaron inicialmente a la cámara. Esto implica una operación de traslación seguida de una rotación yaw, pitch, y roll.

Dados estos tres ángulos y que la posición de la cámara es roll, pitch, yaw y $(t_x, t_y, t_z)$, las cuatro matrices siguientes realizarán los movimientos deseados, y luego regresará la cámara al origen:

Traslación $(-t_x, -t_y, -t_z)$:

$$\begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación -*yaw* alrededor del eje-y:

$$\begin{bmatrix} \cos(yaw) & 0 & \text{sen}(yaw) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(yaw) & 0 & \cos(yaw) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

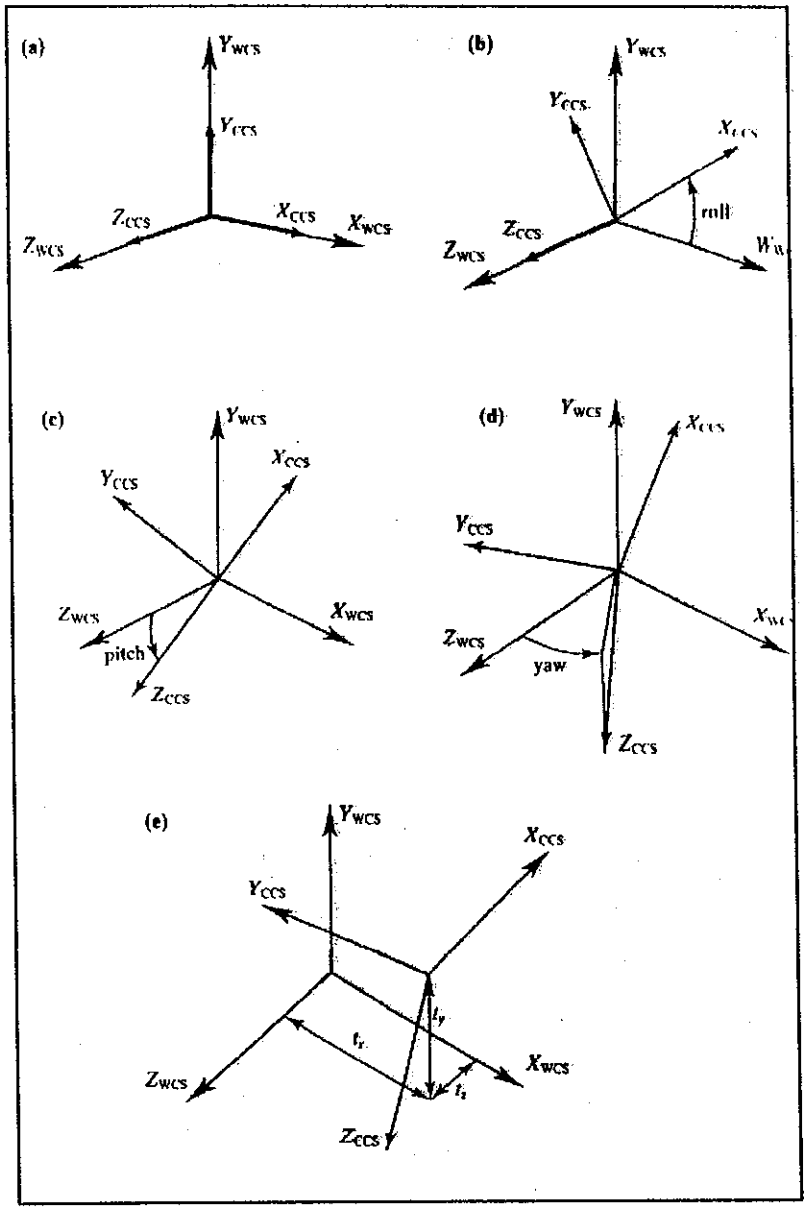



**Figura 1.12**: estos cinco diagramas ilustran como la cámara puede ser colocada en el espacio con una orientación arbitraria: (a) muestra el CCS alineado con el WCS; (b) muestra el CCS después de la rotación roll; (c) muestra la posición del CCS después de una rotación pitch; (d) muestra el CCS después de una rotación roll; (e) muestra el CCS después de la traslación de $(t_x, t_y, t_z)$.

Rotación *-pitch* alrededor del eje-x

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\text{pitch}) & -\text{sen}(\text{pitch}) & 0 \\ 0 & \text{sen}(\text{pitch}) & \cos(\text{pitch}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación *-roll* alrededor del eje-z

$$\begin{bmatrix} \cos(roll) & -sen(roll) & 0 & 0 \\ sen(roll) & \cos(roll) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Las matrices anteriores pueden ser simplificadas usando las siguientes identidades de trigonometría:

$$-sen(ß) = \sin(-ß)$$
$$\cos(ß) = \cos(-ß)$$

Rotación -yaw alrededor del eje-y:

$$\begin{bmatrix} \cos(yaw) & 0 & -sen(yaw) & 0 \\ 0 & 1 & 0 & 0 \\ sen(yaw) & 0 & \cos(yaw) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación -pitch alrededor del eje-x

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(pitch) & sen(pitch) & 0 \\ 0 & -sen(pitch) & \cos(pitch) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Las matrices anteriores pueden ser simplificadas usando las siguientes identidades de trigonometría:

$$-sen(ß) = \sin(-ß)$$
$$\cos(ß) = \cos(-ß)$$

Rotación -roll alrededor del eje-z

$$\begin{bmatrix} \cos(roll) & sen(roll) & 0 & 0 \\ -sen(roll) & \cos(roll) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para obtener una simple matriz para trasladar puntos del WCS al CCS las rotaciones de las matrices anteriores son concatenadas con la matríz de traslación en la secuencia: traslación, yaw, pitch y roll. Esto produce la matriz siguiente:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{bmatrix}$$

donde:

$$
\begin{aligned}
T_{11} &= \cos(yaw)\cos(roll) + \text{sen}(yaw)\text{sen}(pitch)\text{sen}(roll)\\
T_{12} &= \cos(pitch)\text{sen}(roll)\\
T_{13} &= -\text{sen}(yaw)\cos(roll) + \cos(yaw)\text{sen}(pitch)\text{sen}(roll)\\
T_{14} &= -(txT_{11} + tyT_{12} + tzT_{13})\\
T_{21} &= -\cos(yaw)\text{sen}(roll) + \text{sen}(yaw)\text{sen}(pitch)\cos(roll)\\
T_{22} &= \cos(pitch)\cos(roll)\\
T_{23} &= \text{sen}(yaw)\text{sen}(roll) + \cos(yaw)\text{sen}(pitch)\cos(roll)\\
T_{24} &= -(txT_{21} + tyT_{22} + tzT_{23})\\
T_{31} &= \text{sen}(yaw)\cos(pitch)\\
T_{32} &= -\text{sen}(pitch)\\
T_{33} &= \cos(yaw)\cos(pitch)\\
T_{34} &= -(txT_{31} + tyT_{32} + tzT_{33})\\
T_{41} &= 0\\
T_{42} &= 0\\
T_{43} &= 0\\
T_{44} &= 1
\end{aligned}
$$

Además, cualquier punto $(x,y,z)$ en el WCS puede ser convertido a $(X,Y,Z)$ en el CCS de la siguiente forma:

$$
\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

## 1.3 DELIMITACIÓN DEL CAMPO DE VISIÓN DE LA CÁMARA

Una cámara convencional puede solamente capturar una imagen de una escena que está físicamente delante o dentro del campo de visión del lente. Sin embargo, la cámara matemática usada en gráficas por computadora efectivamente 'capta' todo. Para apreciar esto, hay que considerar un punto en el CCS con una coordenada-z negativa. Cuando este valor de $z$ es substituido en las ecuaciones de proyección perspectiva los valores de $x_1$ e $y_1$ tendrán signos opuestos a un punto correspondiente teniendo una coordenada-z positiva. Si esto es permitido, la imagen final va a aparecer algo distorsionada.

Obviamente, los vértices detrás de la cámara tienen que ser removidos antes de que sean transformados en una proyección perspectiva, pero solamente con remover dichos puntos con una coordenada-z negativa, no será suficiente. Los bordes que conectan dos vértices deben ser removidos para obtener una parte visible. Existen otros bordes que están distantes y que no son visibles, y no pueden ser removidos. De hecho, existen tres clases de vértices que requieren ser reducidos: los vértices demasiado cercanos a la cámara, los vértices demasiado alejados de la cámara y los vértices frente a la cámara, y que se encuentran fuera del campo de visión de la misma. Si uno delimita el volumen de espacio que contiene los vértices que son visibles para la cámara, tendrá seis planos: dos delimitando los planos cercanos y lejanos, y cuatro planos restantes que la rodean.

La Figura 1.13 ilustra esta configuración, que es llamada visión frustum. Para empezar, hay que considerar el problema de identificar aquellos objetos que estarán completamente invisibles para la cámara. Asumiendo que $W = 1$, y el plano de proyección, donde $z = 1$, es el plano más cercano, entonces un objeto cuyas coordenadas-z son todas menores que 1 pueden ser totalmente reducidas. Cuando esto sucede en la práctica, es que un parámetro es colocado en la base de datos manteniendo la geometría, e indica que es invisible y no deberá ser mostrado. Similarmente, los objetos cuyas coordenadas-z todas exceden la distancia más allá

del plano, también son invisibles. Ahora ya se está en los planos izquierdo, derecho, superior e inferior. Refiriéndose a la figura 1.14a, se observa una visión del plano del frustum, y se puede observar que cualquier punto $(x_i,y_i,z_i)$ es invisible cuando $x_i/z_i >= 1/2$. Similarmente, cuando $x_i/z_i <= -1/2$. Esto puede ser escrito como:

$$2x_i >= z_i$$
$$2x_i <= -z_i$$

Refiriéndose a la Figura 1.14b solamente se puede observar que cualquier punto $(x_i,y_i,z_i)$ es invisible cuando $y_i/z_i >= 1/2$. Similarmente, cuando $y_i/z_i <= -1/2$. Esto también, puede ser escrito como:

$$2y_i >= z_i$$
$$2y_i <= -z_i$$

Además, un objeto puede ser completamente reducido, si todos sus vértices satisfacen una de las relaciones anteriores. Si las coordenadas no son ajustadas al tamaño de la ventana W, entonces las condiciones de invisibilidad son:

$$\frac{2x_i}{W} >= z_i$$
$$\frac{2x_i}{W} <= -z_i$$
$$\frac{2y_i}{W} >= z_i$$
$$\frac{2y_i}{W} <= -z_i$$

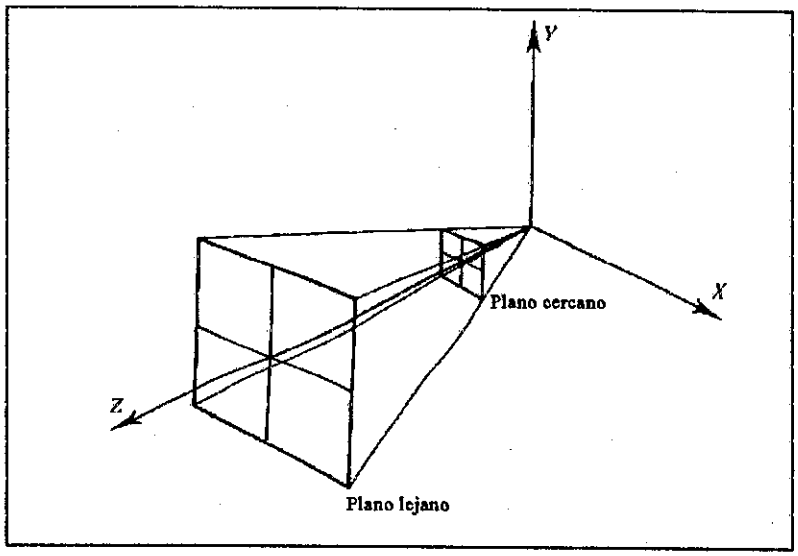



**Figura 1.13**:la visión frustum define el volumen de puntos visibles para la cámara. El plano cercano es usado para recortar objetos que están demasiado cerca y detrás de la cámara, mientras que el plano más lejano recorta aquellos objetos que están demasiado lejos para verlos. Cuatro de los planos recortan objetos que están parcialmente o completamente fuera del campo de visión de la cámara.

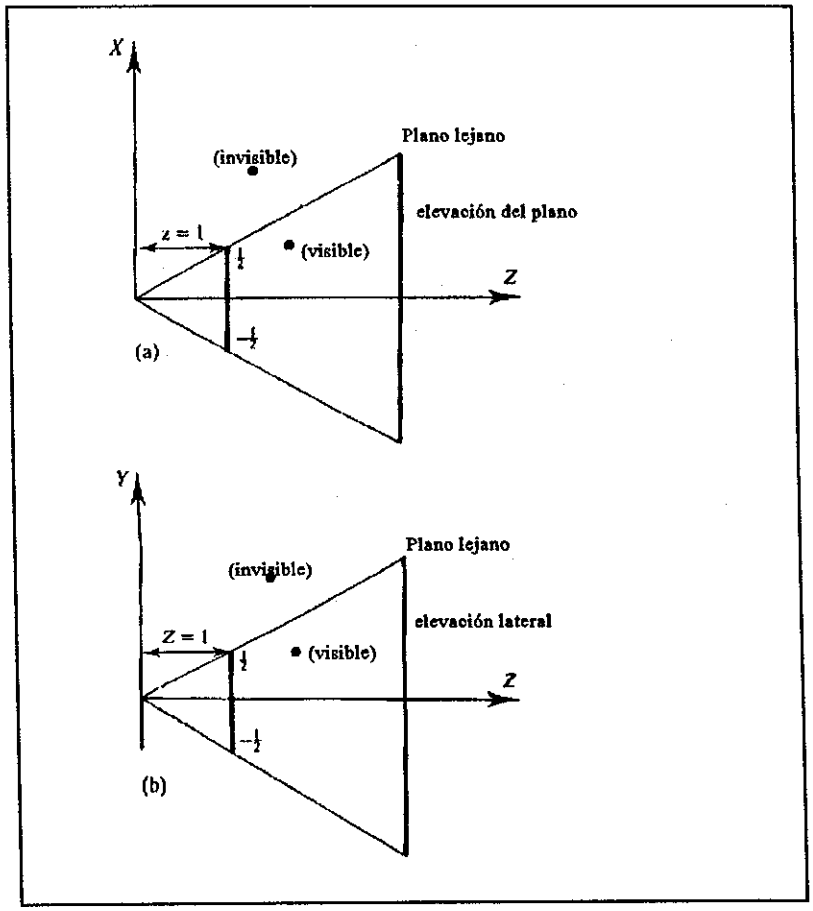



**Figura 1.14**: el plano de elevación del CCS muestra como el plano que es reducido determina aquellos puntos visibles y aquellos puntos invisibles para la cámara. En el lado de elevación del CCS los planos recortados superior e inferior forman una división similar.

Los objetos que pasan la prueba anterior serán total o parcialmente visibles.

Ahora, se puede colocar la cámara en el WCS y apuntarla a una colección de objetos y calcular una visión de la perspectiva. De hecho, cinco etapas están involucradas en este proceso:

1) localizar y posicionar la cámara;
2) calcular la matriz para convertir los vértices del WCS al CCS;
3) convertir cualquier vértice desde el WCS hacia CCS usando la matriz;
4) reducir las partes innecesarias de los objetos;
5) realizar la transformación perspectiva.

Así cada vez que la cámara o la escena cambia, se deberán ejecutar los cinco pasos anteriores.

Un Sistema de Animación por Computadora debe básicamente proveer una variedad de herramientas para construir objetos desde unos vértices en 3-D y posiciones de la cámara. Además, deben existir facilidades para mover los objetos y la cámara de una forma controlada, y tener como salida una imagen en tiempos predeterminados.

# 1.4 HARDWARE

## 1.4.1 DIGITALIZADOR

Se ha visto cuán importante son las coordenadas cartesiandas en el diseño de gráficas por computadora, el primer dispositivo que se debe mencionar es el digitalizador, el cual genera coordenadas que son bastante aceptables en el uso de la computadora.

El digitalizador como se muestra en la Figura 1.15 consiste de una superficie activa, una aguja, y una unidad de control. La superficie activa está generalmente construida de alambres conectados horizontal y verticalmente en forma de matriz, el cual posee un campo electromagnético con un nivel bajo de radiación. Este campo es codificado con señales, tal que cuando una aguja es colocada en la proximidad de la superficie, una bobina dentro de la aguja actúa

en forma aérea y regresa una señal a la unidad de control. Este decodifica la señal y calcula las coordenadas-x y coordenadas-y de la aguja, y las transmite al computador central (host) . Las conexiones dentro del digitalizador están bien unidos, así que no se pueden mover y están protegidas por una superficie laminada.

Para crear algún diseño, un programa en el computador central (host) debe recibir antes los números desde el digitalizador, para poder montar el diseño. El programa ha sido diseñado tal que, antes de crear el diseño, el usuario indica en el programa la localización precisa del origen. Este podrá ser colocado en medio del diseño. Una vez que se ha hecho esto, el usuario procede a trazar los contornos tocando el bolígrafo en ciertos puntos. Cuando el digitalizador es operado en un modo 'point mode', cada vez que el usuario presiona la aguja en el digitalizador, un interruptor interno es operado, el cual manda a la unidad de control del digitalizador la transmisión de la posición actual del bolígrafo. En el 'continuos mode', el sistema se sale de las coordenadas a una velocidad controlable, mientras el interruptor del bolígrafo está en estado de encendido.

Generalmente una parte del diseño consiste de un número distinto de contornos, y al final del contorno y al inicio de otro deben estar comunicados por el programa principal. Mientras estas coordenadas son alimentadas al computador, el programa está construyendo una lista de series de coordenadas que representan un ejemplo de la versión del diseño. Obviamente si el usuario del digitalizador no sigue exactamente los contornos del diseño, las coordenadas almacenadas internamente tendrán errores.

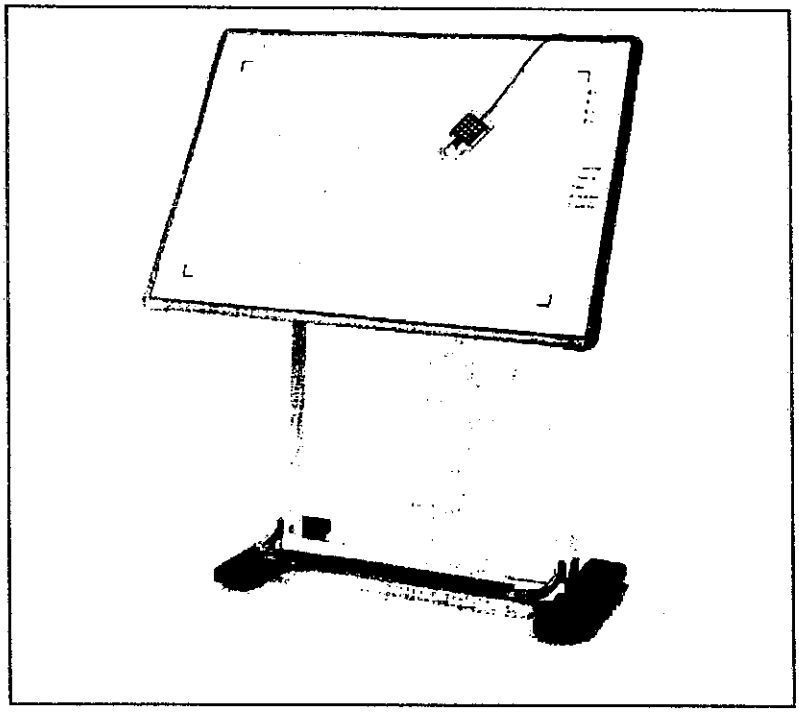


**Figura 1.15**: el digitalizador utiliza un disco para probar el área activa del dispositivo el cual proporciona en tiempo real al computador central con coordenadas 2-D.

## 1.4.2 DIGITALIZADOR 3-D

Cuando el detalle de un modelo sólido necesita ser capturado, entonces el digitalizador 3-D es muy útil. Primero, el modelo se prepara dibujando una red de polígonos sobre la superficie, estando seguro que los polígonos más pequeños son usados para áreas de detalles finos. Se hará una prueba para identificar los

vértices de los polígonos en alguna secuencia específica. El digitalizador utiliza dos pequeñas bobinas colocadas dentro del volumen que contiene el modelo para radiar una señal, la cual es recibida por la prueba; con estos datos, un procesador calcula la posición de la prueba en el espacio, la cual es enviada al computador principal (host). El software en el computador principal puede organizar los datos para que los polígonos originales puedan ser reconstruidos.

## 1.4.3 DIGITALIZACIÓN MANUAL

Un digitalizador convencional es una pieza de equipo en cualquier computadora de animación, que puede ser usada en una variedad de formas para ingresar figuras que forman un modelo en 3-D. Se ha visto que en las técnicas de modelación de superficies, la digitalización manual proporciona un método conveniente para suministrar la guía de formas para estas herramientas.

El tamaño de cualquier figura que es manualmente digitalizada es muy importante, porque un digitador es capaz solamente de trabajar para una cierta resolución (0.02 mm); esto, junto con los errores producidos en la etapa de dibujar y de aquellos introducidos por el digitador pueden resultar visibles en el modelo final. Para mantener estos errores en un nivel de aceptación, el trabajo de arte original podrá ser hecho a gran escala; esto asegura que los errores son, en proporción a las dimensiones de las coordenadas, muy pequeños. Una vez dentro del ambiente de la modelación, estas coordenadas pueden ser cambiadas de tamaño para obedecer con las dimensiones físicas asociadas con el WCS.

Los sistemas CAD también pueden ser usados para diseñar modelos 3-D donde las elevaciones son dibujadas para un objeto e ingresadas usando comandos específicos del software. Dichos sistemas permiten el ingreso de la geometría de la superficie de objetos donde los tres valores de las coordenadas son obtenidos por la interface. Esto lo logra el usuario seleccionando una elevación en particular que permita al software identificar donde la vista es descrita en coordenadas *x* e *y*, coordenadas *y* e *z* o coordenadas *x* e *z*, e identificando el mismo punto en dos elevaciones cualquiera; las tres coordenadas del punto pueden ser extraídos por el software y usados para construir un modelo coherente en 3-D.

## 1.4.4 PLOTTER DE GRÁFICAS

Una plotter revierte la acción del digitalizador en que, si una secuencia de coordenadas es mantenida dentro del computador y enviarlas a la plotter, el bolígrafo de la plotter trazará sobre el papel, el diseño codificado por las coordenadas. Existen diferentes tamaños de plotters que varían de A3 a A0, pueden dibujar con bolígrafos de fibra de vidrio o bolígrafos especiales para dibujo.

Algunas plotters son capaces de dibujar a velocidades de 1m/seg, pero esto sólo puede ser obtenido cuando las líneas son relativamente largas. Si los segmentos de líneas son muy cortas, los motores de la plotter no podrán acelerar al máximo de la velocidad para dibujar, y sólo se obtendrá una velocidad promedio para dibujar.

El llenar áreas con color representa un problema para el bolígrafo de la plotter; es por eso que otros tipos de dispositivos de impresión tienen la capacidad de convertir imágenes de la pantalla a color sobre el papel. Similarmente; existen plotters electroestáticas (monocromo y color) que transforman un archivo de datos de coordenadas en un formato raster, tal que la imagen final en la plotter aparece cuando el papel está finalmente libre de la máquina.

## 1.4.5 ALMACENAMIENTO DE CUADROS

Un dispositivo de almacenamiento tal como un formato es llamado un frame store (almacenamiento de cuadros) o frame buffer. Este tiene un elemento de memoria para cada color del pixel, y una construcción emplea un byte de memoria para este elemento. Esto significa que para almacenar todos los colores de una imagen que consiste de 1024 líneas, donde cada línea tiene 1024 pixels, es necesaria 1 Mb de memoria para cada separación de color, y hace un total de 3 Mb. Cuando el frame store es conectado a la computadora, el software residente puede cargar los números que representan los niveles de rojo, verde y azúl para cualquier pixel. Mientras sucede esto, la memoria del frame store será recorrida a la velocidad del video y la señal digital es convertida en una forma análoga.

Como un byte de la memoria puede almacenar cualquier número entre 0 y 255, puede representar la intensidad de los componentes almacenados del rojo, verde o azul. Esto significa que un pixel puede tener 256 niveles de rojo, 256 niveles de verde y 256 niveles de azul; juntos hacen 256 x 256 x 256 diferentes combinaciones posibles de colores, que es aproximadamente 16.7 millones en total. Esto puede ser excesivo pero la industria de efectos especiales utiliza 12 bits para los colores primarios ya que sus imágenes son transferidas a películas de 35 mm.

Además, los números individuales almacenados dentro de los tres bytes del pixel pueden representar las intensidades de los colores primarios; los números son frecuentemente usados para direccionar indirectamente el color final a través de un look-up table (LUT) como se muestra en la Figura 1.16. Como los LUT tienen 256 entradas, los valores del frame store podrán ser interpretados como posiciones dentro de las tablas, las cuales contienen las intensidades de los colores.

A primera vista, esto no parece lograr nada, pero su referencia indirecta al color final tiene ciertas ventajas. Primero, significa que los valores en las tablas no necesitan aumentarse linealmente uno por uno, pero son ajustados por medio de un concepto no lineal; este procedimiento es usado para una corrección y control de cualquier no linealidad que pudiera existir en el sistema. Segundo, las tablas pueden ser cargadas con valores arbitrarios para que la imagen desplegada pierda toda la información del color original; esto es llamado pseudo-color y es encontrado frecuentemente en aplicaciones de procesamiento de imágenes. Tercero, para ciertas imágenes, si las entradas de las tablas son movidas más de una posición, y reciclando dichas entradas que terminan al final de la tabla, se crea un efecto de pseudo-animación conocido como una animación LUT.

## 1.4.6 MONITOR A COLOR

Un monitor de color es muy similar a una televisión: ambos utilizan tubos de rayos catódicos para desplegar la imagen a color, pero difieren en el tipo de señales de entrada. Una televisión acepta señales en un formato de PAL, NTSC o SECAM, donde la información de sonido y color son codificados juntos; donde un monitor acepta las señales componentes para rojo, verde y azúl, y el sonido de su propio canal.

Algunos monitores pueden desplegar sus imágenes en un solo recorrido de la pantalla, y producir 60 o más cuadros cada segundo. Este modo de despliegue es llamado no-interconexión, donde una imagen de televisión está comprendida de dos campos interconectados para formar un simple cuadro. Un campo es formado por líneas pares o impares de la imagen.

# 1.4.7 CINTA DE VIDEO

La mayoría de la animación por computadora es registrada en una cinta de video, la calidad no es aceptable para ser vista en una pantalla de cine, que es el lugar donde es mostrada. Un sistema de cintas de bajo costo pueden ser logradas filmando la pantalla del monitor, un cuadro a la vez. Esto requiere de filtros contra la luz para proteger la cámara y la pantalla de los reflejos. Desafortunadamente, la imagen es distorsionada por la curvatura de la pantalla. El monitor puede no ser capaz de mantener una intensidad constante sobre largos períodos de tiempo.

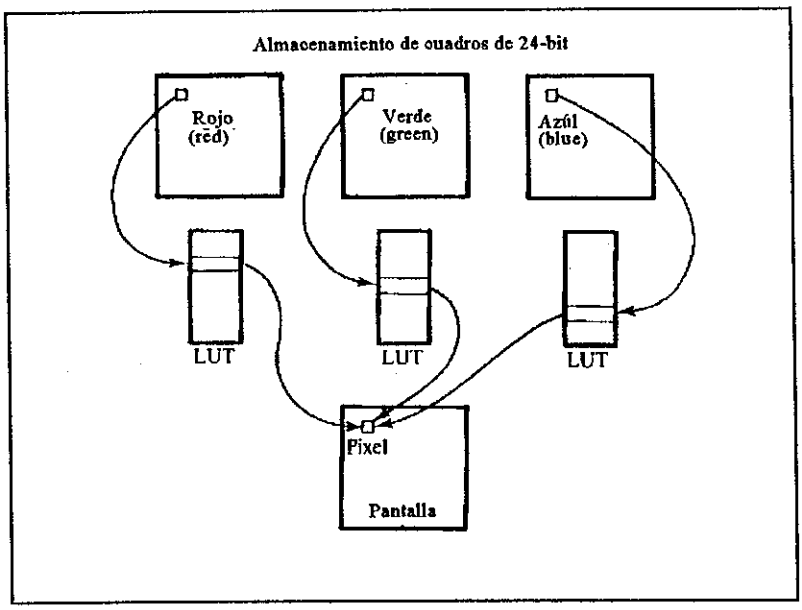



**Figura 1.16:** el frame store utiliza look-up tables (LUT) para identificar la intensidad final del rojo, verde y azúl asociados con un pixel. Esto proporciona un mecanismo para una corrección extra, el pseudo-color y la LUT de animación donde los valores de la tabla son circulares.

Para sobreponerse de lo anterior, unas cintas especiales pueden ser desarrolladas sobre pantallas planas montadas sobre un ambiente sin reflejos. Los monitores, que son monocromáticos, automáticamente despliegan las tres separaciones de colores. Por ejemplo, la señal roja crea una imagen monocromática, la cual es fotografiada a través de un filtro rojo. Después de esta exposición, la señal del verde crea otra imagen monocromática, la cual también es fotografiada y finalmente es agregado el componente azul. Así un simple cuadro es creado desde tres exposiciones de tres separaciones de color. Para el siguiente cuadro, la cinta es automáticamente adelantada y se repite el proceso.

# 1.4.8 WORKSTATIONS

Actualmente, la filosofía de las estaciones de trabajo (Workstations) es dar al usuario individual suficiente poder de procesamiento. El usuario puede diseñar y probar programas localmente lo cual, si crean gráficas, las pueden desplegar a altas velocidades. Las workstations tienen los siguientes componentes: algunos son monocromo, mientras que otros pueden desplegar entre 256 y 16.7 millones de colores. La resolución puede ser variable, pero es aproximadamente de un millón de pixels.

# CAPITULO 2

# ESQUEMAS DE MODELOS

## INTRODUCCIÓN

En el mundo real, se sabe que cada cosa es hecha de átomos, que se juntan para formar moléculas y forman objetos tales como manzanas, niebla, etc. El animador desea crear muchos objetos dentro de la computadora, pero su memoria solamente puede almacenar códigos eléctricos que representan números o símbolos, los cuales pueden ser utilizados para representar otras estructuras tales como coordenadas cartesianas, ecuaciones, volúmenes de espacio, inclinaciones, propiedades de superficies y atributos físicos tales como tensión, velocidad y blandéz, juntándolos todos proporcionan herramientas para modelar una variedad de objetos. Para arreglárselas con las diferentes estructuras físicas de objetos familiares, la modelación por computadora consiste de un amplio rango de técnicas diferentes. Por ejemplo, un simple cubo tiene seis lados planares, ocho vértices y doce bordes. (Esto satisface la regla de Euler la cual establece que para cada poliedro sin agujeros, el número de bordes es también dos menos que la suma de los lados y vértices). Por lo que un objeto puede ser representado en muchas formas, y quizá la más simple es construir un objeto basado en sus esquinas usando coordenadas cartesianas; estos datos pueden ser usados para construir los bordes, los cuales formarán los lados. Finalmente, estos pueden ser arreglados para definir la superficie del cubo, dicho modelo es conocido como una representación preliminar. Pero podría la misma técnica ser usada para modelar una esfera?. Bueno, la respuesta es si, pero la esfera deberá ser construida usando un gran número de superficies planas. Otro modelo para representar la esfera podría ser la siguiente ecuación:

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

donde r es el radio y *(x,y,z)* es cualquier punto sobre la superficie de la esfera. Este método implícito de definir la superficie requiere que los valores de *x*, *y* e *z* tienen que ser exactos para satisfacer el valor de r. Este método no solamente identifica los puntos sobre la superficie, sino que además determina aquellos puntos dentro y fuera de la esfera, por lo tanto será usado para modelar esquemas volumétricos.

En este capítulo, se tratará sobre algunos de los modelos de esquemas que pueden ser usados en la Animación por Computadora.

## 2.1 WIRE FRAME

Uno de los métodos más simples para crear objetos en 3-D, se basa en la construcción de líneas que representan los bordes que identifican la geometría del objeto. Dicho método es conocido como wire frame, en donde el objeto es construido físicamente con pequeñas piezas de alambre. La Figura 2.1 ilustra una mesa construida con este método.

Dichos objetos son construídos a base de un conjunto no coordinado de líneas, aunque esto puede ser imposible probar por medio de programas que automáticamente eliminan dichas líneas potencialmente cubiertas por otras superficies. Esto es porque los datos numéricos no almacenan información asociada con los bordes de sus superficies. Consecuentemente, el wire frame no es considerado como un esquema de modelos, además, el término todavía es utilizado para describir una vista transparente de un modelo construido con base en sus bordes.

La idea de definir la superficie del objeto en base a sus bordes es todavía eso, una idea, sin embargo, la estructura de datos (la organización interna de datos en el computador) deben ser adquiridos con otra descripción, de la información de cómo los bordes son conectados para formar las superficies de los elementos, lo que lleva a un esquema de representación de límites.

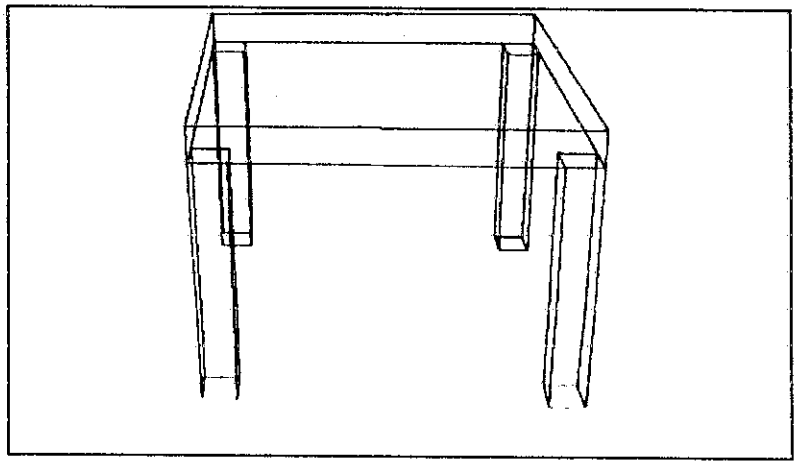


**Figura 2.1**: este modelo wire frame de una mesa aparece algo confuso en los bordes que podría ser invisible como se puede ver cuando se construye por medio de conexiones. Si el modelo es construido por medio de polígonos, un procedimiento puede ser usado para esconder las líneas que no se desean mostrar.

## 2.2 MÉTODOS DE MODELACIÓN DE OBJETOS

Este esquema se refiere a técnicas de modelación que construyen objetos en 3-D con base en la descripción de la superficie, más que del volumen que contiene. Como un simple ejemplo, el cubo que se describió antes para una superficie que usa las coordenadas de los vértices para definir los bordes, el cual crea los polígonos. Esto es un método de modelación de objetos, que es una geometría explícita para identificar cualquier punto sobre la superficie del cubo. Sin embargo, lo que no se tiene son datos similares para identificar puntos que están dentro del cubo y aquellos puntos que están fuera.

### 2.2.1 POLÍGONOS PLANOS

Un polígono define una clase de figura construida basado en una cadena de bordes. Esto incluye formas cuyos límites son convexos, cóncavos o se intersectan, y pueden o no ser planos. Con este amplio rango de construcciones, muchos modelos poligonales pueden ser realizados con polígonos planos, y que cada uno puede ser convexo o cóncavo, además, los polígonos cóncavos tienden a ser más complicados. Por ejemplo, con un polígono convexo cualquier punto contenido dentro de los límites sólo tiene que ser intersectado con los límites una vez que se salga del polígono, donde los límites cóncavos pueden crear cualquier número de intersecciones impares o irregulares. La Figura 2.2 ilustra este efecto. Además, cualquiera de los dos bordes adyacentes en un polígono convexo pueden ser usados como vectores para obtener un vector ortogonal consistente para la superficie, donde un polígono cóncavo podría revertir la dirección del vector dependiendo del ángulo encerrado entre los bordes.

Construir objetos usando polígonos es relativamente fácil, si sólo se necesita uno para identificar bordes importantes y construir la familia de facetas que cubrirán la superficie. Pero en la Animación por Computadoras, se debe estar consciente de cómo el objeto será animado cuando sea modelado. Para describir este último punto habrá que construir un cubo. La Figura 2.3 muestra su posición relativa para algún sistema de coordenadas, y por simplicidad hay que asumir que cada lado es de una longitud uno, sin importar su unidad de medida.

El cubo pudo haber sido construio en una variedad de formas y uno de los métodos podría almacenar las coordenadas de los ocho vértices en una tabla como se muestra en la Figura 2.3. Otra tabla puede referenciar dichas entradas de los vértices para formar los seis polígonos. Entonces, un programa puede construir cualquier polígono sacando los cuatro vértices referenciados en la tabla de polígonos.

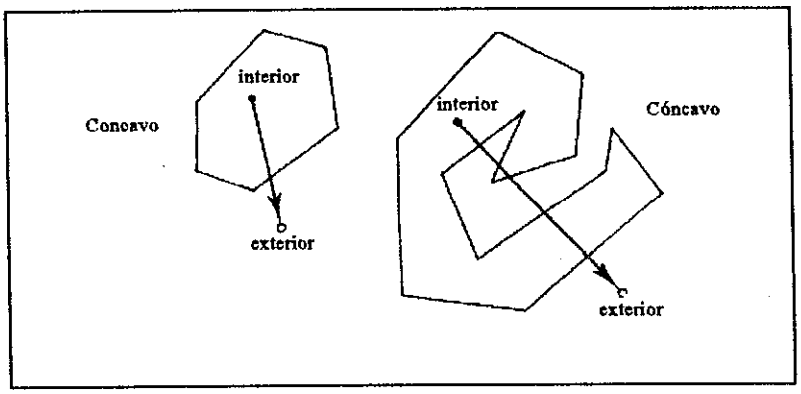
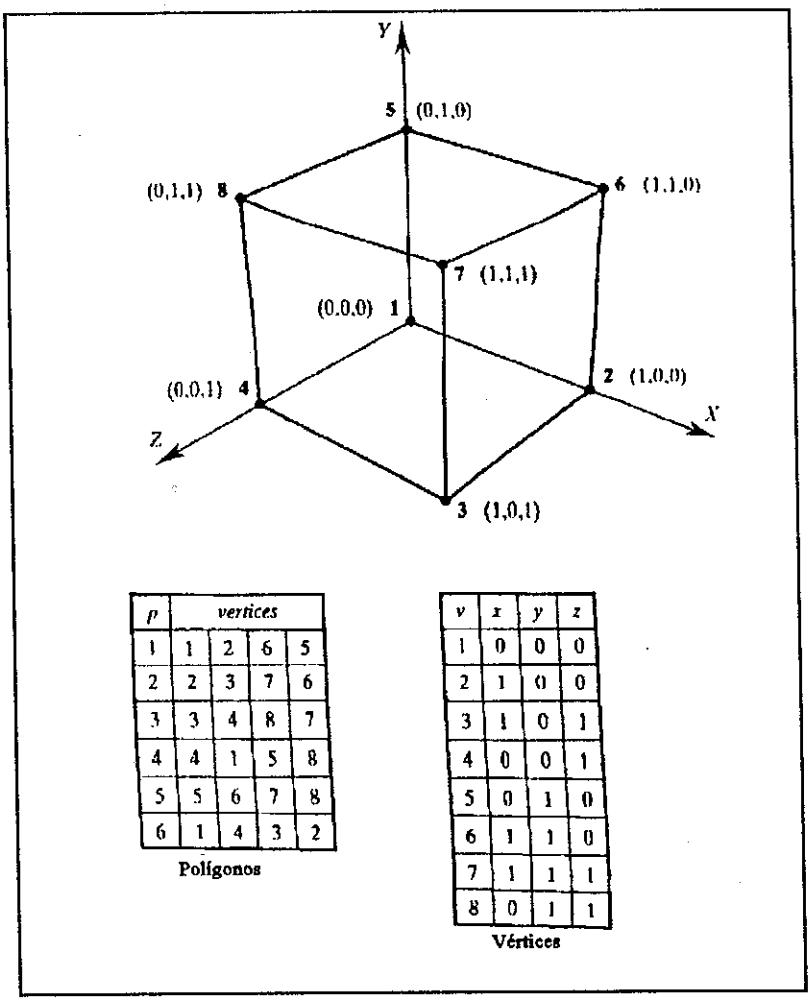



**Figura 2.2:** cualquier punto dentro del polígono convexo solo tiene que cruzar los límites una vez para llegar al exterior, donde un polígono cóncavo podría tener un número de cruces que depende del grado de concavidad. Esto es realizado por algunos algoritmos para gráficas.

**Figura 2.3:** si a este cubo se le asignan coordenadas como se muestra, las tablas son creadas para relacionar las seis superficies con los ocho vértices. Las dos tablas muestran como seis polígonos están formando un cubo que está almacenado como una tabla de entradas de vértices. Nótese que cada grupo de vértices está en una secuencia en el mismo sentido de las agujas del reloj como se ve desde el exterior del cubo.




| p | vertices | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 6 | 5 |
| 2 | 2 | 3 | 7 | 6 |
| 3 | 3 | 4 | 8 | 7 |
| 4 | 4 | 1 | 5 | 8 |
| 5 | 5 | 6 | 7 | 8 |
| 6 | 1 | 4 | 3 | 2 |

Polígonos

| v | x | y | z |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 1 | 0 | 1 |
| 4 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 | 0 | 1 | 1 |

Vértices

Una característica de la animación es que pueda tratar de aplanar el cubo en el vértice 7 con las coordenadas (1,1,1). Este vértice podría descender lentamente, a (1,0.5,1) y podría ser afectado simplemente alterando la entrada-y para el vértice 7 de 1 a 0.5 sobre un número específico de cuadros. El programa representará, la imagen actual con color, el que hará referencia a las dos tablas y construir los polígonos del cubo para cada cuadro de la animación, y si la geometría es alterada por el animador, entonces la imagen final reflejará estos cambios. Sin embargo, no se puede cambiar la geometría arbitrariamente y esperar que sea capaz de interpretar dichos cambios, especialmente si los cálculos asumen que son polígonos planos.

Si la coordenada-y del vértice 7 es reducido, el polígono superior del cubo no será plano, sino estará torcido. Además, si realmente se desea que la parte superior del cubo se aplane en el vértice 7, entonces los vértices 8 y 6 deberán ser disminuidos para que la parte superior sea plana.

En realidad, la forma en que se altera la geometría del cubo realmente depende de su composición. Hay que imaginarse los cambios que podrían ocurrir en la construcción de un cubo de esponja a uno hecho de arcilla. También construir un cubo con paneles de aluminio reaccionaría totalmente diferente a uno hecho de papel. Ahora tal vez se puede empezar a notar cómo el modelar y la animación están técnicamente juntos.

Se puede ver que la animación requiere ser anticipada a la etapa de modelación, y se debe interpretar la geometría para realizar este trabajo, parte de lo cual implica tener acceso a los bordes que forman los límites de los polígonos. En tales casos, los objetos son definidos en términos de polígonos jerárquicos construidos desde los bordes desde los cuales se construirán los vértices, y serán representados por tres listas de datos: de polígonos, bordes y vértices.

## 2.2.2 EXTRUSION

Aunque la superficie poligonal provee un método conveniente para construir objetos, se puede volver tedioso cuando se crean modelos complejos. Para eliminar lo tedioso, existen varias formas para ayudar a construir formas genéricas. La primera técnica es la de extrusión. Esta automáticamente construye una representación por medio de ampliar una sección transversal de la forma a lo largo de un eje para crear una extrusion de una longitud dada. La Figura 2.4 ilustra una sección transversal y una superficie 3-D creada, cuando es desplazada en una dirección-z.

La tarea de construir una superficie puede ser lograda automáticamente por un programa de computadora; todo lo que se requiere es que los datos de las coordenadas describan la sección transversal para tener dentro una estructura de datos conveniente y la longitud de la extrusión. No existe suficiente información geométrica para localizar cada vértice sobre la superficie de extrusión. En este proceso de construcción, el programa puede utilizar una convención de polígonos construidos a base de vértices en el sentido de las agujas del reloj.

Otro desarrollo de este método, es construir un conjunto de polígonos en distintas etapas de la extrusión, y en cada etapa la sección transversal es rotada. Esto permite dar vueltas a las formas que van a ser modeladas. La Figura 2.5 ilustra un ejemplo de este proceso.

Tal vez aún el tamaño de la sección transversal podría cambiar durante este proceso, o que la sección transversal se construya en otra forma. Idealmente, un programa de extrusión podría permitir al usuario especificar las secciones transversales inicial y final, y un ángulo de rotación aplicado a la sección transversal a cada etapa de la extrusión. Además, en vez de una distancia de extrusión, el procedimiento podría permitir un contorno 3-D para formar la línea central del objeto final.

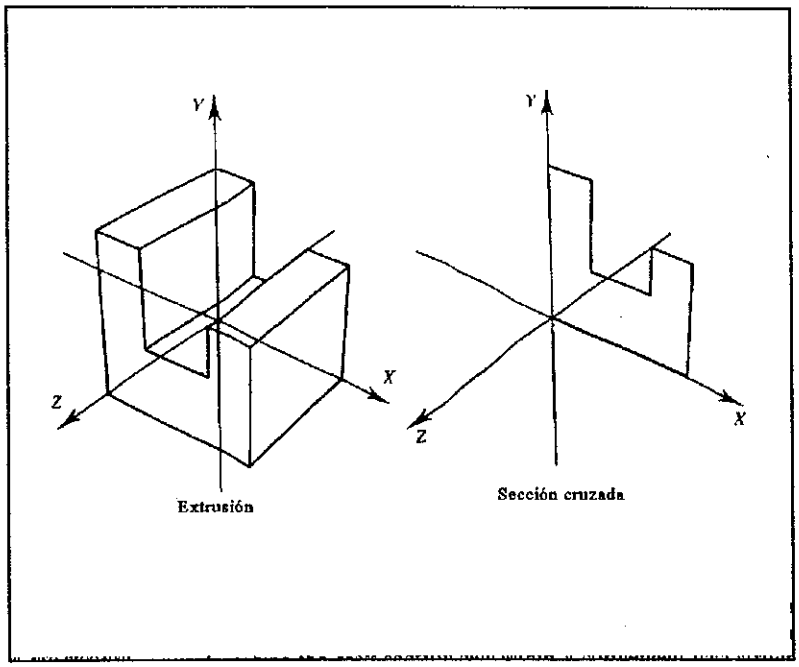



**Figura 2.4**: la sección transversal es usada para construir el volumen de extrusión 3-D, y es en el sentido de las agujas del reloj, el programa de modelación probablemente construirá todos los polígonos 3-D en el mismo sentido. Algunos sistemas de modelación proporcionan facilidades para revertir estas secuencias de los vértices.

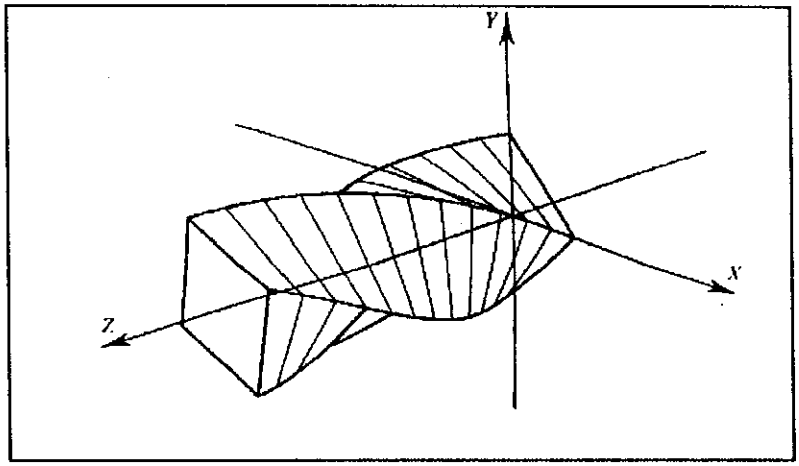



**Figura 2.5**: esta extrusión enrrollada es lograda al rotar una sección transversal referenciada a través de un ángulo pequeño, y cada vez es movido en la dirección-z.

## 2.2.3 SUPERFICIES DE REVOLUCIÓN

Las superficies de revolución ayudan a construir objetos como vasos, esferas, botellas, pimienta y ciertos pedazos de queso. Estos objetos exhiben una simetría alrededor de un eje central. Una vez más es necesario un contorno 2-D, sin embargo, esta vez es rotado alrededor de ejes en 3-D para ampliar una superficie de revolución. La Figura 2.6 muestra cómo se puede formar una copa de vino con base en un contorno inicial de 2-D. Nótese que la superficie final está todavía construida de polígonos planos, y el número actual depende del número de vértices en el contorno original, y el tamaño del ángulo se amplía en cada etapa de la construcción.

Tal como un programa de extrusión, las superficies de revolución pueden ser construidas automáticamente; todo lo que el usuario necesita es especificar la forma del contorno, el número de incrementos que va a hacer durante la rotación de 360°, y los ejes sobre los cuales se realizará la rotación. Un programa sólo

producirá objetos simétricos; sin embargo, con una pequeña modificación, puede ser usado para crear superficies asimétricas. Para empezar, sólo hay que imaginar la superficie final si; mientras se rota el contorno alrededor de un eje determinado, el eje mismo se va moviendo a lo largo de algún espacio. Similarmente, se pueden imaginar las formas distorsionadas que pueden ser creadas al modificar el contorno principal de 2-D mientras está en rotación.

## 2.2.4 SUPERFICIES CON GEOMETRÍA NO DEFINIDA

En la actualidad, existen objetos que no están hechos a base de polígonos. Tal vez las paredes de una habitación y el piso y cielo son en su totalidad polígonos, la forma de un teléfono, la manecilla de la puerta y la forma de una flor, son superficies complejas que se resisten a una solución matemática análiticа. En gráficas por computadoras, dichas formas son clasificadas como superficies free-form.

Esta filosofía del diseño detrás del modelo con superficies cortadas está basada en el hecho de que será difícil, si no imposible, identificar las ecuaciones que describan la geometría de la superficie de una manzana o una tetera. Sin embargo, el problema se simplifica a que si la superficie se divide en pequeñas áreas que individualmente tienen una simple solución geométrica. Este corte de superficie simplifica la complejidad de la totalidad de la superficie, pero aparece otro problema concerniente a las uniones de los cortes. Para cualquier técnica que sea usada, las uniones tienen que ser invisibles para el ojo; de otra forma no existe ninguna razón para usar los cortes.

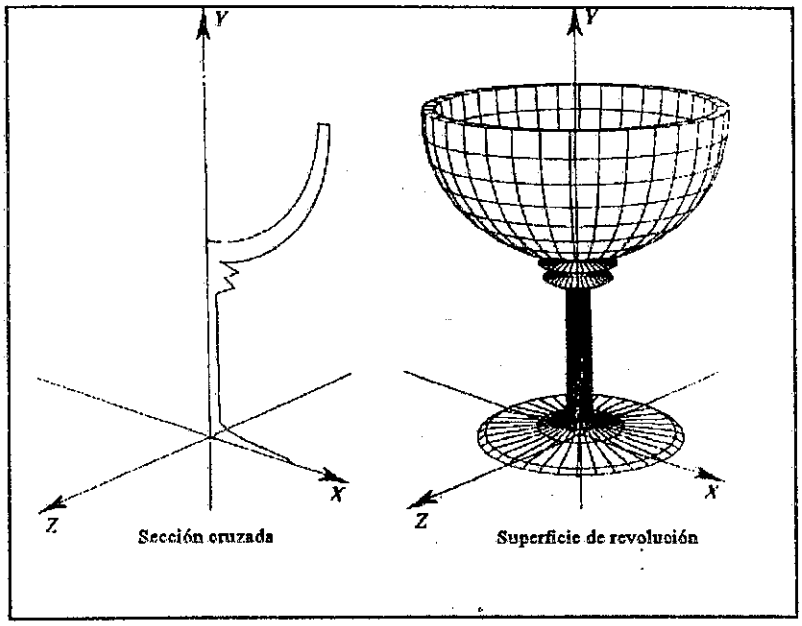



**Figura 2.6:** si la sección transversal es rotada alrededor del eje-y en incrementos angulares pequeños, se puede construir una superficie poligonal, la cual eventualmente forma una superficie continua de revolución.

No siempre se necesita revisar visualmente donde una superfice está marcada con una unión, solamente ampliarlo rápidamente para que revele cualquier pequeña discontinuidad en la geometría. Lo que se detecta son cambios en las inclinaciones de la superficie. Además, si los cortes están posicionados uno contra otro, la continuidad de la inclinación se mantendrá hasta estos límites.

## 2.2.5 SUPERFICIES BILINEALES

Antes de estudiar los cortes de las superficies Bézier, hay que considerar primero la idea de mezclar coordenadas usando interpolación lineal. Si se tienen cuatro puntos en el espacio 3-D como se muestra en la Figura 2.7, y la tarea es derivar una función lineal, la cual identifique cualquier punto contenido dentro de los límites del corte. Primero es construido un parámetro de espacio; también se muestra en la figura, el cual relaciona dos parámetros *u* y *w* con los puntos

del corte. Ahora una función lineal la cual mezcla los puntos $P_{11}$ y $P_{21}$ que es expresado como:

$$P_a(u) = (1 - u)P_{11} + uP_{21},$$

donde $P_a(u)$ es cualquier punto sobre la línea $\{P_{11},P_{21}\}$
Una función similar para los puntos $P_{12}$ y $P_{22}$ es:

$$P_b(u) = (1 - u)P_{12} + uP_{22},$$

donde $P_b(u)$ es cualquier punto sobre la línea $\{P_{12},P_{22}\}$. Los puntos pueden ser mezclados usando el segundo parámetro w y la expresión será:

$$P(u,w) = (1 - w)P_a(u) + wP_b(u)$$

la cual se expande a:

$$P(u,w) = (1-u)(1-w)P_{11} + u(1-w)P_{21} + (1-u)wP_{12} + uwP_{22}$$

cuya matriz es:

$$P(u,w) = [(1 - u)u]\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}\begin{bmatrix} (1 - w) \\ w \end{bmatrix}$$

Igualmente esto puede ser rearreglado para:

$$P(u,w) = [\ u\ \ 1\ ]\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} w \\ 1 \end{bmatrix}$$

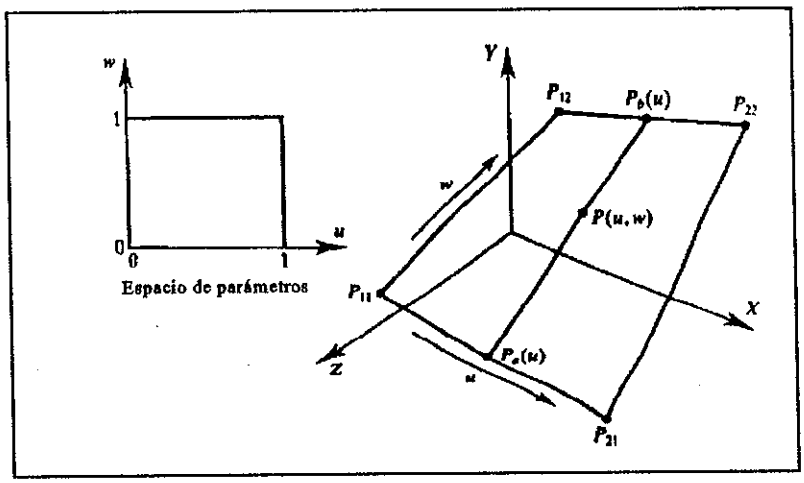



**Figura 2.7**: el diagrama muestra que los parámetros u y w varían entre 0 y 1 y están asociados con la interpolación bilineal de los puntos $P_{11}$, $P_{21}$, $P_{12}$ y $P_{22}$. $P_a(u)$ y $P_b(u)$ son puntos sobre dos orillas opuestas, y $P(u,w)$ es una interpolación bilineal entre ellos.

El algoritmo 2.1, llamado mip_map, regresa un color del mip-map dados $u,v$ y $d$. Usando como referencia una figura piramidal.

**Algoritmo 2.1**

```
float piso();                    /* regresa un entero */
float clamp();

void mip_map(map,u,v,d,color)
char map[1024][1024];            /* mip-map */
float u,v                        /* coordenadas en el rango [0,1] */
```

```
float d;                        /* compresión en el rango [0,1] */
float color[3];          /* valores de rvb (rojo,verde,azúl)*/

{
      int id;
      float color_claro[3],color_obscuro[3];
      int nivelsimple;  /* booleano que se mantiene en el tope
                           o en la base de la pirámide*/
      int mapsize;            /* longitud del nivel en los pixels
                                 de textura */
      int iu,iv,ui,vi,iu_n,iv_n,ui_n,vi_n;
                              /* offsets en el mip-map */
      float inf,sup,ru,rv,dinterp /*parámetros de interpolación*/

      /*Encuentra entre que capas se localiza la pirámide d*/

      d = clamp(d,0.,1.);
      id = piso(d*512);
      nivelsimple = ((id==0)|| (id==512));
      fo (mapsize = 512;id > 1;id >>= 1,mapsize >>= 1);
            iu = piso(u*mapsize);
      ru = u*mapsize - iu;
      iu_n = iu + 1;
      iu = iu % mapsize;       /*Cubre alrededor de la orilla de
                                   la textura*/
      iu_n = iu_n % mapsize;

      iu = iu + mapsize;
      iu_n = iu_n + mapsize;

      iv = piso(v * mapsize);
      rv = v*mapsize - iv;
      iv_n = iv + 1;
      iv = iv % mapsize;
      iv_n = iv_n% mapsize;

      vi = vi + mapsize;
      vi_n = iv_n + mapsize;

      /*Se realiza la interpolación bilinear de los componentes
        en forma separada en el nivel más inferior en el árbol
        de colores*/

      inf=map[ui][iv]+ru*(map[ui_n][iv]-map[ui][iv]);
      sup=map[ui][iv_n]+ru*(map[ui_n][iv_n]-map[ui][iv_n]);
      color_claro[0] = inf+rv*(sup-inf);

      inf=map[ui][vi]+ru*(map[ui_n][vi]-map[ui][vi]);
      sup=map[ui][vi_n]+ru*(map[ui_n][vi_n]-map[ui][vi_n]);
      color_claro[1] = inf+rv*(sup-inf);


      inf=map[iu][vi]+ru*(map[iu_n][vi]-map[iu][vi]);
      sup=map[iu][vi_n]+ru*(map[iu_n][vi_n]-map[iu][vi_n]);
      color_claro[2] = inf+rv*(sup-inf);

      /*Si el nivelsimple los mantiene, se está en la parte
        inferior o en la superior de la pirámide´, y se necesi-
        tará sólo una interpolación bilinear en un nivel.
        De otra forma, la realización de una interplación
        bilinear en el siguiente nivel superior, el cual es
```

```
        la mitad de la dimensión lineal del nivel previo*/

      if (nivelsimple) {
            color[0] = color_claro[0];
            color[1] = color_claro[1];
            color[2] = color_claro[2];
      }
      else {
            mapsize >>= 1;
            ru = (ru + (iu % 2))/2;
            iu >>= 1;
            iu_n = iu + 1;
            iu = iu % mapsize;
            iu_n = iu_n + mapsize;
            rv = (rv + (iv % 2))/2;
            iv >>= 1;
            iv_n = iv + 1;
            iv = iv % mapsize;
            iv_n = iv_n % mapsize;
            iu = iu + mapsize;
            iu_n = iu_n % mapsize;

      inf=map[ui][iv]+ru*(map[ui_n][iv]-map[ui][iv]);
      sup=map[ui][iv_n]+ru*(map[ui_n][iv_n]-map[ui][iv_n]);
      color_obscuro[0] = inf+rv*(sup-inf);

      inf=map[ui][vi]+ru*(map[ui_n][vi]-map[ui][vi]);
      sup=map[ui][vi_n]+ru*(map[ui_n][vi_n]-map[ui][vi_n]);
      color_obscuro[1] = inf+rv*(sup-inf);

      inf=map[iu][vi]+ru*(map[iu_n][vi]-map[iu][vi]);
      sup=map[iu][vi_n]+ru*(map[iu_n][vi_n]-map[iu][vi_n]);
      color_obscuro[2] = inf+rv*(sup-inf);

      /*Interpolación lineal entre los niveles por una
        cantidad dinterp*/

      dinterp = mapsize*d - 1;
      color[0] = color_claro[0]+dinterp*(color_obscuro[0]-
                 color_claro[0]);
      color[1] = color_claro[1]+dinterp*(color_obscuro[1]-
                 color_claro[1]);

      color[2] = color_claro[2]+dinterp*(color_obscuro[2]-
                 color_claro[2]);

      }

}
```

Para propósitos más claros, el algoritmo anterior presenta la indexación para el mip-map de los componentes de colores codificados. De cualquier forma, se puede tomar ventaja de los cálculos hechos en el algoritmo anterior, de la organización de la memoria, ya que todo se hace a la potencia de 2, la indexación puede ser hecha usando una escala binaria.

## 2.2.6 SUPERFICIES RAYADAS

Una superfice rayada se desarrolla con base en una familia de líneas angostas. Por ejemplo, si se toman dos barras y se trabajan aparte en el espacio, se pueden encajar juntas con cordones, la cual formará una superfice rayada. Ejemplos similares de superficies rayadas son encontrados en cilindros y conos.

Aquí lo importante es que los puntos sobre las dos formas están conectados con líneas.

Un método conveniente para crear dichas superficies es usar una aproximación paramétrica donde dos curvas paramétricas están linealmente interpoladas. Como un ejemplo, hay que considerar las expresiones paramétricas para desarrollar una curva cuadrática de Bézier. Si la función $B(1,w)$ calcula una curva 3-D de Bézier de los puntos de control $P_{11}$, $P_{12}$ y $P_{13}$, y la función $B(2,w)$ calcula una curva similar de Bézier de los puntos de control $P_{21}$, $P_{22}$ y $P_{23}$, entonces la superfice rayada puede estar definida como:

$$P(u,w) = (1 - u)B(1,w) + B(2,w)$$

donde $0 <= u <= 1$ y $0 <= w <= 1$

Si se seleccionan valores para $w$, se pueden calcular los puntos correspondientes sobre las dos curvas de Bézier. Esto podrá ser entonces usado para identificar puntos a lo largo de la línea de conexión por medio de la seleccion de valores para $u$, como se muestra en la Figura 2.8.

## 2.2.7 CORTES DE SUPERFICIES BEZIER

Para construir una superficie cuadrática de Bézier, son necesarios nueve puntos de control en la forma de una matriz de 3 x 3. Cualquier punto asociado sobre la superficie de Bézier puede ser calculado si se especifican los valores de dos parámetros que varían sobre el rango de 0 a 1. Si estos parámetros son llamados $u$ y $w$, y los puntos de control referenciados, como se muestra en la Figura 2.9, entonces cualquier punto $P(u,w)$ puede ser localizado evaluando los componentes $x$, $y$ e $z$ de los puntos de control tal que:

$$P(u,w) = [(1 - u)^2 2u(1 - u) u^2] \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} (1 - w)^2 \\ 2w(1 - w) \\ w^2 \end{bmatrix}$$

o puede ser arreglado como:

$$P(u,w) = [u^2 \; u \; 1] \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} w^2 \\ w \\ 1 \end{bmatrix}$$

La tabla 2.1 ilustra cómo se calcula $P(u,w)$ para ciertos valores de $u$ y $w$, donde se ve que cualquier punto sobre la superficie cuadrática de Bézier es creado al sumar los puntos de control por términos generados por las matrices anteriores.

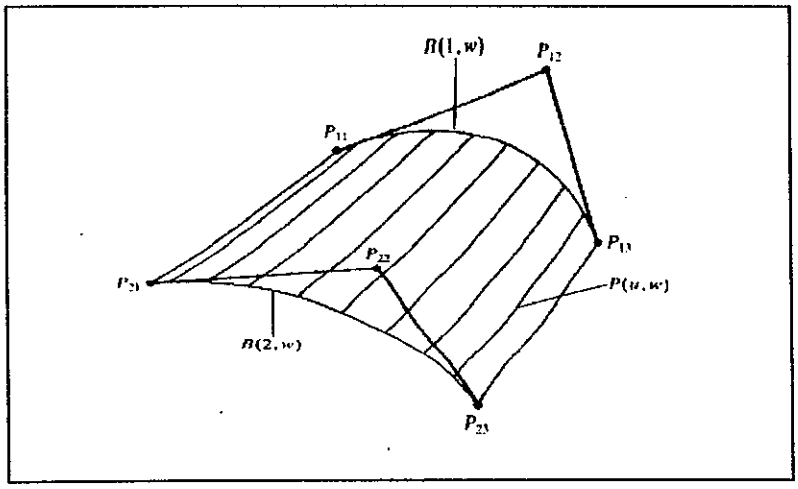



**Figura 2.8**: esta superfice está formada de dos curvas cuadráticas de Bézier. Si *B(1,w)* y *B(2,w)* son las dos funciones de Bézier, entonces podrá ser interpolada linealmente para crear cualquier punto *P(u,w)*.

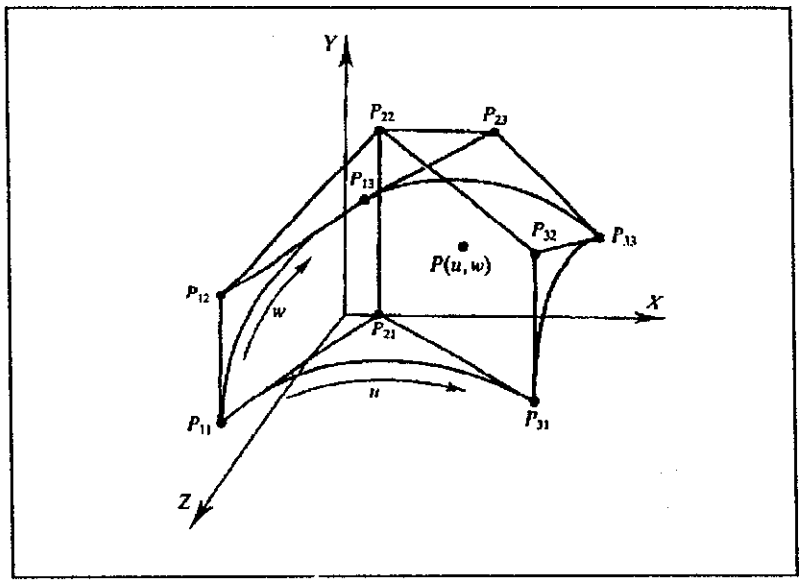



**Figura 2.9**: este diagrama ilustra los nueve puntos de control usados para desarrollar cortes de superficies cuadráticas de Bézier, los cuales tocan los puntos de las cuatro esquinas $P_{11}$, $P_{31}$, $P_{13}$ y $P_{33}$. Cualquier punto $P(u,w)$, donde los parámetros $u$ y $w$ varian entre 0 y 1, pueden ser derivados evaluando las operaciones descritas en la matriz.

Si en lugar de los términos cuadráticos de una relación cuadrática, son cúbicos, los términos serán:

$$(1 - u)^3, (1 - 2)^2 3u, (1 - u) 3u^2, u^3,$$

los cuales pueden ser usados para desarrollar una superficie cúbica de Bézier. Para esto es necesario una matríz de 4 x 4 de puntos de control, y cualquier punto *P(u,w)* puede ser encontrado al evaluar las coordenadas *x*, *y* e *z* de los puntos de control tal que:

$$P(u,w)=[(1-u)^3 3u(1-u)^2 3u^2(1-u) u^3] \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{43} \end{bmatrix} \begin{bmatrix} (1 - w)^3 \\ 3w(1 - w)^2 \\ 3w^2(1 - w) \\ w^3 \end{bmatrix}$$

lo anterior puede ser arreglado como:

$$P(u,w)=[u^3\ u^2\ u\ 1]\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}P_{11} & P_{12} & P_{13} & P_{14}\\ P_{21} & P_{22} & P_{23} & P_{24}\\ P_{31} & P_{32} & P_{33} & P_{34}\\ P_{41} & P_{42} & P_{43} & P_{43}\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}w^3\\ w^2\\ w\\ 1\end{bmatrix}$$

**Tabla 2.1**

| u | w | $P_{11}$ | $P_{21}$ | $P_{31}$ | $P_{12}$ | $P_{22}$ | $P_{32}$ | $P_{13}$ | $P_{23}$ | $P_{33}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 0.0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.0 | 0.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1.0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0.5 | 0.0 | 1/4 | 1/2 | 1/4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.0 | 0.5 | 1/4 | 0 | 0 | 1/2 | 0 | 0 | 1/4 | 0 | 0 | 0 |
| 1.0 | 050 | 0 | 0 | 1/4 | 0 | 0 | 1/2 | 0 | 0 | 1/4 | |
| 0.5 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/4 | 1/2 | 1/4 | |
| 0.5 | 050 | 1/16 | 1/8 | 1/16 | 1/8 | 1/4 | 1/8 | 1/16 | 1/8 | 1/16 | |

Así con solo nueve puntos de control para una unión cuadrática, y 16 para una unión cúbica, se puede definir un corte contínuo de espacio, cuya geometría puede ser modificada al ajustar uno o más puntos de control.

Además no es muy difícil imaginar una superficie ondulada, midiendo su inclinación en tres dimensiones; es mejor pensar que son más útiles las curvas 2-D de Bézier para ver cómo las características de la inclinación pueden ser definidas.

Hay que considerar, entonces, las curvas cuadráticas de Bézier ilustradas en la Figura 2.10. Se puede ver que la inclinación de la curva $P_1$ es igual a la inclinación del primer arco $P_1$ a $P_2$, la inclinación de la curva $P_3$ es igual a la inclinación del último arco $P_2$ a $P_3$. Además, para mantener la continuidad de la inclinación, a través de los dos segmentos de curva de Bézier, hay que asegurarse de que la inclinación del punto de control del arco encaja con el punto de control del arco principal de la segunda curva. Similarmente, se puede demostrar que la continuidad en tres dimensiones a través del corte de superficie puede mantenerse si:

(1) los puntos de control de los límites entre los dos cortes son idénticos, y
(2) los polígonos formados por los puntos de control en las orillas son coplanares para ambos cortes.

Estas condiciones están mostradas en la Figura 2.11 para los dos cortes de superficie bicúbicos de Bézier.

El algoritmo que crea los cortes de superficie de Bézier que utiliza el vector normal se muestra en el algoritmo 2.2, llamado *crear_cortes_normales()*.

**Algoritmo 2.2**

```
int   crear_cortes_normales(hull,normals)
float hull[4][4][3],normals[2][2][3];
{
      float utangente[3],vtangente[3];
      int i,j;
      int vector_cero();

      j=1;
      do {
            if (j==4) return(0);
```

```
      for (i=0;i<3;i++) utangente[i] = hull[j][0][i]-
                                       hull[0][0][i];
      j++;
} while (vector_cero(utangente));
j=1;
do {
      if (j==4) return(0);
      for (i=0;i<3;i++) vtangente[i] = hull[0][j][i]-
                                       hull[0][0][i];
      j++;
} while (vector_cero(vtangente));
producto_cruz(utangente,vtangente,&normals[0][0][0]);
j=1;
do {
      if (j==4) return(0);
      for (i=0;i<3;i++) utangente[i] = hull[j][3][i]-
                                       hull[0][3][i];
      j++;
} while (vector_cero(utangente));
j=2;
do {
      if (j== -1) return(0);
      for (i=0;i<3;i++) vtangente[i] = hull[0][3][i]-
                                       hull[0][j][i];
      j--;
} while (vector_cero(vtangente));
producto_cruz(utangente,vtangente,&normals[0][1][0]);

j=2;
do {
      if (j== -1) return(0);
      for (i=0;i<3;i++) utangente[i] = hull[3][0][i]-
                                       hull[j][0][i];
      j--;
} while (vector_cero(utangente));
j=1;
do {
      if (j==4) return(0);
      for (i=0;i<3;i++) vtangente[i] = hull[3][j][i]-
                                       hull[3][0][i];
      j++;
} while (vector_cero(vtangente));
producto_cruz(utangente,vtangente,&normals[1][0][0]);

j=2;
do {
      if (j== -1) return(0);
      for (i=0;i<3;i++) vtangente[i] = hull[3][3][i]-
                                       hull[j][3][i];
      j--;
} while (vector_cero(vtangente));
j=2;
do {
      if (j== -1) return(0);
      for (i=0;i<3;i++) utangente[i] = hull[3][3][i]-
                                       hull[3][j][i];
      j--;
} while (vector_cero(utangente));
producto_cruz(utangente,vtangente,&normals[1][1][0]);

return(1);
```



}

}

El algoritmo 2.1 calcula los vectores tangentes y calcula el producto cruz para cada esquina de cada corte o parte.

## 2.2.8 SECCIONES DE SUPERFICIE DE CONOS BICUBÍCOS

Una sección de superficie de cono bicúbico utiliza funciones de mezcla para desarrollar superficies geométricas. Básicamente dados dos valores $N_1$ y $N_2$, y los correspondientes índices de cambio $S_1$ y $S_2$, se puede calcular un valor $N(t)$ de la siguiente forma:

$$N(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ S_1 \\ S_2 \end{bmatrix}$$

donde $0 <= t <= 1$.

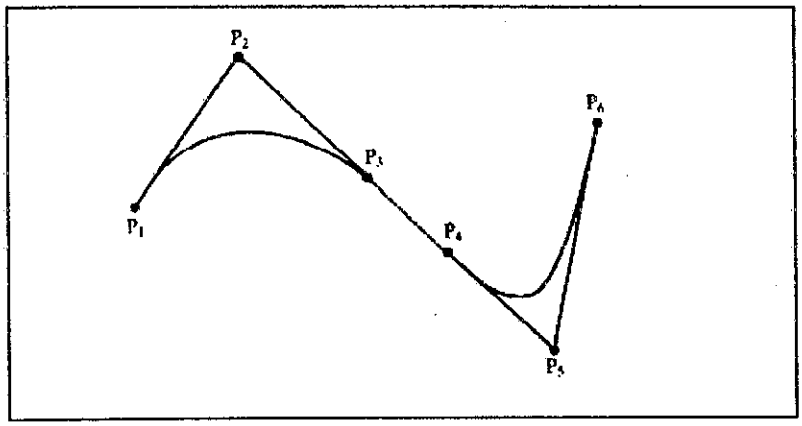



**Figura 2.10:** ee puede demostrar que la inclinación de la curva $P_1$ es igual a la inclinación de la línea $(P_1,P_2)$. Similarmente, en $P_3$ la inclinación es igual a la inclinación de la línea $(P_2,P_3)$. Para mantener la continuidad de la inclinación a través de los dos segmentos curvos, la inclinación $(P_2,P_3)$ deberá ser igual a la inclinación $(P_4,P_5)$.

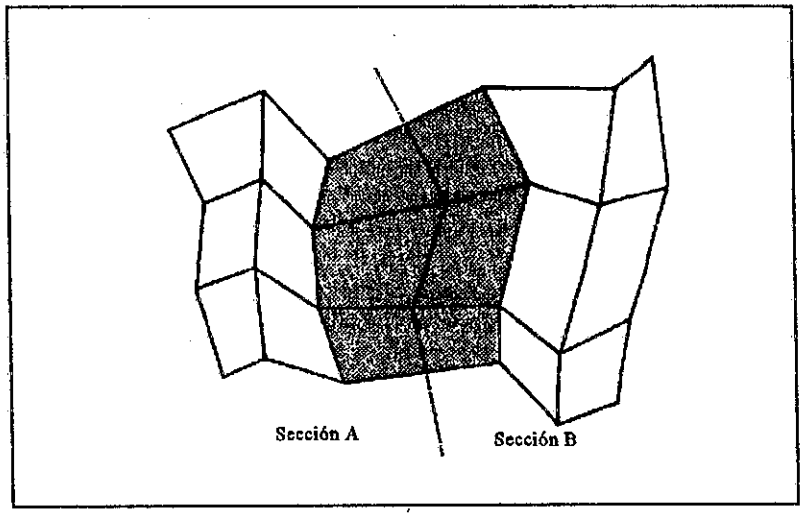



**Figura 2.11:** para garantizar que la continuidad de la inclinación se mantiene a través de los cortes *A* y *B*, los vértices límites deberán coincidir y los polígonos sombreados deberán ser coplanares. Para un primer orden, los dos polígonos centrales no necesariamente serán planos.

Sin embargo, esta técnica puede también ser desarrollada para calcular puntos sobre superficies 3-D de la siguiente forma:

$$Q(u,w) = [U]\ [C]\ [P]\ [C]^T\ [W]^T$$

(Nota: $[C]^T$ es la transpuesta de $[C]$, la cual cambia cada elemento $C_{fila,col}$ en $C_{col,fila}$).
donde

$$[U] = [u^3 \quad u^2 \quad u \quad 1]$$

$$[C] = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$[C]^T = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

$$[WU] = [w^3 \quad w^2 \quad w \quad 1]$$

$$[P] = \begin{bmatrix} P(0,0) & P(0,1) & P_w(0,0) & P_w(0,1) \\ P(1,0) & P(1,1) & P_w(1,0) & P_w(1,1) \\ P_u(0,0) & P_u(0,1) & P_{uw}(0,0) & P_{uw}(0,1) \\ P_u(1,0) & P_u(1,1) & P_{uw}(1,0) & P_{uw}(1,1) \end{bmatrix}$$

Los elementos de la matriz anterior están divididos de la siguiente manera: los cuatro términos en la esquina superior izquierda son vectores de posición que almacenan las coordenadas de los cuatro vértices de la sección de la superficie. Los cuatro términos en la esquina superior derecha son las inclinaciones de las esquinas de la sección respecto al parámetro *w*, y los cuatro términos en la parte inferior izquierda son las inclinaciones de las esquinas de la sección con respecto al parámetro *u*. Los cuatro términos en la esquina inferior derecha son llamados los vectores de enlace, los cuales controlan la propagación de las inclinaciones de las esquinas en el resto de la sección. Efectivamente, los vectores de enlace obligan a la superficie a ser más convexa o cóncava, para preservar las tangentes de las esquinas.

Las secciones Bézier necesitan de las posiciones de los puntos de control para manipular la geometría de la superficie, pero las secciones del cono requieren que esto se logre sólo si se ajustan las tangentes en las cuatro esquinas, y si se seleccionan valores para los vectores de enlace.

## 2.2.9 SECCIONES DE SUPERFICIE B-SPLINE

En la sección 2.2.7, se vio cómo una matriz de puntos de control puede ser usada para definir secciones de superficie Bézier, y si uno de estos puntos es modificado, entonces toda la sección es distorsionada. La razón, para esta perturbación global radica en que el algoritmo de Bézier calcula las coordenadas de cualquier punto de la superficie al variar las proporciones de todos los puntos de control. Sin embargo, si el algoritmo se modifica para satisfacer cualquier tamaño de matriz de puntos de control, y solamente referencia los puntos que son necesarios para calcular la parte de la superficie asociada, se

origina el algoritmo B-spline. Una consecuencia de esta modificación es que mientras el algoritmo de Bézier utiliza un parámetro que varía entre 0 y 1, el algoritmo B-spline requiere una serie de valores de parámetros determinados por el número de puntos de control y el tipo de curva, ya sea cuadrática, cúbica o de cuarto grado. Estos valores paramétricos son llamados vector de grupo.

Los valores de este vector influyen en la naturaleza de la superficie, ya que son usados en el algoritmo para variar la influencia que tienen al final sobre un punto de control.

Otra utilidad del B-spline es que los valores del vector pueden ser repetidos; este tiene el efecto de enfatizar la existencia de un punto de control, el cual es útil para formar cúspides.

El B-spline puede ser representado como un radio de polinomios; éstos permiten construir arcos y círculos completos. La ventaja principal del B-spline es que cuando un punto de control es movido influye solamente en una parte de la curva. Esto es útil cuando se diseñan formas complejas.

## 2.3 REPRESENTACIÓN VOLUMÉTRICA

Las representaciones volumétricas son objetos que se construyen a partir de un conjunto de volúmenes simples en 3-D. Los dos esquemas más cercanos a una representación volumétrica son el de la geometría sólida constructiva (CSG Constructive Solid Geometry) y la subdivisión espacial.

### 2.3.1 GEOMETRÍA SÓLIDA CONSTRUCTIVA

Este tipo de representación construye un objeto a partir de un conjunto pequeño de formas en 3-D como una caja, esfera, cilindro, cono, hélice. Esto puede ser una tarea imposible, y en algunas aplicaciones sí lo es, pero si sería bastante difícil construir un rostro humano con esta técnica. Además, en general, solamente es usado para construir escenas u objetos que tienen forma geométrica inherente. Por ejemplo, ciertos componentes mecánicos tales el motor de un carro, la cabeza de un cilindro, o varias estructuras arquitectónicas se pueden categorizar como una representación CSG (geometría sólida constructiva, por sus siglas en inglés).

Uno de los inusuales usos de esta técnica es la habilidad de agregar y restar volúmenes, y también identificar un volumen de espacio que es compartido por dos intersecciones. Primero, se examinará la operación de unión la cual combina dos objetos.

Por ejemplo, se necesita construir el objeto que se muestra en la Figura 2.12a. Se puede decir que consiste de tres elementos A, B y C, pero si B y C son idénticos, entonces tal vez los objetos pueden ser imaginados para obtener dos bloques intersectados A y D, como se ilustra en la Figura 2.12b. Usando la notación CSG, el objeto final es construido de la unión de A y D, como se muestra en los tres diagramas de la Figura 2.13.

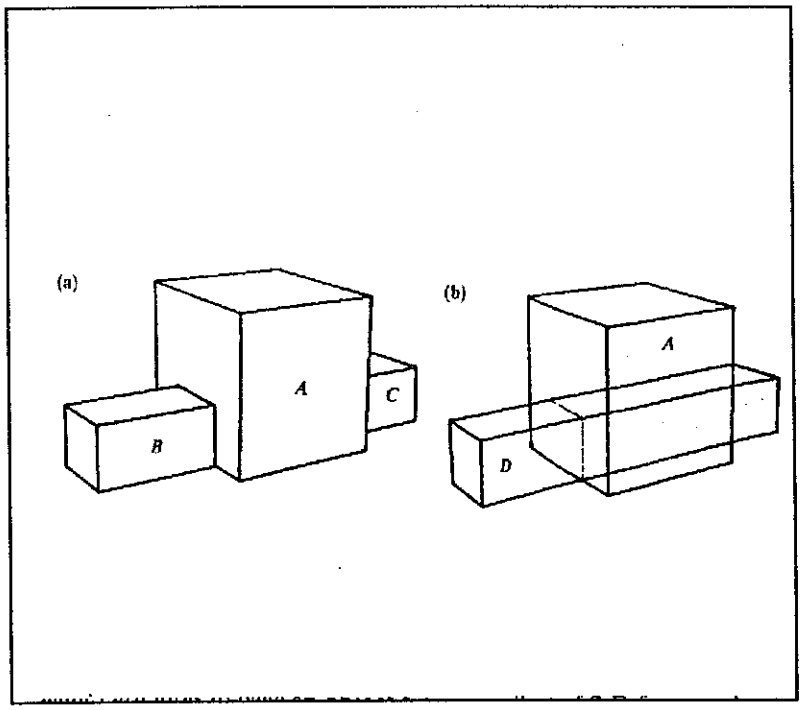



**Figura 2.12:** esta estructura podría ser construida a partir de tres bloques *A*, *B* y *C*. Pero si *B* tiene las mismas dimensiones como *C*, entonces los dos bloques *B* y *C*, pueden ser reemplazados por un bloque *D*. En la forma CSG, el montaje es definido por la unión de *A* y *D*.

Como en otro ejemplo, el objeto que se muestra en la Figura 2.14a, básicamente es un bloque que tiene un agujero. Esto puede ser definido usando la operación de diferencia la cual resta un volumen de otro, y es mostrada en la Figura 2.14b con el diagrama de árbol.

Existe una ecuación para una esfera, pero no existe una ecuación equivalente para una caja; sin embargo, es posible derivar una ecuación de otra. Hay que considerar la ecuación para un plano:

$$ax + by + cz + d = 0$$

i se eligen valores específicos para *a*, *b*, *c* y *d* como:

$$a=0, b=0, c=1 \text{ y } d=0,$$

entonces las ecuaciones cambian a:

$$0x + 0y + z = 0$$

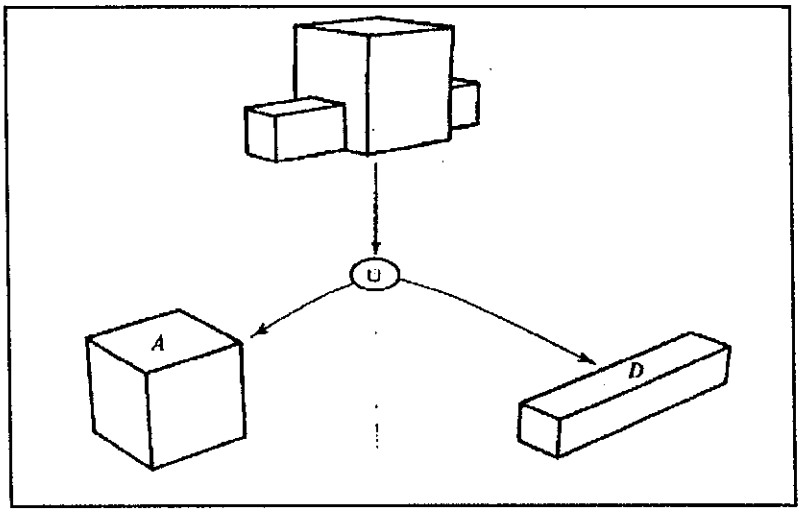



**Figura 2.13:** si se usa CSG, el objeto superior puede ser definido como la unión (U) de dos bloques *A* y *D*.

Si se eligen valores específicos para *a*, *b*, *c* y *d* como:

a=0,b=0,c=1 y d=0,

entonces las ecuaciones cambian a:

0x + 0y + z = 0

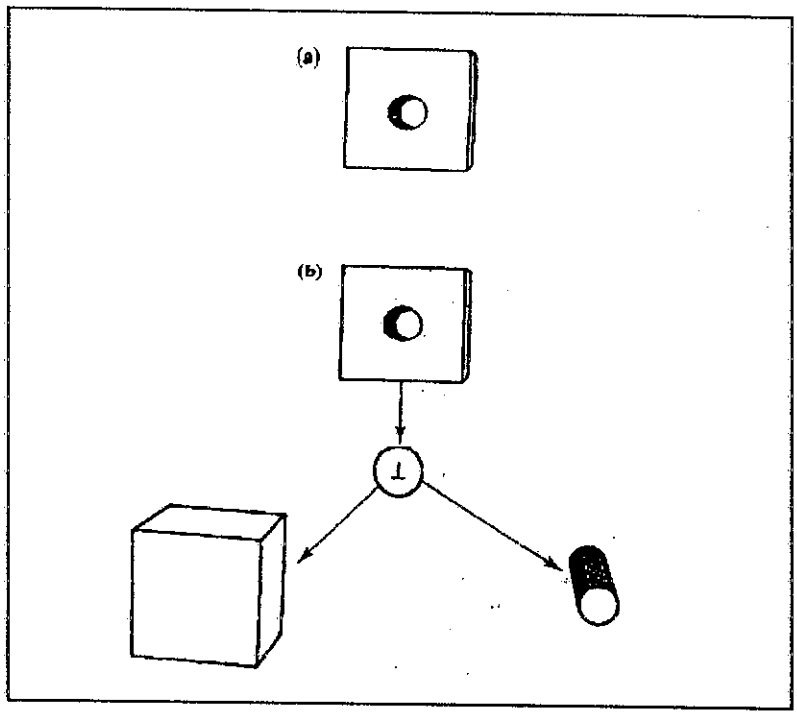



**Figura 2.14:** (a) una definición CSG para el bloque que tiene un agujero es un cilindro *B* restado del cuboide *A*, y es representado por la operación de diferencia. (b) La operación de diferencia es mostrada como una estructura de árbol Booleano.

Existe un número infinito de puntos que satisfacen esta ecuación, y no importa que los valores de *x* e *y* sean cero, menos el valor de *z*. Geométricamente, esta ecuación define una superficie plana coincidente con los ejes *x* e *y* donde *z* = 0, como se muestra en la Figura 2.15.

## 2.3.2 SUBDIVISIÓN ESPACIAL

Los algoritmos de este tipo de esquemas tratan de construir objetos de 3-D a base de volúmenes anidados o de un volumen de espacio dividido en voxels (es un volumen de espacio y representa el equivalente en 3-D de un pixel). Por ejemplo, el cubo ilustrado en la figura 2.16 puede ser dividido en ocho cubos pequeños que pueden ser divididos en otros ocho, sin ningún límite, tal que una división espacial puede ser representada por un diagrama de árbol, como se muestra en la misma figura, que identifica los volúmenes con números. El diagrama de árbol, que es conocido como octree (es una estrategia de particionamiento en el espacio 3-D donde un volumen es dividido recursivamente en una organización jerárquica de octantes), es usado para describir la geometría del objeto que registra aquellas divisiones de espacio que son usados por el objeto.

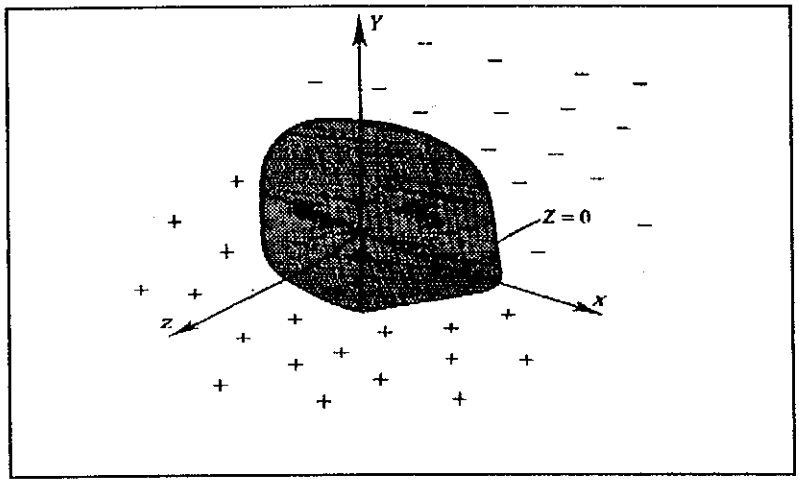



**Figura 2.15:** la superficie está definida por la ecuación $0x + 0y + z = 0$, la que se satisface cuando $z = 0$. El plano efectivamente divide el espacio en dos espacios.

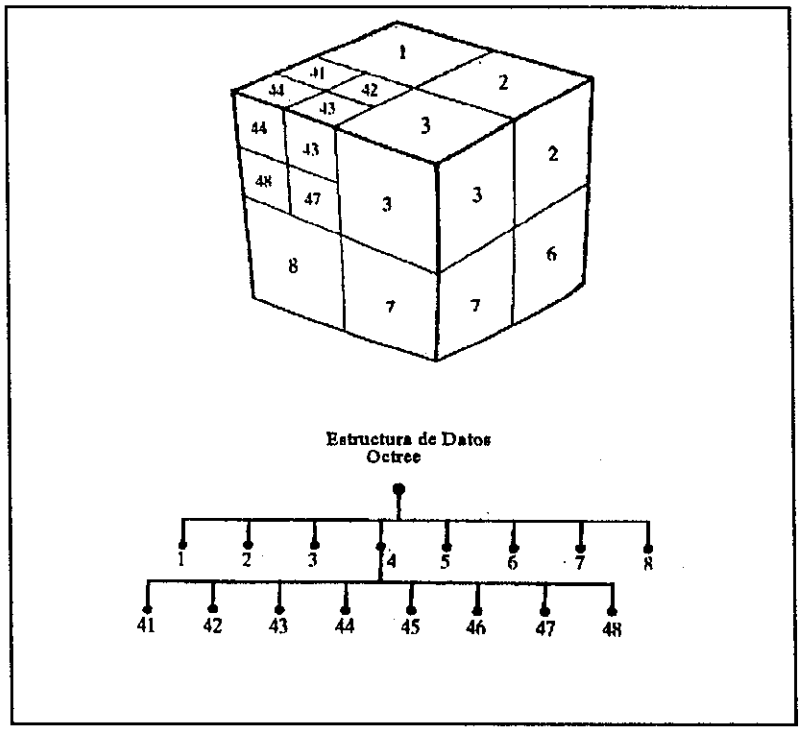



**Figura 2.16:** un cubo se puede particionar en ocho sub-cubos, los cuales pueden ser divididos en otros cubos pequeños. Este es el esquema de subdivisión octree para modelos en 3-D. La estructura de datos está representada por un árbol que registra el estado de cada cubo en términos de los elementos que estén vacíos o no.

La Figura 2.17 muestra como un simple bloque del objeto puede ser representado por un octree.

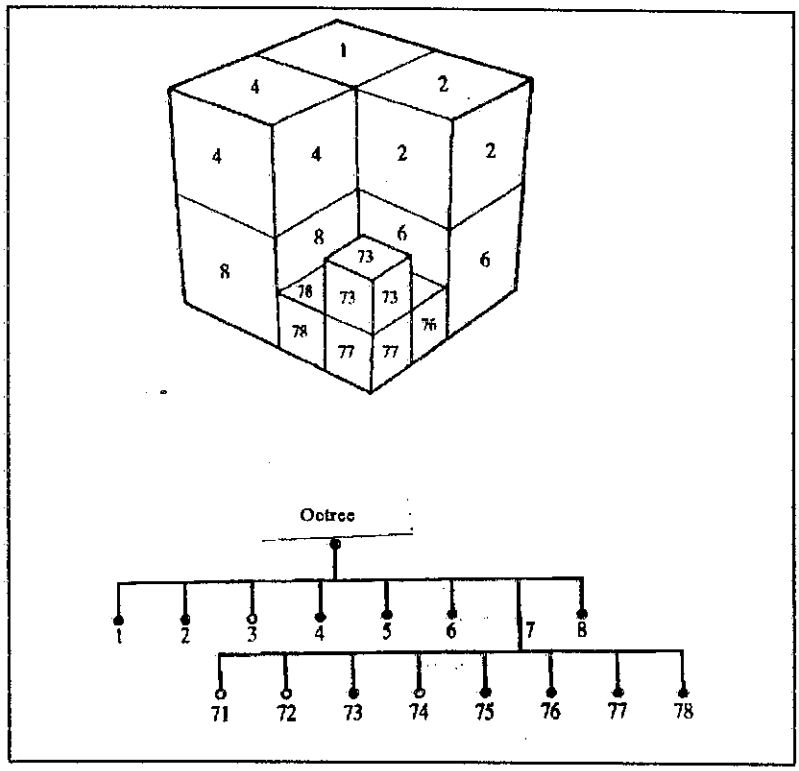



**Figura 2.17**: este objeto puede ser representado por un octree que define los volúmenes de elementos que están llenos o vacíos.

## 2.4 MODELO PROCEDURAL

El modelo procedural permite una clase de técnicas que involucran un procedimiento computacional para calcular la geometría de alguna forma. Por ejemplo, si se desea modelar una montaña será imposible describir cada cima, valle y acantilado. Además, es difícil posicionar cada piedra.

### 2.4.1 ESQUEMA FRACTAL

Este esquema trabaja con la dimensión, que resulta de la descripción geométrica de las cosas que existen en el universo y pueden ser descritas por medio de tres mediciones (coordenadas) para obtener puntos en el espacio. Dicho espacio es llamado Euclideano, si las leyes para medir distancias, áreas y volúmenes son lineales.

Otro concepto importante del esquema fractal es la similitud. Esta propiedad la tienen algunas formas, en las cuales, cuando son aumentadas, aparecen los mismos detalles que se encontraron en la forma original antes de ser aumentada. Un copo de nieve es un buen ejemplo: sus orillas arrugadas aparecen igual cuando el copo es aumentado de tamaño bajo un microscopio.

Otro ejemplo, el cubo. Este puede ser construido a base de ocho cubos similares pequeños. La figura 2.18 ilustra dicha relación. Lo que se necesita ahora es una fórmula para relacionar la dimensión (3) con el número de partes similares (8) y el factor de reducción (1/2), pero antes se revisarán los dos casos, el de una dimensión y el de 2-D.

La figura 2.18b muestra que un cuadrado es equivalente a cuatro cuadrados similares que pueden ser disminuidos en tamaño por un factor de 1/2. Una vez más, habrá que encontrar una relación entre la dimensión (2), el número de partes similares (4) y el factor (1/2).

Finalmente, un segmento de línea es similar a su equivalente a dos segmentos de línea que pueden ser reducidos por un factor de 1/2, como se muestra

en la Figura 2.18c. Así, se necesita encontrar la relación entre la dimensión (1), el número de partes similares (2) y el factor (1/2).
Esto se puede ver en la tabla 2.2.

**Tabla 2.2**

| Dimensión | Partes Similares | Factor |
|---|---|---|
| D | N | S |
| 1 | 2 | 1/2 |
| 2 | 4 | 1/2 |
| 3 | 8 | 1/2 |

La regla que une estos tres factores juntos es:

$$N = 1/S^D,$$

donde *N* es el número de partes similares, *S* es el factor de división y *D* es la dimensión. Arreglando ésta fórmula, nos proporciona una definición en términos de dimensión, tal que:

$$D = \log(N)/\log(1/S)$$

El valor fractal que se obtiene está relacionado con cualquier cosa que exista físicamente en el universo; es una construcción matemática y el término 'dimensión' no puede ser usada para reemplazar el significado original de dimensión el cual tiene una asociación topológica. De hecho, existe una clase de curvas llamadas space-filling (son curvas capaces de convertir completamente un área con una longitud infinita pero con área finita) que tienen una dimensión topológica de 1 y una dimensión fractal de 2.

No existe una función matemática que proporcione las coordenadas *x* e *y* del copo de nieve, tienen que ser obtenidas algorítmicamente. Los fractales son generados por procedimientos, y el programa de computación es capaz de dibujar una curva del copo de nieve en varios niveles de detalles. Los fractales son útiles cuando se hacen movimientos en 3-D y se introducen números aleatorios. Éstos pueden ser utilizados para crear arreglos de números que serán generados usando procesos subdivididos, los cuales son usados como mapas de texturas, que representan niebla y nubes. Los sistemas de animación comercial proporcionan herramientas para modelar terrenos fractales.

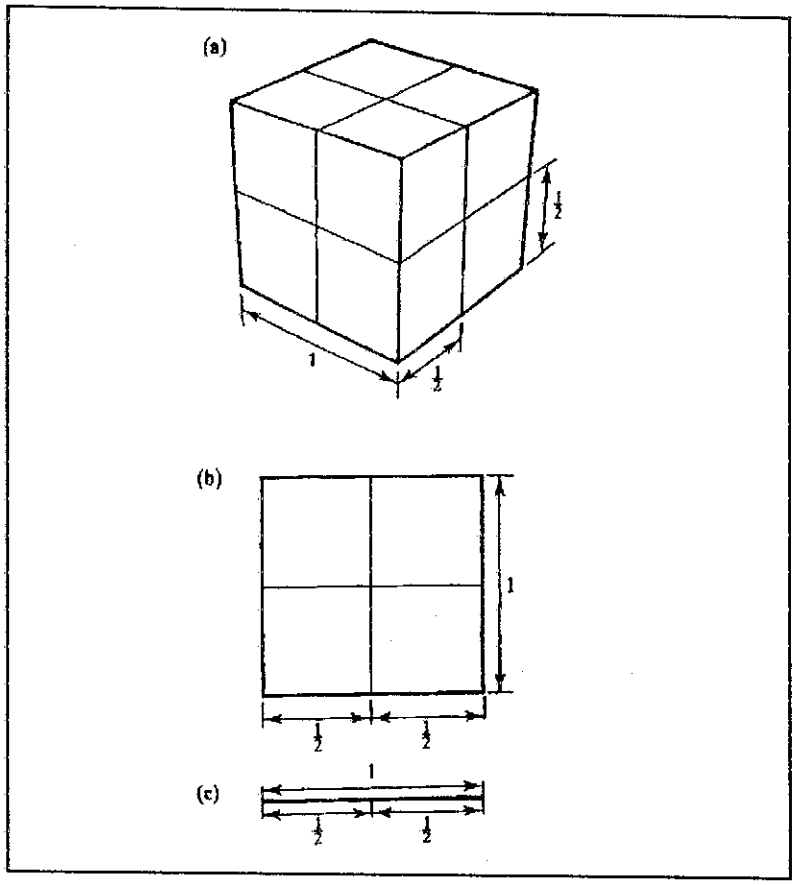



**Figura 2.18**: la idea de la similitud puede ser demostrada en un cubo, cuadrado o línea. En (a), un cubo puede ser reemplazado por ocho cubos similares multiplicado por un factor de 1/2. En (b), un cuadro puede ser reemplazado por cuatro cuadrados similares por un factor de 1/2; y en (c), una línea puede ser reemplazada por dos líneas similares por un factor de 1/2.

## 2.4.2 OBJETOS BLANDOS

Una forma de describir objetos blandos es físicamente. Por ejemplo, una esfera cargada eléctricamente es suspendida del techo en medio de una habitación y no se puede mover, y se coloca un contador, el cual mide la resistencia del campo eléctrico. Este contador puede ser usado para trazar la resistencia del campo eléctrico colocándolo en varias posiciones y tomar las lecturas. Una forma de realizar esta operación es tomar medidas en puntos a distancias constantes, tal vez en la forma de una ventana cúbica regular. Con esta matriz 3-D de lecturas, se pueden trazar superficies de resistencia eléctrica constante; dichas 'isosuperficies' (isosurfaces) aparecen como una colección de esferas concéntricas con lecturas altas en el centro y bajas en los extremos.

Si una segunda esfera es introducida, tocando la primera, y se repite la misma medición, aparecerá una familia diferente de isosuperficies que consiste en dos esferas interconectadas. Si una de las esferas se mueve lentamente en relación con la otra, la isosuperfice original reflejará este cambio acercándose para mantener una superfice contínua. Pero, en algún punto, una superficie se dividirá en dos isosuperficies separadas alrededor de dos esferas, además todavía existirá alguna forma de atracción y resultará distendida a lo largo de una línea uniendo sus centros. La Figura 2.19 muestra este experimento como secciones en 2-D.

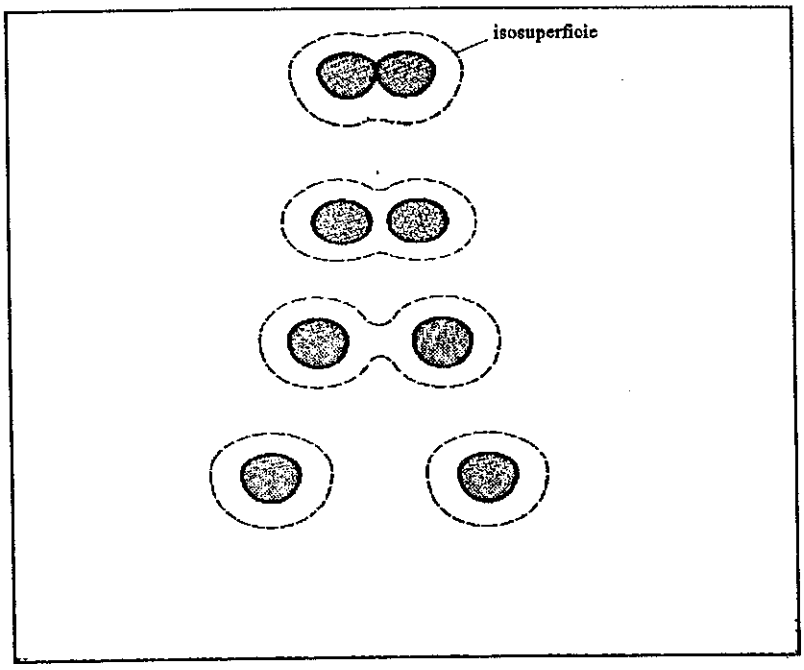



**Figura 2.19:** esta secuencia muestra como una isosuperficie es influenciada por posiciones relativas de dos objetos. Si los objetos se mueven lejos uno del otro, una simple isosuperficie se forma en dos, la cual todavía mostrará la existencia de otra, hasta que se muevan lo suficientemente lejos para que su influencia mútua sea insignificante.

Los objetos blandos no requieren de cargas eléctricas; están ímplicitamente definidos por ecuaciones que describen un campo escalar. Dichas ecuaciones pueden definir una esfera, cilindro, elipsoide, y un plano los cuales, cuando son combinados a varias distancias, pueden ser usados para construir una amplia variedad de estructuras.

Como la isosuperficie es creada de la adición o substracción de efectos de campos escalares, el mismo procedimiento puede ser usado para manipular otros atributos como el color.

## 2.4.3 MANIPULACIÓN PROCEDURAL

Las técnicas procedurales pueden ser usadas para manejar datos de coordenadas para crear objetos en 3-D, y un ejemplo es el de transformar datos de 2-D a estructuras en 3-D.

Otra técnica de manipulación es enrrollar el diseño alrededor de alguna forma, como un cilindro o una esfera. En el caso de un cilindro, se necesita definir el diseño en 2-D como un mapa direccionado por coordenadas $u$ y $v$, y encontrar funciones de mapeo para transformalas en un punto $(x,y,z)$ en el cilindro. Dadas las siguientes condiciones que se muestran en la figura 2.20, donde:

$r$ es el radio del cilindro
$h$ es la altura del cilindro
$0 <= u <= 1$
$0 <= v <= 1$

entonces el ángulo ß es dado por

$$ß = 2\pi u$$

y los valores de $x$, $y$ e $z$ son:

$$x = r\,sen(u), \qquad y = vh, \qquad z = r\cos(u)$$

Estas funciones se pueden verificar con substituir valores.

La otra función útil es mapear un punto $(u,v)$ para un punto $(x,y,z)$ sobre una esfera. La geometría para esta operación es mostrada en la Figura 2.21, donde una esfera de radio $r$ tiene su centro localizado en el origen. Una vez más se tiene un mapa direccionado en 2-D por las coordenadas-uv donde $u$ controla la longitud, y $v$ la latitud. El ángulo longitudinal ß está dado por:

$ß = 2\pi u,$

y el ángulo de latitud $\mu$ está dado por:

$\mu = \pi(v - 0.5);$

además los valores de $x$, $y$ e $z$ son:

$x = r\text{sen}(ß)\cos(\mu), \qquad y = r\text{sen}(\mu), \qquad z = r\cos(ß)\cos(\mu)$

Estas funciones se pueden verificar con substituir valores.

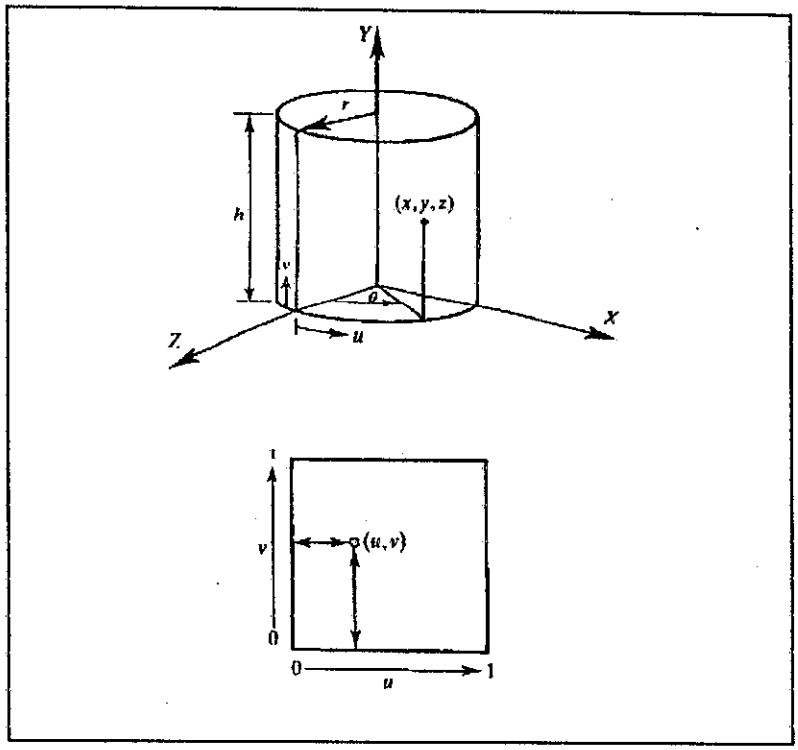



**Figura 2.20:** cualquier imagen 2-D direccionada por las coordenadas *(u,v)* puede ser mapeada para un punto *(x,y,z)* en el cilindro usando las funciones mostradas anteriormente.

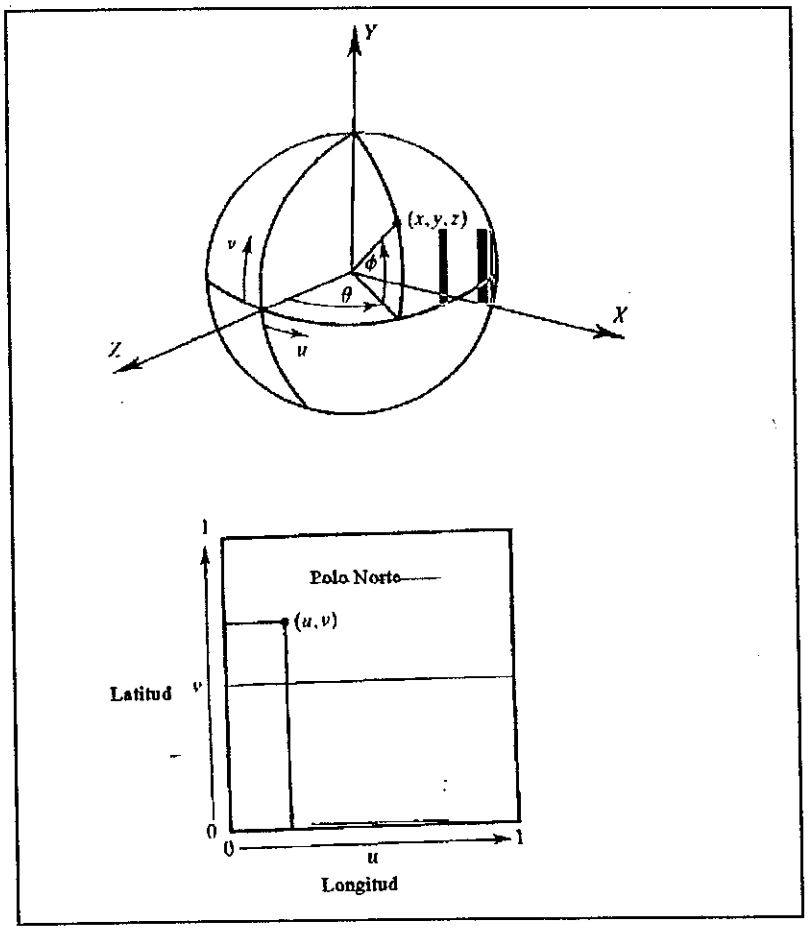



**Figura 2.21:** cualquier imagen direccionada en 2-D por las coordenadas *(u,v)* puede ser mapeado por un punto *(x,y,z)* sobre una esfera usando las funciones de mapeo que se indican arriba.

## 2.5 ESTRUCTURAS DE DATOS

Los lenguajes de programación proporcionan una variedad de mecanismos para organizar estructuras de datos tales como: tablas multi-dimensionales, árboles binarios, listas, anillos, árboles cuadráticos, y octrees. En las aplicaciones de gráficas por computadora, estas estructuras de datos son usadas para almacenar datos asociados con objetos, superficies livianas y de la cámara. Los parámetros de la cámara son fáciles de almacenar, como datos que controlan su posición y orientación, y quizá la longitud de enfoque. Los datos de la fuente de luz podrán ser almacenados en una tabla de dos dimensiones, donde cada entrada de la tabla es, el tipo de fuente de luz (punto, dirección y lugar), intensidad, color, posición, dirección, factor de disminución y ángulo del cono que pueden ser asignados cuando se necesiten.

El almacenar objetos ocasiona un problema, no existe un sistema geométrico en 3-D definido. Los objetos construidos con polígonos planos necesitan referenciar sus vértices, bordes, superficies normales, coeficientes de reflexión, transparencia, y coeficientes de brillo y aspereza. Los objetos construidos con cortes de superficies, necesitan los datos sobre las propiedades de la superficie, pero en lugar de los polígonos, necesitará mantener control de los vértices de los vectores, parámetros de tensión e inclinación. Los objetos procedurales tales como fractales no solamente requieren de un programa, sino también de diferentes estructuras de datos para el control de parámetros.

Los objetos construidos de elementos articulados pueden necesitar almacenar

los ejes alrededor de los cuales rotan los elementos, también los datos necesarios para controlar el grado de rotación permitido por cada eje.

En sistemas de geometría sólida constructiva, implica ecuaciones para describir la geometría primitiva y se necesitará una estructura de datos totalmente diferente de un sistema poligonal.

Desafortunadamente, no existen soluciones generales para estos problemas, ya que frecuentemente dependen de la disponibilidad de facilidades para definir estructuras de datos en los lenguajes de programación. FORTRAN, Pascal, C, C++ y LISP son lenguajes que proporcionan un ambiente que puede solucionar algunos de los problemas descritos anteriormente.

Los datos sobre los vértices, bordes y polígonos pueden ser almacenados en forma tabular, los cuales pueden ser mapeados fácilmente en arreglos. Sin embargo, la misma estructura de datos no permite objetos tales como octaedros, tetraedros, cilindros y dodecaedros. El problema principal se refiere a cuántos bordes deberá tener un polígono, y algunos sistemas de modelación imponen un límite máximo.

Entre algunos ejemplos de estructuras de datos que pueden ser usadas están las siguientes:

```
char map[1024][1024]     /* mip-map */
float u,v                /* coordenadas en el rango [0,1] */
float d                  /* compresión en el rango [0,1] */
float color[3]     /* para manejo de valores de rvb (rojo,verde,azúl)*/
float color_claro[3],color_obscuro[3] /* para manejar las tonalidades */
int nivelsimple   /* booleano que se mantiene en el tope
                o en la base de la pirámide*/
int mapsize       /* longitud del nivel en los pixels
                        de textura */
int iu,iv,ui,vi,iu_n,iv_n,ui_n,vi_n /* offsets en el mip-map */
float inf,sup,ru,rv,dinterp         /*parámetros de interpolación*/
```

Por ejemplo para el mapeo de textura se utiliza la siguiente estructura de datos:

```
char map[1024][1024] /* para el mip_map */
int vertex_no;      /* número de vértices que conforman un polígono */
point *vertex_list   /* lista de vértices */
void (*uvproject()   /* función de proyección que muestra las
                        coordenads u,v sobre un objeto */
float color[3]
point *screen_vertex,object_vertex  /* para los vértices de la pantalla */
                                    /*  vértices de los objetos a animar */
float last_u,last_v,u,v,uav,vav,d,texture_area=0
int i
```

Para el mapeo del cromado se utiliza la siguiente estructura:

```
/* Map_cromo() regresa el color del polígono mapeado */

void map_cromo(map,nu_vert,list_vert,list_normal,color)
char mapa[1024][1024]          /* mip map */
int nu_vert              /* número de vértices que forman el polígono */
point *list_vert         /* lista de vértices */
point *list_normal       /* lista de normales */
float color[3]
```

Para el algoritmo de mapeo del medio ambiente de alguna escena a animar, se usa la siguiente estructura:

```
/* numeración de las caras y bordes del cubo */

#define cara0 1
#define cara1 2
#define cara3 8
#define cara4 16
#define cara5 32

#define borde01 3
#define borde02 3
#define borde03 9
#define borde04 17
#define borde12 6
#define borde23 12
#define borde34 24
#define borde41 18
#define borde51 34
#define borde52 36
#define borde53 40
#define borde54 48


point cubo[nu_max] /* almacena la proyección del haz reflejado sobre
                      el cubo */
int nu_cubo        /* almacena el número de puntos que se forman en la
                      proyección*/
float uv[nu_max][2] /* almacena los valores uv de la proyección para una
                       cara dada */
```

# CAPITULO 3

# ILUMINACION

## INTRODUCCION

Se han visto las diferentes estrategias para crear modelos y describir formas de objetos, aunque no se ha mencionado cualquier atributo físico de dichos objetos como el color, transparencia, etc. El proceso de representación crea una imágen a color que requiere de la geometría y datos de los atributos.

Se sabe que el sol o una pantalla de televisión son lumínicos o son iluminados por alguna fuente de luz. Además, los sistemas de Animación por Computadora deben ser capaces de servir para una variedad de tipos diferentes de fuentes de luz las cuales puedan ser alterados en intensidad y color, y obtener diferentes posiciones en el WCS.

Para desarrollar una imágen a color se necesita saber el color de cada objeto, cómo su superficie refleja la luz y la naturaleza de las fuentes de luz. Durante el proceso de representación, la posición relativa de los objetos se debe mantener.

## 3.1 MODELOS DE ILUMINACION

Los modelos de iluminación se refieren a los diferentes tipos de fuentes de luz que pueden ser simulados dentro de las gráficas por computadora; ésto intenta diseñar los distintos tipos de fuentes de luz que se pueden encontrar en la vida real.

En los hogares, las instalaciones de luz incluyen paneles de difusión, interruptores, focos. Las habitaciones tienen ventanas para dejar entrar la luz externa. Además si en una representación de imágenes, ésta puede calcular los niveles de luz sobre las superficies de los objetos, entonces requerirá de datos exactos para relacionar diferentes modos de iluminación.

La simulación de la realidad representa algunos problemas. Los reflejos que se obtienen dentro de una habitación. Tal vez puede que exista una luz que cuelga del techo que refleja la luz en todas direcciones. La luz reflejada en el techo es generalmente blanca. El resto de la luz que se refleja en las paredes y el piso el cual absorbe algo de la energía, y refleja el resto. La televisión podría ser un ejemplo de luz a color que se refleja en la habitación la cual varía según la naturaleza del programa de televisión. Las ventanas en una habitación permiten que entre la luz del día en muchos ángulos, eso depende de las superficies cercanas a la ventana.

La simulación de las fuentes de luz reales requiere de una exactitud entre el tiempo de representación y el nivel de realismo. Podría ser imposible imitar el comportamiento de cada rayo de luz de una lámpara, la superficie de absorción, refracción, reflejo y los patrones de interferencia que ocurren en situaciones naturales. Además, se espera que algunas imágenes generadas por computadora no parezcan naturales. Esto es, porque las sombras se pueden perder en el diseño; las intensidades no son balanceadas; o que las superficies aparenten ser plásticas, donde deberían ser metálicas o de otro material.

### 3.1.1 PUNTOS FUENTES DE LUZ

El punto de la fuente de luz es la fuente de iluminación para diseñar, además se tiene que especificar su posición en el espacio con su intensidad y color. Su posición no representa problemas sólo implica identificar un lugar en

el WCS para que pueda iluminar las partes importantes de una escena. Cuando una fuente de luz es insuficiente, se puede usar luz extra en diferentes posiciones. Sin embargo, se necesita codificar su color e intensidad.

El manejo de colores significa que la representación tiene que crear tres intensidades de colores para cada pixel, uno para cada color primario. Esto implica que las fuentes de luz pueden consistir de tres luces de color en una misma posición y sus intensidades son ajustadas para crear el color requerido. Por ejemplo, si se necesita un tono amarillo, el componente azúl se colocará a cero y los componentes rojo y verde se ajustarán para dar la intensidad requerida. Así, en general, una fuente de luz requerirá de seis parámetros, la localización de las coordenadas *x*, *y* e *z*, y los tres números que definen las intensidades de los colores primarios.

Especificar el color de una fuente de luz si se usan niveles de rojo, verde y azúl es bastante incómodo, además es normal encontrar el matíz *(H)*, saturación *(S)* y valor de notación *(V)* a ser utilizado por la interface. Los valores *H*, *S* y *V* son convertidos internamente a valores equivalentes de R (rojo), G (verde) y B (azúl).

Como ejemplo, se consideran las características de la iluminación de un punto de la fuente de luz colocado en (0,10,0) y un punto sobre el plano. Ahora la energía de la luz incidente por unidad de área es proporcional al ángulo contenido entre la dirección de la luz y la superficie normal (Figura 3.1). Además para calcular la energía de la luz incidente en cualquier punto *(x,y,z)* se necesita saber como es la superficie en este punto, y un vector para el punto de luz. Para el plano la superficie normal **n** es igual a:

$$\mathbf{n} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Para calcular la energía de la luz incidente en cualquier punto *(x,y,z)* de una fuente de luz localizada en $P(p_x,p_y,p_z)$, el vector **s** que representa la luz incidente es igual a:

$$\mathbf{s} = \begin{bmatrix} p_x - x \\ p_y - y \\ p_z - z \end{bmatrix}$$

y el ángulo ß entre los vectores **s** y **n** se encuentra al usar la operación punto:

$$s \cdot n = |s||n|\cos(ß)$$

donde $|\mathbf{s}|$ es la magnitud de **s**, $|\mathbf{n}|$ es la magnitud de **n**, y la operación punto es igual a:

$$s \cdot n = (s_1n_1 + s_2n_2 + s_3n_3)$$

En este ejemplo $|\mathbf{n}| = 1$, por lo que la expresión punto se puede reescribir como:

$$\cos(ß) = \frac{s.n}{|s|} = \frac{((p_x - x)0 + (p_y - y)1 + (p_z - z)0)}{|s|}$$

$$= \frac{p_y}{|s|}$$

$$= \frac{10}{|s|}$$

Además, inmediatamente baja la fuente de luz (el orígen) ß = 0° y la intensidad es máxima; pero el punto (10,0,0), cos(ß) = 0.707, y la intensidad incidente es disminuida a 70.7%.

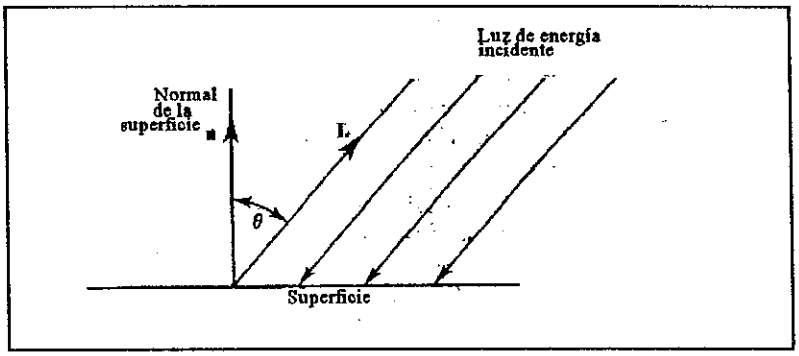



Figura 3.1: Cuando la luz es incidente a la superficie, su intensidad es proporcional al ángulo que se forma con la superficie normal; en este caso el ángulo ß.

## 3.1.2 FUENTES DE LUZ DIRECCIONAL

El sol -el cual se encuentra a una gran distancia de la Tierra- es una fuente de luz direccional. La gran distancia significa que los rayos de luz llegan a pequeñas partes de la superficie de la Tierra y que son virtualmente paralelas y de la misma intensidad. Esto significa que la fuente direccional de luz puede ser especificada en términos de un color, intensidad y dirección que son codificados en la forma de vector **s**. El ángulo entre este vector y la superficie normal es usado para representar la cantidad de energía de luz en una superficie inclinada. Por ejemplo, para modelar el sol en una posición elevada, la dirección de luz **s** podrá ser representada como el vector unitario:

$$\mathbf{s} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

Así la dirección de la luz solar del vector **s** es ortogonal al suelo. Pero como se ha visto, al calcular la energía de luz incidente sobre una superficie, el vector para la luz incidente fue dirigido hacia la fuente de luz. Además, el vector **s** deberá ser inverso para:

$$\mathbf{s} = -\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

Ahora si la superficie normal **n** del plano es:

$$\mathbf{n} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$ (verticalmente hacia arriba)

se puede calcular de nuevo el ángulo entre los dos vectores usando la operación punto:

$$s \cdot n = (0 + 1 + 0) = 1$$

y como $|\mathbf{s}|$ y $|\mathbf{n}|$ son iguales a la unidad:

$$s \cdot n = \cos(ß) = 1$$

lo cual significa que el ángulo entre los dos vectores es 0°.

Si la dirección del sol se cambia a:

$$\mathbf{s} = -\begin{bmatrix} 0.707 \\ -0.707 \\ 0.0 \end{bmatrix} \begin{bmatrix} -0.707 \\ 0.707 \\ 0.0 \end{bmatrix}$$

entonces el ángulo entre **s** y **n** es:

$$s \cdot n = 0.707$$

Además, cos(ß) = 0.707, y ß = 45°. Así la intensidad original de la luz solar es efectivamente reducida por un factor de 0.707 sobre el plano.

Esta es una simple operación para mostrar los efectos de iluminación del sol que se mueve a través del cielo, esto es necesario para la animación del vector dirección en una forma controlada.

## 3.1.3 SPOT LIGHT

Un spot light simula el comportamiento de un foco convencional el que crea un haz de luz en forma de cono; esto es llamado fuente de luz dirigida. El rango mínimo de parámetros necesarios para modelar esta fuente de luz son posición, intensidad, color y ángulo del foco. La función coseno puede ser usada para crear una intensidad reducida de un foco.

Por ejemplo, si un foco tiene las siguientes características:

Posición: $(p_x, p_y, p_z)$

Dirección: $\mathbf{s} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix}$

Angulo del cono: $\phi$

entonces un punto $(x,y,z)$ cae dentro del cono de iluminación si el ángulo ß entre el vector **s** y un vector **u** del foco para el punto muestra, es menor o igual a $\phi/2$.

Además si **u** es dado por:

$$\mathbf{u} = \begin{bmatrix} x - p_z \\ y - p_y \\ z - p_z \end{bmatrix}$$

entonces ß está dado por:

$$ß = \cos^{-1}(s.n/|s||u|)$$

Por ejemplo, la Figura 3.2 muestra el punto (10,0,0), que estará en el centro dede la base de impresión del spot.

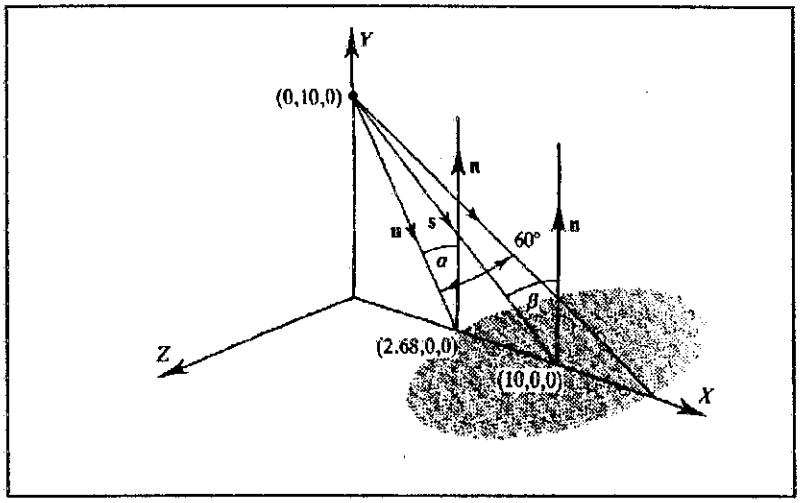



**Figura 3.2:** El spot light está localizado en el punto (0,10,0) y dirigido hacia el punto (10,0,0). Como el ángulo de incidencia es de 45 grados, su intensidad efectiva es reducida por el coseno de 45 grados.

### 3.1.4 OTRAS CARACTERISTICAS DE LAS FUENTES DE LUZ

*DISMINUCION DE LA INTENSIDAD CON RESPECTO A LA DISTANCIA*

Se sabe que la intensidad de la luz disminuye con la distancia: de hecho, la relación es la ley del cuadrado inverso. Esto significa que la intensidad observada es medida a una distancia de un metro, a dos metros la intensidad será un cuarto de la original. Los experimentos muestran que una buena aproximación puede ser lograda dividiendo la intensidad de luz incidente por $(d + k)$, donde $d$ es la distancia del observador desde la superficie iluminada y $k$ es una constante. Sin embargo, el cálculo de $d$ involucra una raíz cuadrada y tres operaciones cuadráticas.

Una función alternativa basada en observacion empíricas es:

$$F(d) = R_d$$

donde $d$ es la distancia desde la fuente de luz, y $R$ es un parámetro. Por ejemplo, si a 2 unidades desde la fuente de luz la intensidad es disminuida por 0.5, entonces $R$ necesita ser puesta a conveniencia. $R$ es calculada de la siguiente forma:

$$R = \text{atten}^{(1/d)}$$

*NIVELES DE ILUMINACION BALANCEADOS*

Cuando son usados diferentes fuentes de luz, es muy probable que algunas superficies puedan resultar saturadas con luz y son interpretadas como un color constante. Esto significa que se deben ajustar las diferentes intensidades para lograr un balance aceptable, lo que resulta en más tiempo de trabajo.

*SOMBRAS*

Aunque las sombras son naturales en el mundo real, en el mundo virtual de las gráficas por computadora deben ser calculadas durante el proceso de representación. Cuando dichos cálculos no son seguros, el modelo de iluminación permite para los rayos de luz iluminar una superficie y todas las superficies detrás de la primera como si ésta no existiera. Obviamente esto puede crear anomalías visuales, y es más difícil generar imágenes que no parezcan naturales.

*LUZ NEGATIVA*

Un aspecto interesante de las fuentes de luz por computadora es que, como su intensidad es definida en términos de un número positivo haciendo su signo negativo en lugar de agregar iluminación para una superficie, se puede quitar iluminación y crear una fuente de 'obscuridad'. Aunque esto tiene aplicaciones limitadas, es, sin embargo, una característica interesante y es muy útil para crear efectos suaves de sombra.

## 3.2 MODELOS DE REFLEJOS

Los modelos de reflejos son usados para describir la forma del reflejo de la luz incidente en una superficie -existen tres tipos de reflejo de luz: de ambiente, difusa y especular (es un reflejo de una fuente de luz visto por un observador sobre una superficie reflectiva). La luz ambiental simula el nivel de luz constante que puede ser considerada suficiente para reflejos múltiples, mientras que la luz difusa es creada para superficies cuya rugosidad causa una

luz incidente para reflejar igualmente en todas direcciones. La luz specular se refiere a la luz reflejada por superficies brillosas y crea una iluminación de la fuente de luz y de su entorno.

El color de un objeto es determinado por la naturaleza de la radiación de la iluminación y las proporciones que son absorbidas y reflejadas. En general, la luz que recibimos del sol es una mezcla igual de todas las frecuencias visibles, depende de las condiciones climáticas. Esta mezcla balanceada de frecuencias causa una sensación de blancura, cuando se habla de papel blanco, significa que es una superficie que refleja la luz radiada de todas las frecuencias; cuando se observa un objeto de otro color dicho objeto absorbe algunas de las frecuencias y refleja otras.

Las fuentes de luz son consideradas como tres componentes de brillo: rojo, verde y azúl. Además las superficies, tienen tres coefecientes de reflejo, los cuales especifican la cantidad fraccional de luz roja, verde y azúl reflejada. De esta forma es posible simular la interacción de objetos de color con luces de color.

## 3.2.1 LUZ AMBIENTAL

En una habitación normal iluminada por una lámpara colgante y tal vez expuesta a la luz del día a través de la ventana, muchas de las superficies internas aparecerán en un nivel constante de iluminación modulada por otros cambios pequeños de intensidad. Este medio ambiente puede ser simulado si se asume que existe un nivel de iluminación de fondo, sobre el cual se pueden sobreponer otras intensidades de luz. De hecho no existe un nivel de iluminación que permita que toda la habitación, para cada superficie, sea iluminada por la luz que ha logrado algún nivel natural de equilibrio a través de muchos niveles de absorción y reflexión.

Este nivel constante de luz es llamado el componente ambiental. No tiene fuente como se representa la luz que llega a todas direcciones.

Cuando se calcula la luz total vista por el observador, el componente ambiental es representado por:

$$I_a K_a$$

donde $I_a$ es la intensidad ambiental incidente y $K_a$ es un parámetro asociado con una superficie para controlar la proporción reflejada por la superficie. Como se ha descrito, esta expresión deberá ser evaluada por las porciones de rojo, verde y azúl del espectro.

Durante una secuencia de animación la intensidad y el color de la luz ambiental puede ser cambiada bajo el control automático de la computadora.

## 3.2.2 REFLEXION DIFUSIVA

Las superficies rugosas como alfombras, algunos textiles y papeles reflejan la luz en direcciones aleatorias. Esto significa que cuando estas superficies son observadas, parecen tener una intensidad constante sin importar la posición del observador, aunque su brillantés cambia cuando su orientación con la fuente de luz se altera. Dicha reflexión termina difusa. Para simular dicho comportamiento dentro del ambiente de las gráficas por computadora, no es necesario saber donde se encuentra el observador, pero si el ángulo entre la fuente de luz y la superficie.

Se considera el caso cuando la fuente de luz direccional está arriba de la superficie: la luz incidente se reflejará en el medio ambiente en varias direcciones aleatorias. Sin embargo, si una superficie está colocada tal que los rayos de luz están a 45°, entonces la iluminación incidente efectiva es reducida por un factor de 0.707. Si, en un caso extremo, el ángulo fue aumentado a 90°, teóricamente la superficie no recibirá del todo ninguna luz incidente. Para

calcular el componente de difusión se usará el término coseno para disminuir la intensidad -esta ley es conocida como la ley de Lambert.

El término de difusión para una fuente de luz es calculado por:

$$I_i K_d (L.n)$$

donde $I_i$ es la intensidad de la fuente de luz, $K_d$ es una constante fraccional asociada con una superficie para controlar la proporción de luz difusa reflejada, **L** es el vector dirigido hacia la fuente de luz, y **n** es el vector normal de la superficie. Esta expresión de difusión debe ser evaluada tres veces para los componentes rojo, verde y azúl.

## 3.2.3 REFLEXION SPECULAR

Las superficies lisas o pulidas producen reflexiones especulares, que son reflejos que se pueden ver también en ventanas y espejos, y las reflexiones de las fuentes de luz que se pueden ver en otras superficies que no son lisas. La trayectoria del rayo fue desarrollada para calcular los reflejos del medio ambiente.

Al usar espejos y superficies pulidas muestran que la posición relativa del observador a la superficie es importante. De hecho, dos leyes de reflexión especular establecen que los ángulos de incidencia y reflexión son iguales, y que el rayo de incidencia, el rayo reflejado y la superficie normal en el punto de reflexión, todas están situados en el plano. La Figura 3.3 ilustra lo anterior.

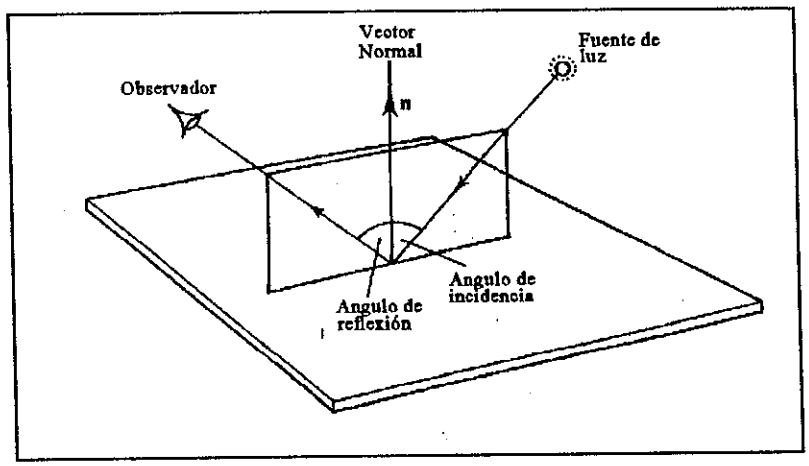



**Figura 3.3:** Las dos leyes de la óptica: la primera establece que el ángulo de reflexión es igual al ángulo de incidencia. La segunda establece que en el punto de reflexión, el rayo incidente, el rayo reflejado y el vector normal están situados en el mismo plano.

Como las fuentes de luz son teóricamente puntos, podría ser no real hacer que todos los reflejos de las fuentes de luz aparezcan como un simple pixel, especialmente cuando las fuentes de luz física tienen un tamaño finito y los reflejos resultan manchados por diferentes grados de superficie rugosa. Por lo que hay que tratar de simular su comportamiento. Una aproximación más cercana a este efecto puede ser creado al permitir al observador ver una reflexión especular aún cuando no se encuentra localizado en un punto óptimo. Como ilustra la Figura 3.3, si los ángulos de incidencia y reflexión son ß, y el ángulo de error es $\phi$, entonces el componente especular puede ser calculado por la siguiente expresión:

$$I_i K_s \cos^g(\phi)$$

donde $I_i$ es la intensidad de la luz incidente, $K_s$ es el coeficiente especular de color independiente, $\phi$ es el ángulo de error y $g$ es un término que determina el brillo de la superficie. Para calcular $\phi$, ß debe conocerse, además la expresión es escrita como:

$$I_i K_s (R \cdot V)^q$$

donde el término coseno es reemplazado por la operación punto de los vectores **R** y **V**; **R** es el vector unitario que representa el rayo reflejado, y **V** es el vector unitario dirigido hacia el observador. El cálculo de **R** no es fácil, **R** . **V** es calculado indirectamente de la siguiente forma usando el vector unitario **L**:

$$\begin{aligned} \mathbf{V} \cdot \mathbf{L} &= \cos(2\beta + \phi) \\ &= \cos(2\beta)\cos(\phi) - \operatorname{sen}(2\beta)\operatorname{sen}(\phi) \\ &= \cos(\phi)(\cos^2(\beta) - \operatorname{sen}^2(\beta)) - 2\operatorname{sen}(\beta)\cos(\beta)\operatorname{sen}(\phi) \end{aligned}$$

y

$$\begin{aligned} \mathbf{n} \cdot \mathbf{V} &= \cos(\beta + \phi) \\ &= \cos(\beta)\cos(\phi) - \operatorname{sen}(\beta)\operatorname{sen}(\phi) \end{aligned}$$

Combinando las dos ecuaciones se tiene:

$$\mathbf{R} \cdot \mathbf{V} = 2(\mathbf{n} \cdot \mathbf{L})(\mathbf{n} \cdot \mathbf{V}) - \mathbf{V} \cdot \mathbf{L}$$

así la expresión specular original queda como:

$$I_i K_s (2(\mathbf{n} \cdot \mathbf{L})(\mathbf{n} \cdot \mathbf{V}) - (\mathbf{V} \cdot \mathbf{L}))^q$$

### 3.2.4 EXPRESION FINAL PARA REPRESENTAR EL REFLEJO

Este modelo simple de reflexión tiene tres componentes para ambiente, difusión y reflexión specular. El total de la luz reflejada por una superficie puede ser expresada como la suma de éstos tres componentes:

$$I = I_a K_a + [I_i K_d (L.n) + I_i K_s (R \cdot V)^q]$$

pero es solamente para una fuente de luz, además la expresión entre corchetes debe ser calculada para cada fuente de luz dentro del medio ambiente de animación.

La naturaleza virtual de éstas fuentes de luz y de los objetos significa que se pueden tomar ciertas libertades sobre las leyes de la física. Por ejemplo, una fuente de luz que es establecida tal que esté solamente asociada con ciertos grupos de objetos. No existe una forma de prevenir sobre como colocar las fuentes de luz dentro de otros objetos. Cualquier parámetro puede ser cambiado durante una secuencia de animación, y este nivel de flexibilidad puede ser impresionante; sin embargo, la representación es una operación que consume bastante tiempo.

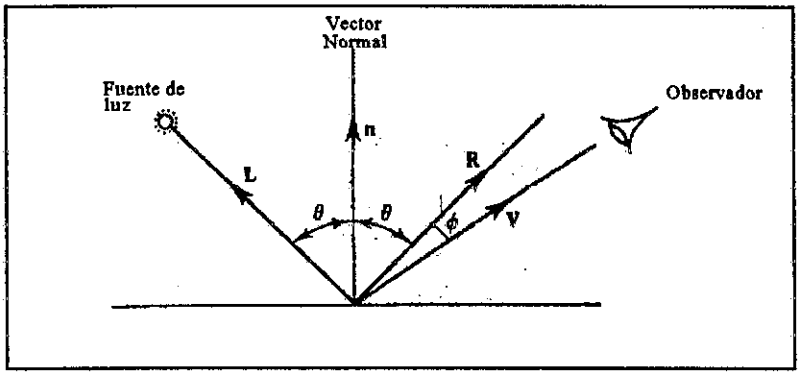



**Figura 3.4:** El diagrama muestra a un observador no alineado con el vector de reflejo V por 15 grados.

## 3.3 TECNICAS DE SOMBREADO

En las secciones anteriores se han visto las ideas sobre difusión y reflexión especular, así como la luz ambiental, disponibles para realizar un modelo para calcular niveles de luz sobre las superficies. En esta sección de desarrollarán éstas ideas para crear vistas sombreadas de objetos en 3-D.

Si bien existen diferentes formas de almacenar una imágen sombreada, por simplicidad, se asumirá la existencia de un frame store de 24-bits, donde cada color primario es mantenido para una precisión de 8-bits. Además, los algoritmos de sombreado se describirán asumiendo que los procedimientos de remosión de superficies escondidas se asegurarán que las superficies estan representadas en el frame store en una secuencia correcta.

Para los programas de sombreado las intensidadesde color para un frame store, se introducen dos tipos de error en la imágen. El primero es originado por las intensidades de color que son mantenidas como enteros, aunque una intensidad puede ser calculada como 230.45, por ejemplo, tiene que ser redondeado a 230 para ser almacenado en el frame store. Así el coloreado será dominado para un error de cuantización. El segundo error es más importante, y es originado por la imágen a ser construída de innumerables puntos.

### 3.3.1 SOMBREADO DE GOURAUD

Henri Gouraud (Gouraud, 1971) hizo la primera contribución para algoritmos de sombreado, procesando una técnica de interpolación lineal para mezcla de colores a través de una superficie plana. Gouraud sugirió que el ojo podría ser engañado en el momento de observar las intensidades de color de las superficies que han sido mezclados por algunas muestras. Este algoritmo puede ser ejemplificado en la Figura 3.4, la cual muestra un triángulo 3-D iluminado por un simple punto de fuente de luz. Esto implica que los vértices *A*, *B* y *C*, cada uno tendrá diferentes intensidades de luz, ya que los rayos de luz incidentes de una fuente cercana de luz subtendrá un escaso ángulo diferente a la superficie normal. En el caso de una fuente infinita, los rayos de luz serán paralelos, y los ángulos de incidencia serán idénticos.

Si las intensidades de luz en los tres vértices de un triángulo pequeño son $I_a$, $I_b$ e $I_c$, entonces parece razonable asumir que a lo largo de las uniones de los bordes de los vétices *A* a *B* los cambios de intensidad de luz son linealmente de $I_a$ a $I_b$. Esto significa que la interpolación puede ser hecha en el espacio de la imágen usando las coordenadas *x* e *y* de los vértices (Figura 3.5). Si los tres vétices *A*, *B* y *C* tienen coordenadas en el espacio de imágenes de:

$$A(x_a, y_a), \qquad B(x_b, y_b), \qquad \text{y} \qquad C(x_c, y_c)$$

entonces la intensidad $I_L$ en el punto $(x_L, y_s)$ es dado por una interpolación lineal usando $y_a$, $y_c$ e $y_s$:

$$I_L = \frac{I_a(y_s - y_c) + I_c(y_a - y_s)}{y_a - y_c}$$

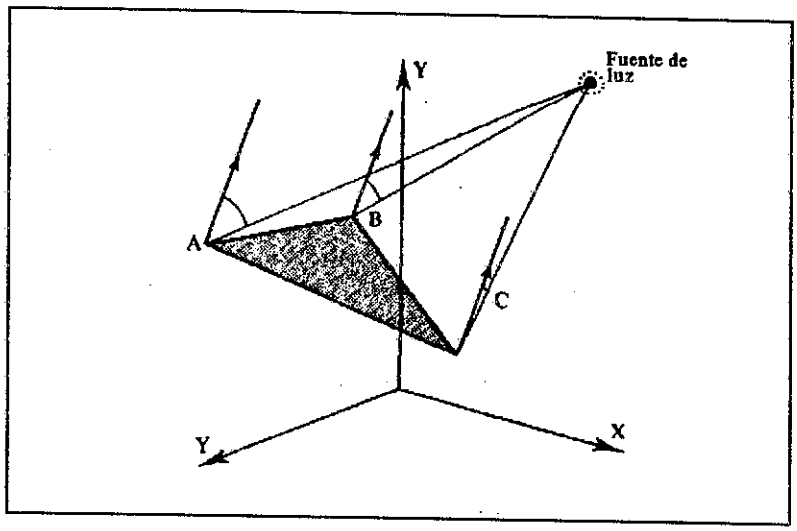



**Figura 3.5:** Si una fuente de luz está relativamente cercana al triángulo, el rayo de luz incidente subtendrá ángulos diferentes en los vértices *A*, *B* y *C*. El sombreado de Gouraud interpola éstas intensidades a lo largo de los bordes a través del interior del triángulo para crear un efecto de sombreado de difusión continua.

Similarmente, para la parte derecha, la intensidad $I_R$ en el punto $(x_R, y_s)$ está dado por una interpolación lineal usando $y_a$, $y_b$ e $y_s$:

$$I_R = \frac{I_a(y_s - y_b) + I_b(y_a - y_s)}{y_a - y_b}$$

Ahora se tiene la intensidad $I_L$ en la parte izquierda del triángulo e $I_R$ en la parte derecha del triángulo, por lo que se puede interpolar linealmente entre ellos para calcular la intensidad $I_s$ en $(x_s, y_s)$ usando $x_L$, $x_R$ y $x_s$:

$$I_s = \frac{I_L(x_R - x_s) + I_R(x_s - x_L)}{x_R - x_L}$$

Ahora se tiene una intensidad de luz $I_s$ para una posición $(x_s, y_s)$ sobre algún raster, y al procesarlo el triángulo se podrá sombrear el mismo.

Se ha encontrado que las imágenes a color pueden ser creadas si se separa la imágen dentro de los colores primarios componentes, rojo, verde y azúl, los cuales pueden ser almacenados dentro del frame store de 24-bits. Así, para producir una imágen total a color al usar el sombreado de Gouraud, el algoritmo proporciona las intensidades de los pixels interpolados para rojo, verde y azúl. Si los polígonos individuales que definen un objeto son sombreados al usar esta técnica, la imágen final obviamente aparecerá como se había pensado construir desde una superficie facetada, por lo cual es llamada 'facetada' o 'sombreado liso'.

Gouraud hizo una extensión de esta técnica donde un modelo 3-D podría ser sombreado para que pareciera blando, y que sería construído de una colección de superficies planas. Para obtener esto, hay que asegurarse que la intensidad de luz asociada con cualquier vértice deberá ser idéntico para cada superficie compartida por dicho vértice.

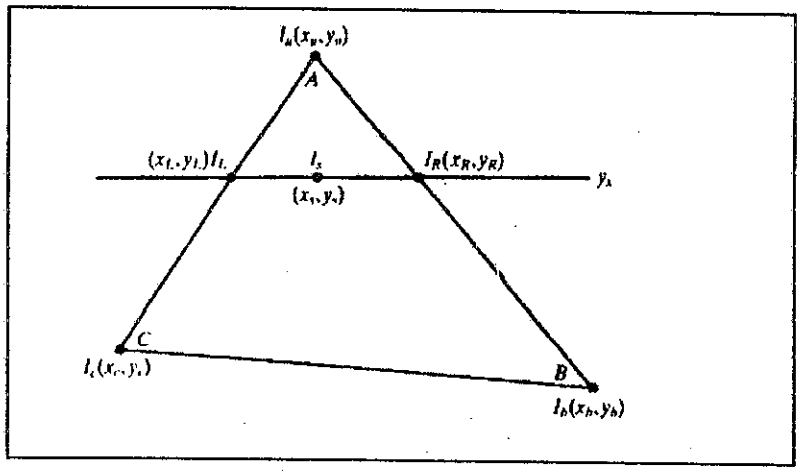



**Figura 3.6:** El sombreado de Gouraud obtiene la intensidad de pixels al mostrar la iluminación incidente en los vértices de este triángulo para dar $I_a$, $I_b$ e $I_c$. Estos son linealmente interpolados a lo largo de los bordes para producir $I_L$ e $I_R$, los cuales son interpolados a través de los pixels encerrados para calcular $I_s$.

## 3.3.2 SOMBREADO DE PHONG

Donde el sombreado de Gouraud generalmente asume una superficie reflejada difusa, el sombreado de Phong simula una superficie púlida al agregar reflejos especulares para el componente difuso. Esto es una aproximación para lo que en realidad es un fenómeno complejo, pero en la práctica crea imágenes aceptables.

Para calcular el componente especular, el algoritmo de sombreado de Phong necesita retener un conocimiento de las relaciones geométricas entre el observador, el objeto y la fuente de luz. Para cuando se mantienen ciertos ángulos entre éstos elementos se observará un brillo especular. El centro del cálculo especular es la superficie normal en el rayo de luz incidente como se muestra en la Figura 3.2 y Figura 3.3, al observador se le permite ver un brillo atenuado aún y cuando el ángulo de observación no es precisamente igual al ángulo de reflexión. Para superficies planas con sombreado liso, la superficie normal es constante, además el sombreado de Phong evalúa el componente especular adicional que deberá ser agregado al componente de ambiente y al componente de difusión.

Para calcular el componente especular se necesita calcular un vector normal para cada pixel, el cual involucra un cálculo geométrico considerable; pero la idea de Phong dice que este vector normal es interpolado en el espacio de la imágen al usar promedios normales en los vértices del polígono. Como un ejemplo, se considera el triángulo *ABC* en la Figura 3.6, donde las normales $\boldsymbol{n}_a$, $\boldsymbol{n}_b$ y $\boldsymbol{n}_c$ son promedios normales calculados de las superficies normales sombreadas a un vértice común. Ahora se puede interpolar linealmente $\boldsymbol{n}_a$ y $\boldsymbol{n}_c$ para producir $\boldsymbol{n}_L$ usando los valores de $y_a$, $y_c$ y $y_s$ (Figura 3.7):

$$n_L = \frac{n_a(y_s - y_c) + n_c(y_a - y_s)}{y_a - y_c}$$

Se puede dar una expresión similar para $\boldsymbol{n}_R$:

$$n_R = \frac{n_a(y_s - y_b) + n_b(y_a - y_s)}{y_a - y_b}$$

Además, al igual que el sombreado de Gouraud, se puede interpolar linealmente entre $\boldsymbol{n}_L$ y $\boldsymbol{n}_R$ para calcular $\boldsymbol{n}_s$:

$$n_s = \frac{n_L(x_r - x_s) + n_R(x_s - x_L)}{x_R - x_L}$$

Ahora se tiene una pseudo-superficie normal $\boldsymbol{n}_s$ que puede ser substituída en el componente de reflexión especular:

$$I_i K_s (2(\mathbf{n_s} \cdot \mathbf{L})(\mathbf{n_s} \cdot \mathbf{V}) - (\mathbf{V} \cdot \mathbf{L}))^q$$

Aunque el sombreado de Phong produce imágenes bastante aceptables, todavía son no reales, ninguna de las técnicas de sombreado de Gouraud y Phong considera los efectos ópticos de sombras, reflexiones múltiples,colores que se corren y espejos.

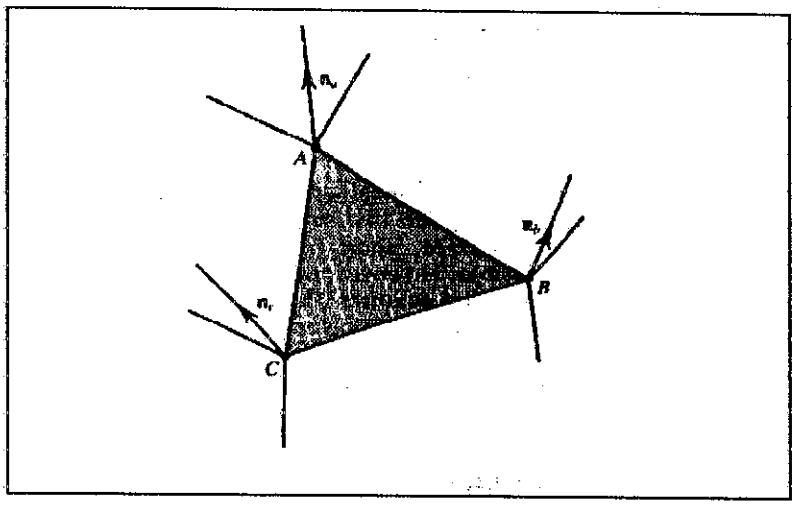



**Figura 3.7:** En esta mezcla de triángulos los vértices tienen que ser asignados a promedios normales; estos son interpolados en la técnica de sombreado de Phong para crear un sombreado especular suave.

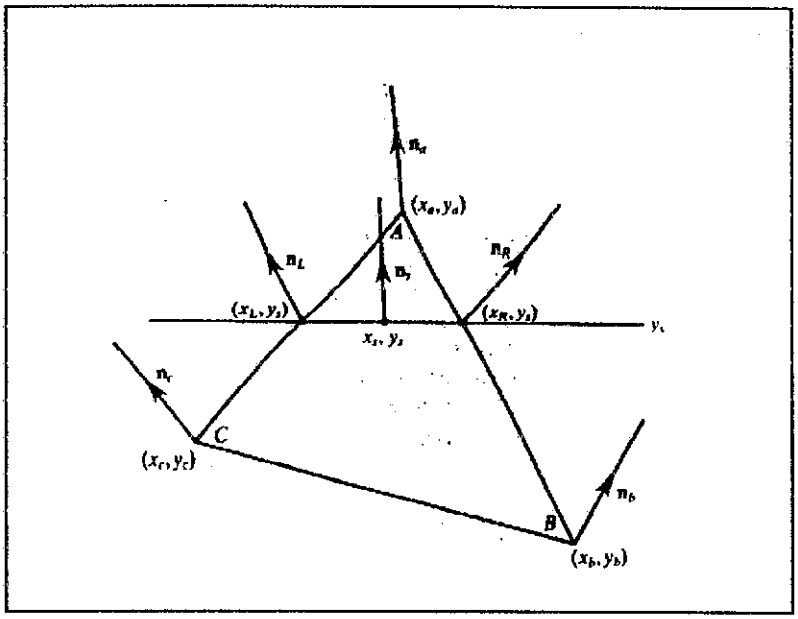



**Figura 3.8:** El sombreado de Phong deriva intensidades de pixel al calcular primero la superficie promedio normal $n_a$, $n_b$ y $n_c$ en los tres vértices. Estos son interpolados linealmente a lo largo de las orillas para producir $n_L$ y $n_R$, los que son interpolados a través de los pixels encerrados para calcular $n_s$. Este vector final es substituido en la expresión de reflexión especular.

## 3.4 TECNICAS PARA OCULTAR SUPERFICIES

Los algoritmos para ocultar superficies aseguran que cuando una escena es realizada, los objetos cercanos a la cámara estarán más ocultos que los demás. Este es un fenómeno que ocurre también en la vida real, así como nosotros -con nuestro sentido de la visión- recibimos ondas de luz que interactúan con objetos opacos y transparentes. En un ambiente de computación, los objetos se describen al usar una variedad de técnicas para modelar, y básicamente son mantenidos como una colección de números. Relaciones espaciales también son mantenidas numéricamente en forma de coordenadas.

Muchos objetos en la vida real tienen una estructura rígida que deja a un objeto penetrar en otro; como, líquidos y gases, permiten pasar objetos a través de ellos; algunos son elásticos y pueden ajustar su forma cuando tienen contacto con otros objetos. Estos efectos no se pueden

asociar con un simple modelo numérico; deben ser diseñados en un sistema de animación.

La naturaleza numérica de los modelos de computación origina otros problemas: imaginarse, por ejemplo, el caso de dos o más objetos que coexisten en el mismo espacio de volumen. En realidad, un objeto normalmente desplaza a otro cuando es movido en el mismo espacio, aunque los líquidos y gases (por su estructura molecular) pueden compartir un espacio común -mientras dentro de una computadora un objeto es trasladado a otra posición en el WCS al modificar los valores de sus coordenadas.

## 3.4.1 REMOCION BACK-FACE

Una forma simple de remover superficies ocultas es ignorar aquellas superficies que están demasiado lejos de la cámara. Para objetos convexos como un cubo, esfera, cilindro, tetraedro, octaedro, icosaedro y un dodecaedro, la remoción back-face crea una imágen sólida natural. Sin embargo, si uno de estos objetos es colocado en frente de otro, tendrán una apariencia similar.

Cualquier polígono planar, puede ser probado por una condición back-facing, al calcular la operación punto de su superficie normal con el vector, como se muestra en la Figura 3.8. El polígono B es forward-facing y además es visible como los vectores $n_b$ y $V_b$ que contienen un ángulo menor de 90°; sin embargo, el polígono $A$ es back-facing como el ángulo entre $n_a$ y $V_a$ que excede 90°.

## 3.4.2 ALGORITMO DE LOS PINTORES

Si un conjunto de polígonos en 3-D es mantenido en una secuencia arbitraria y son seleccionados uno a la vez y dejados en el frame store, no se puede esperar que la imágen final refleje una pintura verdadera de una escena en 3-D. Probablemente lo que sucede es que algunos polígonos que están relativamente cercanos a la cámara serán dejados en el frame store. Dicho efecto parecerá raro, aunque puede ser evitado si los polígonos originales están almacenados en una secuencia más segura tal que el primer polígono a ser representado es el más lejano, y el último polígono es el más cercano. Esta simulación es cercana a la presentación de un paisaje donde el artista desarrolla el lienzo pintando primero el cielo, seguido por las montañas, luego el campo y los árboles, y por último decorar la escena con objetos de primer plano -de aquí el nombre de esta técnica.

La clasificación de los polígonos en una secuencia no siempre es fácil cuando se tienen que resolver situaciones donde, por ejemplo, un polígono rectangular pasa a través del centro de un anillo circular. Una parte del polígono está en frente del anillo, y la otra parte detrás de él; además el proceso de clasificación no será el adecuado para resolver el problema de la prioridad. Esto se soluciona fracturando los polígonos en pequeños elmenetos. El algoritmo de los Pintores tiene algunas restricciones como su incapacidad de manejar objetos interpenetrados y anti-aliasing.

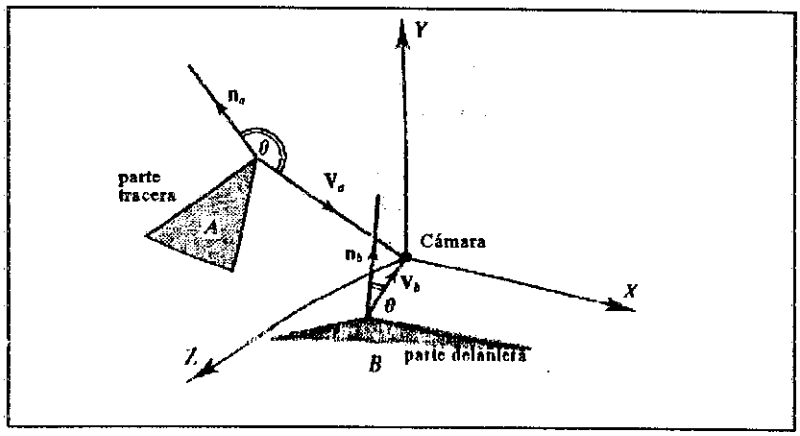



Figura 3.9: Las superficies ocultas pueden ser detectadas al no calcula el ángulo subtendido entre cámara y la superficie normal. En el polígono *A*, el ángulo es menor que 90° y es visible; aunque el ángulo para *B* excede 90° y es invisible.

## 3.4.3 ALGORITMO SCAN-LINE

Este algoritmo se basa en los cálculos de pixels asociados con una línea de la imágen en el frame store, y aplica el algoritmo para cada raster, lo que se hará cientos de veces, y así la escena completa puede ser representada. En general, el proceso empieza con la parte superior del scan-line y sigue con una línea a la vez hasta llegar a la parte inferior, el punto en el cual la imágen ya está hecha. Algunos sistemas de generación de imágenes son capaces de lograr este ciclo en algunos mili-segundos, lo cual proporciona el mecanismo para una animación en tiempo real.

Para calcular las intensidades de los pixels para un scan-line, el algoritmo inicialmente clasifica los objetos visibles en una secuencia vertical para poder mantener un registro de cuales objetos cruzan un scan-line específico. Esto entonces analiza aquellas superficies que intersectan la línea actual y resuelve la prioridad al considerar partes pequeñas de los polígonos necesarios para el raster. En esta forma puede manejar objetos inter-penetrating, e implementar estrategias de anti-aliasing.

## 3.4.4 Z-BUFFER

El algoritmo z-buffer proporciona un mecanismo muy elegante para remover superficies ocultas -desafortunadamente, tiene algunas deficiencias como su incapacidad de manejar superficies transparentes y anti-aliasing.

El z-buffer es un elemento de memoria para almacenar información para cada pixel en el frame store. Cuando el primer polígono es traducido, la intensidad del polígono es calculada para cada pixel afectado y, como estas intensidades serán menores que el valor inicial, serán usados para reescribir el z-buffer. El frame store es cargado con las intensidades de color correspondientes. Los polígonos lejanos serán representados en una secuencia arbitraria, sus intensidades son comparadas con las que ya se tienen, y si son similares, se reescribirán en el z-buffer y el frame store. Si la intensidad del pixel es más grande que los valores actuales, esto implica que el polígono está más distante que el anterior; consecuentemente, dichos pixels no serán actualizados y las superficies permanecerán ocultos. Al continuar de esta manera, el algoritmo z-buffer puede aceptar polígonos en cualquier secuencia y manejar la intersección de objetos al nivel de pixels.

Desafortunadamente no existe un mecanismo para mantener la opacidad de una superficie, es incapaz de manejar la transparencia. Ya que el programa no retiene datos de la opacidad, por lo que es incapaz de mostrar un nuevo polígono atenuado por la superfice transparente.

A continuación se muestra en el Algoritmo 3.1 algunos fragmentos del método sombreado Z-Buffer.

**Algoritmo 3.1:** mapeo_sombreado

```
#include <math.h>

#define CLAMP(a,min,max) ((a)<(min)?(min):((a)>(max)?(max):(a)))

static float ResFactor = 3.0;
static float TamMinimo = 2.0;

static int NumEjemplos = 16;
static int EjemplosMin = 3;

static float Bias0 = 0.03;
static float Bias1 = 0.04;

float MapSombra(s_map,s,t,z,sres,tres)
struct mapasombra *s_map;
float s,t,z,sres,tres;
{
      int ns,nt,sombrainterna,i,j;
      float bias,ds,dt,js,jt,smin,tmin;
      int iu,iv;
      float Rand(),xresb2,yresb2;

      if (z<s_map->smin)
            return(0.0);

      xresb2 = (float)s_map->xres / 2.0;
      yresb2 = (float)s_map->yres / 2.0;

      s = s * xresb2 + xresb2;
      t = t * xresb2 + yresb2;

      sres = xresb2 * sres * ResFactor;
      tres = xresb2 * tres * ResFactor;

      if (sres < TamMinimo)
            sres = TamMinimo;
      if (tres < TamMinimo)
            tres = TamMinimo;

      if (s < (s_map->bl_map_x - sres - 1) ||
            s > (s_map->tr_map_x + sres + 1) ||
            s > (s_map->bl_map_y - tres - 1) ||
            s > (s_map->tr_map_y + tres + 1))
            return(0.0);

      if ((sres * tres * 4.0) < NumEjemplos)
```

```
        }
    else
        {
         ns = sqrt(tres * NumEjemplos / sres) + 0.5;
         ns = CLAMP(nt,EjemplosMin,NumEjemplos);
         nt = ((float)NumEjemplos) / nt + 0.5;
         nt = CLAMP(ns,EjemplosMin,NumEjemplos);
        }

    ds = 2.0 * sres / ns;
    dt = 2.0 * tres / nt;
    js = ds * 0.5;

    jt = dt * 0.5;
    smin = s - sres + js;
    tmin = t - tres + jt;

    sombrainterna = 0;
    for (i = 0,s = smin; i<ns;i++, s += ds)
        for (j = 0,t = tmin; j<nt;j++, t += dt)
            {
             iu = s + Rand() * js;
             iv = t + Rand() * jt;
             iu = s;
             iv =t;
             bias = Rand() * s_map->Bias1 + s_map->Bias0;
             if (iu>=s_map->bl_map_x && iu<=s_map->tr_map_x
                 && iv >= s_map->bl_map_y
                 && iv <= s_map->tr_map_y)
                 if (z>(s_map->map[s_map->map_xres *
                       (iv - s_map->bl_map_y) +
                       (iu - s_map->bl_map_x)] + bias))
                       sombrainterna ++;
            }
    return((float)sombrainterna / (ns * nt));
}
```

El Algoritmo 3.1 convierte las coordenadas de un espacio en un cuadro de la sombra en el plano (s,t), donde s, t pertenecen [-1,1], dados los límites del área subtendida de las imágenes 'sres', 'tres' que están en [0,1] dentro del espacio de coordenadas del mapa de sombras.

Se dice que un punto no puede estar dentro de la sombra si éste está frente a todos los puntos en el mapa de sombras o si se encuentra fuera de los límites de la sombra Z-buffer. Después de comprobar las condiciones anteriores y la ejecución de la transformación de las coordenadas, se determinan los parámetros necesarios para el mapa de sombras. Los valores ns y nt son el número de muestras en las direcciones horizontal y vertical, se calculan de la siguiente forma:

$$ns*nt = NumEjemplos$$

y el cáalculo del rango de ambas direcciones será el mismo para ambos:

$$nt/tres = ns/sres$$

lo que resulta en:

$$nt = (tres*NumEjemplos/sres)^{1/2}$$

ns = NumEjemplos/nt

La muestra de la sombra Z-buffer varía estocásticamente usando ondulaciones en dos formas:

1. En el plano de la sombra Z-buffer por una cantidad (js,jt). Esto, permite reducir la filtracón del área subtendida en el dominio de la sombra Z-buffer por un pixel de la pantalla.

2. Si se establecen valores de z'. Esto moverá la superficie lentamente más cerca a la fuente de luz. Esto evita que una superficie tenga una mala sombra, y si esto sucede se producirá un valor z' que será más grande que el valor almacenado en el Z-buffer, ya que este valor se encuentra en un punto cercano sobre la superficie. Esto crea pequeños puntos grises que aparecen sobre toda la superficie y que no son parte de la sombra, estos puntos son llamados acné. Los valores de los puntos son números aleatorios entre el rango [Bias0,Bias1].

## 3.4.5 A-BUFFER

El algoritmo A-Buffer (Carpenter, 1984) desarrollado por Lucasfil Ltd., obtiene su nombre de la habilidad de representar una imágen con anti-aliasing usando una técnica de acumulación de área promediada. Lo importante del algoritmo es el concepto del bit enmascarado asociado con un pixel, que es mantenido para la línea actual; este registra el impacto que un polígono ha tenido sobre cualquier pixel en este raster, y es usado para controlar el color final de un pixel que contiene uno o más polígonos parcialmente enmascarados.

El algoritmo opera primero identificando aquellas superficies que impactan sobre el raster actual y, toma una en cualquier secuencia, mantiene un registro de cual pixel ha sido parcial o completamente enmascarado. El Algoritmo 3.2 regresa un bitmask dada la lista de vértices y las coordenadas del espacio en la pantalla (x,y) de la parte inferior izquierda del pixel.

```
Algoritmo 3.2: Mask()
#include "masktable.h"
unsigned mask(list_vert,nu_vert,x,y)
point *list_vert;
int nu_vert,x,y;
{
      int i;
      point *vertex;
      unsigned int maskno = 0;
      int p,q,r,s;

      vértice = list_vert + nu_vert - 1;
      x *= 8;
      y *= 4;
      r = vertex->x * 8 + 0.5;
      s = vertex->y * 4 + 0.5;
      q = 45*(r - x + 9 * (s - y));

      for (i=0; i<nu_vert; i++) {
            vértice = list_vert + i;
            r = vertex->x * 8 + 0.5;
            s = vertex->y * 4 + 0.5;
            q = r - x + 9 * (s - y);
            maskno ^= masktable[p + q];
```

```
            q = 45*p;
        }
        retur(maskno);
}
```

### 3.4.6 ZZ-BUFFER

Particiona la imágen a una matríz rectangular de celdas, donde cada celda tiene la habilidad de retener una lista de cada polígono que toca un pixel dentro de su dominio, y por cuanto. La lista registra la opacidad del polígono y además puede manejar muchas capas de superficies transparentes. Por ejemplo, se dice que una celda en particular mantiene una lista de superficies transparentes, y el siguiente polígono representado cubre completamente la celda, esto significa que la lista actual no es más larga que cualquiera usada y puede volver a empezar con el último polígono. Eventualmente, cuando cada polígono ha sido procesado, las listas pueden ser interpretadas para mostrar la imágen final y, debido a la independencia espacial de las celdas, pueden ser usados para un medio ambiente de multi-procesador.

## 3.5 RAY TRACING

El ray tracing simula alguna propiedad óptica de la luz para determinar la intensidad del pixel del frame store. De hecho, el algoritmo básico es solamente capáz de determinar los valores del color para un pixel, lo que significa que el programa se debe ejecutar unos cientos de veces para construir la imágen. Esto tiene ventajas en que se presta asímismo para correr en múltiples procesadores, pero implica que la base de datos tiene que ser replicada para cada procesador.

La acción del ray tracing puede ser apreciado en una matríz de agujeros como se muestra en la Figura 3.9 para representar las posiciones de los pixels de una gráfica desplegada en pantalla. Se puede ver la cámara dirigida a lo largo del eje-z al recibir rayos de luz a través de agujeros, y como son suficientes, y están muy cercanos, entonces la luz incidente a través de los agujeros puede ser usada para fijar la intensidad de los pixels en el frame store.

Al seleccionar cualquier agujero correspondiente a un píxel, es posible trazar en el espacio del objeto el trayecto geométrico del rayo de luz que entra; de hecho, si se convierte este rayo en un vector, es posible probar para una intersección con una superficie en la base de datos. Puede ocurrir una de las siguientes tres cosas: primero, que hayan varias intersecciones con objetos; segundo, no hayan intersecciones, pero si una con alguna fuente de luz; tercera, no hayan intersecciones de ninguna clase.

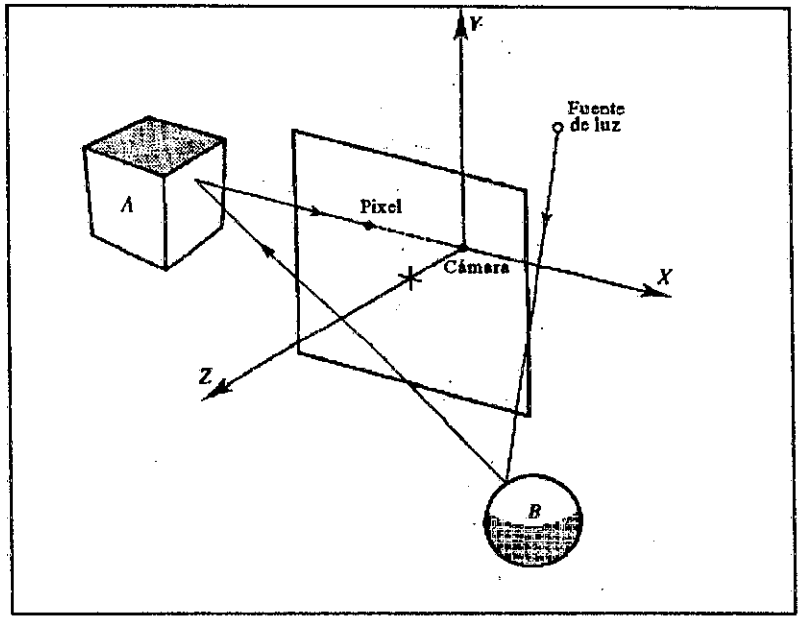



Figura 3.10: En el trayecto del rayo, se forma una imágen al trazar los orígenes ópticos de los rayos que pasan a través de los puntos de los pixels. En este ejemplo, el rayo visto por la cámara tiene una fuente de luz, reflejada a un objeto *B*, y es reflejada al objeto *A*. Así la cámara registrará la escena donde *A* refleja una imágen de *B*, la cual refleja la fuente de luz original.

A continuación se muestra en el Algoritmo 3.2 un simple Ray Tracing en forma recursiva.

**Algoritmo 3.2** TraceRay

```
#include "Types.h"

void T r a c e R a y(inicio,dirección,intensidad,color)
point inicio,dirección;
int intensidad;
colours *color;
{
      int ray_hit();
      point hit_point,direc_reflejo,direc_trans;
      colours color_local,color_reflejado,color_trans;
      object hit_object;

      if (intensidad > MAXDEPTH) *color = BLACK;
      else
            {
             /* Intersección del rayo con todos los objetos y
                encuentra el punto de intersección (si lo hay)
                que se encuentra lo más cercano al inicio del rayo*/

             if (ray_hit(inicio,dirección,&hit_object,&hit_point))
                  {
                   /*definición del color local en un punto de
                     intersección*/
                   sombrear(hit_object,hit_point,&color_local);

                   /*calculo de la dirección del rayo reflejado*/
                   calcular_reflección(hit_object,hit_point,
                                       &direc_reflejo);
                   calcular_transmisión(hit_object,hit_point,
                                       &direc_trans);

                   TraceRay(hit_point,direc_reflejo,intensidad+1,
                            &color_reflejado);
                   TraceRay(hit_point,direc_trans,intensida+1,
                            &color_trans);

                   /*combinación de colores de acuerdo a las
                     propiedades de la superficie del hit_object*/
                   Combinar(hit_object,color_local,color_reflejado,
                            color_trans,color);
                  }
             else *color = BACKGROUND_COLOUR;
             }
      }
```

# CAPITULO 4

# REALISMO

## INTRODUCCIÓN

El realismo puede ser más realzado por una abundancia de efectos extras que son el resultado de investigaciones hechas durante la década pasada. Tal realismo puede incluir los modelos de espejismo, refracción por objetos transparentes o aún la simulación del humo. En este capítulo, solamente algunos de los temas principales en orden de influencia sobre la fidelidad y realismo de la imagen representada serán considerados.

## 4.1 ANTI-ALIASING

Las técnicas de anti-aliasing se refieren a la minimización de artefactos visuales introducidos por métodos de construcción de imagenes generadas por computadora. Estos artefactos están asociados con los bordes de los polígonos donde la 'escalera' con los pixels dentados aparecen ilustrados en la Figura 4.1, y el despliegue de objetos delgados. Otras anomalías visuales aparecen en la forma de modelos moiré (modelo que simula el efecto de olas) cuando es desplegada una textura detallada, la cual resulta extremadamente molesto cuando la textura es movida. Este último efecto es muy común en la televisión convencional, especialmente en telas que contienen tejidos finos.

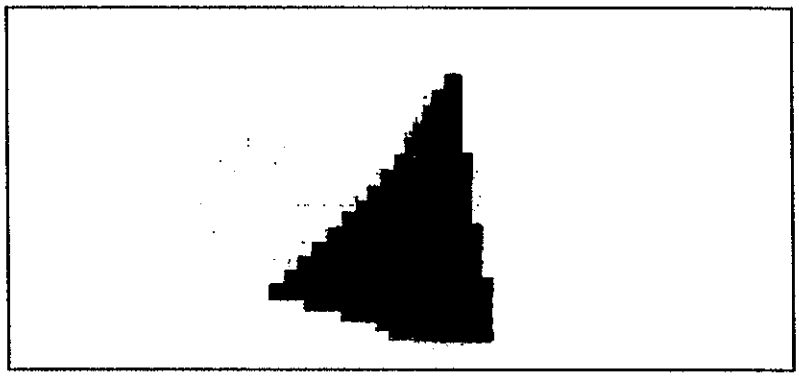


**Figura 4.1:** este diagrama muestra un objeto con orillas cortadas realizadas con polígonos.

Una técnica para reducir el aliasing es incrementar el número de muestras tomadas dentro de un área de pixels; por ejemplo 4, 9, 16 o más muestras de sub-pixels, que son usados para influenciar la intensidad del pixel. Asociando diferentes pesos numéricos con estas muestras, la intensidad puede ser relacionada, a través de varias funciones, para el enmascarado del pixel por un polígono. Cuando un número de polígonos tocan el mismo pixel, el color final puede ser calculado usando esta técnica. Los pesos asociados con las muestras de pixels son derivados de formas como un triángulo, caja, Catmull-Room, seno y Gaussiana, como se ilustra en la Figura 4.2.

Lo expresado anteriormente es el resultado del muestreo en el espacio de imagenes para posiciones discretas; este efecto es referido como 'aliasing espacial'. Otra forma de aliasing, conocido en la animación como 'aliasing temporal', surge cuando un objeto en movimiento es muestreado en diferentes

puntos en el tiempo, y crea efectos como rotación de ruedas en dirección contraria al incremento de la velocidad.

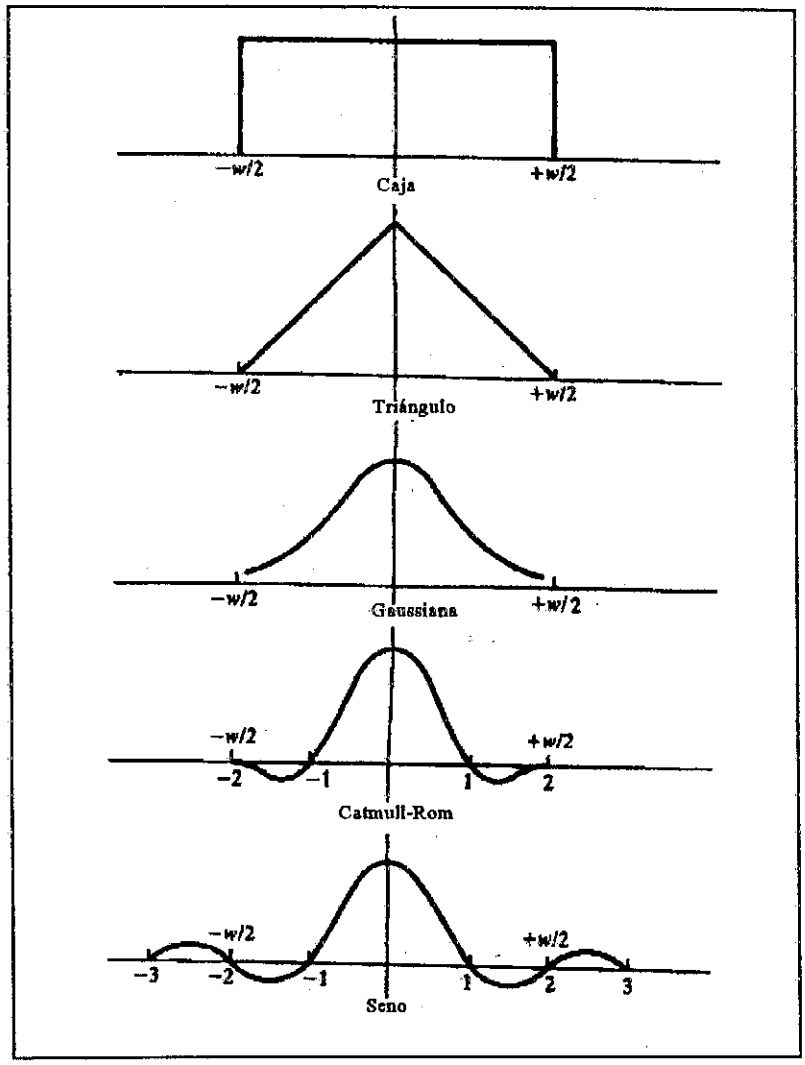



**Figura 4.2:** estas cinco funciones ilustran las diferentes ponderaciones que pueden estar en operación sobre uno o más pixels.

# 4.2 MAPEO DE TEXTURA

Un método simple para aumentar el realismo a las imagenes es cubrir modelos 3-D con textura detallada, quizás derivada de alguna fuente externa. La textura es normalmente mantenida como una matriz de pixels conocidos como un mapa de textura, que puede ser de 24 bits para imagenes totalmente a color. El mapa es direccionado usando coordenadas en la misma forma que en un área de 2-D, y evitar confusión con las coordenadas espaciales, los nombres asignados para los ejes vertical y horizontal son *u* y *v*, por lo que el término 'coordenadas-uv' es muy utilizado en el mapeo de textura.

El mapeo de la textura sobre la superficie puede ser realizado en diferentes formas. Depende mucho de la naturaleza de la superficie geométrica del objeto.

Por ejemplo, si un polígono planar será mapeado con textura, entonces de

alguna forma los pixels asociados con este polígono deben ser asignados a intensidades de color de un mapa de texturas. No obstante, porque el polígono puede tener una orientación arbitraria, diferentes partes de la superficie obtendrán diferentes porciones de textura. La Figura 4.3 muestra un mapa de texturas direccionado con coordenadas-*uv*, y el otro diagrama ilustra como la huella del pixel es proyectada hacia atrás sobre el polígono. El polígono también referencia un conjunto de ejes para controlar la orientación de la textura. Ahora si el mapa de texturas está cubriendo totalmente el polígono, se necesita mapear las trazas del pixel dentro del espacio de textura. Esto es posible porque el polígono proporciona la geometría 3D necesaria contenida en los valores de los vértices y su superficie normal. La siguiente etapa identifica aquellos elementos del mapa de texturas cubiertos por la huella.

Existen tres condiciones que deben ser tomadas en cuenta para la representación. Primero, si el polígono está muy cerca de la cámara, el trazo del pixel podría ser muy pequeño que el elemento de textura (texel); segundo, como el polígono retrocede, la forma del trazo estará cubierto de dos texels; y tercero, cuando el polígono se mueve, el trazo podrá cubrir el mapa de textura totalmente. Obviamente, una representación debe ser capaz de manejar estos tres casos.

Para mapear la textura en una superficie plana, se necesita un mecanismo para alinear la textura con la superfice; esto puede ser realizado al asociar un eje con la superficie.

Segundo, es necesario un punto sobre la superficie para preparar el punto T(0,0) del mapa de texturas. Y tercero, un factor escalar $k$ para controlar como la escala de la textura relaciona el tamaño de la superficie.

La Figura 4.4 ilustra una superficie arbitraria que será mapeada con la textura desde el mapa de texturas por $u$ y $v$. El vector $\boldsymbol{S}$ sobre la superficie se alineará con el eje-$u$ de la textura, y el punto $P_0(x_0,y_0,z_0)$ se alineará con T(0,0) sobre la textura. Si existen las siguientes condiciones:

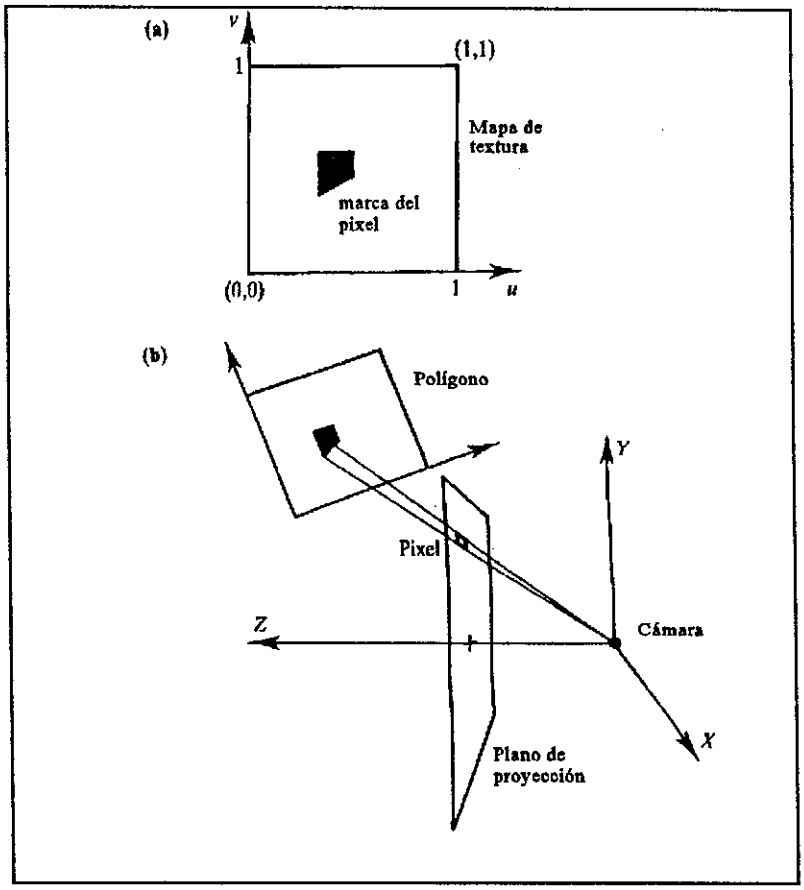



**Figura 4.3:** un mapa de texturas es direccionado usando coordenadas-*uv*, para distinguirlos de las coordenadas-*xyz* asociadas con el espacio e descripciones 3-D. En este diagrama un trazo del pixel es proyectado hacia atrás sobre el polígono, el cual resulta mapeado en el espacio de textura. Este valor del trazo del pixel en el mapa de texturas es asignado al pixel.

$\mathbf{N}$ es un vector unitario normal para la superficie plana
$\mathbf{S}$ es un vector unitario ortogonal a $\mathbf{N}$
$P_0(x_0, y_0, z_0)$ es un punto de referencia sobre el plano mapeado por T(0,0)
$P_1(x_1, y_1, z_1)$ es un punto a ser mapeado dentro de $T(u,v)$
$k$ es un término escalar para la textura

entonces

$$\mathbf{V}_1 = \mathbf{V}_0 + \mathbf{V}$$

donde

$$\mathbf{V} = \begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \\ z_1 - z_0 \end{bmatrix}$$

Entonces

$$\mathbf{V} \cdot \mathbf{S} = |\mathbf{V}||\mathbf{S}|\cos(\beta)$$

pero

$$ku/|\mathbf{V}| = \cos(\beta)$$

Y como $|\mathbf{S}| = 1$

$$\mathbf{V} \cdot \mathbf{S} = uk$$

además

$$u = \mathbf{V} \cdot \mathbf{S}/k$$

Similarmente,

$$v = \mathbf{V} \cdot \mathbf{T}/k$$

donde el vector $\mathbf{T}$ está dado por el producto cruz de $\mathbf{N}$ y $\mathbf{S}$ y es igual a:

$$\mathbf{T} = \begin{bmatrix} N_2S_3 - N_3S_2 \\ N_3S_1 - N_1S_3 \\ N_1S_2 - N_2S_1 \end{bmatrix}$$

Esta técnica sólo relaciona el mapeo de puntos simples sobre el plano para el mapa de texturas, y no considera las estrategias anti-aliasing que deben ser implementadas cuando es evaludado el trazo del pixel.

La proyección esférica es muy útil en mapeo de texturas especialmente para crear planetas artificiales.

La Figura 4.5 muestra un mapa de texturas direccionado por coordenadas-*uv*, y una esfera de radio *r* direccionado por las coordenadas de latitud y longitud. Cuando el trazo del pixel es proyectado sobre la esfera es necesario calcular las coordenadas-*uv* para las cuatro esquinas del pixel para determinar su color. Para

esto, se necesitan funciones que conviertan cualquier punto sobre la esfera en un punto sobre el mapa de texturas. Obviamente la textura se distorsionará cuando sea mapeada sobre la esfera, todos los texel que tienen coordenada-v 0 están mapeados en el polo sur, mientras que los texel que tienen coordenada-v 1 están mapeadas sobre el polo norte.

Para empezar, la esfera necesita una posición y orientación, para ello están los siguientes parámetros:

$\mathbf{n}_e$ es un vector unitario normal para un punto de referencia cero sobre el Ecuador.

$\mathbf{n}_p$ es un vector unitario ortogonal para $\mathbf{n}_e$ y normal al polo norte.

$P_i$ es el punto $(x_i, y_i, z_i)$ a ser mapeado por $T(u,v)$

$\mathbf{n}_i$ es un vector unitario normal a $P_i$

$P_c$ es el centro de la esfera $(x_c, y_c, z_c)$

$r$ es el radio de la esfera.

Entonces

$$\mathbf{n}_i = \begin{bmatrix} (x_i - x_c)/r \\ (y_i - y_c)/r \\ (z_i - z_c)/r \end{bmatrix}$$

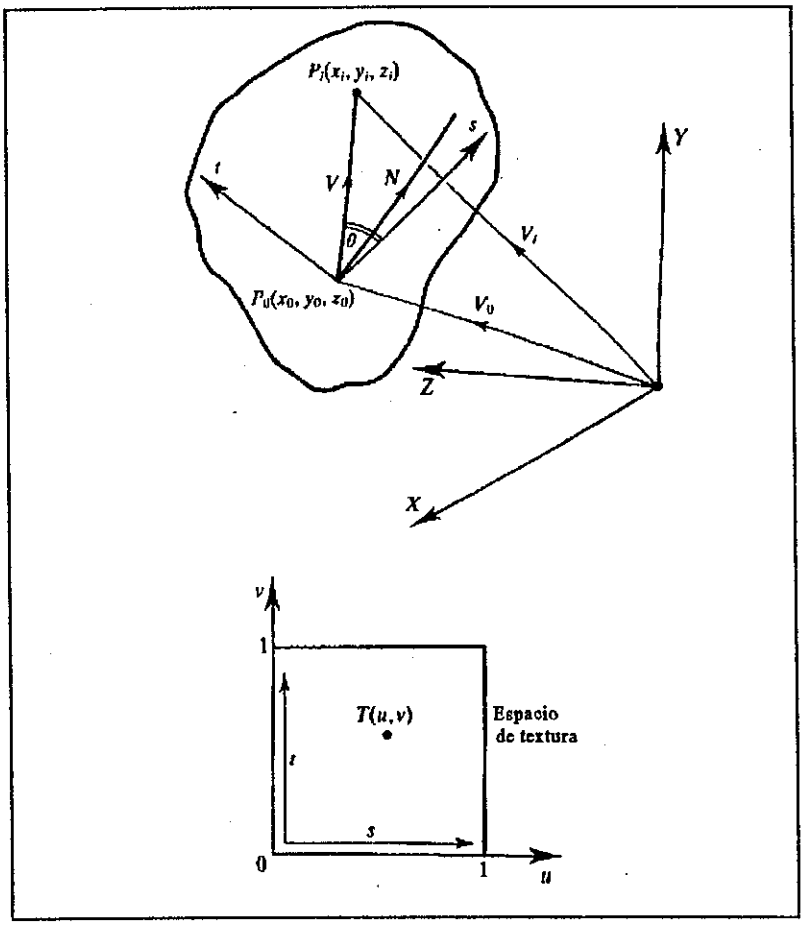



**Figura 4.4**: cuando la textura es mapeada sobre un plano arbitrario, una función de mapeo es necesaria para relacionar el punto $P_i(x_i, y_i, z_i)$ a $T(u,v)$.

**Cálculo de v (latitud)** Como el polo sur ($-\mathbf{n}_p$) será mapeado para $v = 0$, se necesita calcular el ángulo de latitud entre $\mathbf{n}_i$ y el polo sur, el cual está dado por:

$$ß = \cos^{-1}(-\mathbf{n}_p \cdot \mathbf{n}_i),$$

además

$$v = ß/pi$$

Si $v = 0$ (el polo sur) o $v = 1$ (el polo norte) no hay necesidad para calcular u; este no puede estar en cero, y así termina el problema.

**Cálculo u (longitud)** El ángulo de longitud ß está dado por:

$$ß = \cos^{-1}((\mathbf{n}_e \cdot \mathbf{n}_i)/\text{sen}(ß))$$

y si $ß = \pi/2$ (el Ecuador), entonces ß resulta:

$$ß = \cos^{-1}(\mathbf{n}_e \cdot \mathbf{n}_i)$$

Cuando $\mathbf{n}_e = \mathbf{n}_i$ entonces $ß = 0$ o $2\pi$, lo que implica que ß tendrá que ser multiplicado por $1/2\pi$.

No obstante, hay otro problema que corresponde a la posición relativa de $\mathbf{n}_i$ al plano que contiene $\mathbf{n}_e$ y $\mathbf{n}_p$. Esto se puede resolver al realizar la operación cruzada de $\mathbf{n}_p$ y $\mathbf{n}_e$ (Figura 4.5), y cualcular la operación punto con $\mathbf{n}_i$. Si este ángulo excede de cero, está sobre el mismo lado como el vector normal, de otra forma, se encuentra en el lado opuesto.

Así:

$u = ß/2\pi$ si $(\mathbf{n}_p \times \mathbf{n}_e) \cdot \mathbf{n}_i > 0$ de otra forma

$u = 1 - ß/2\pi$

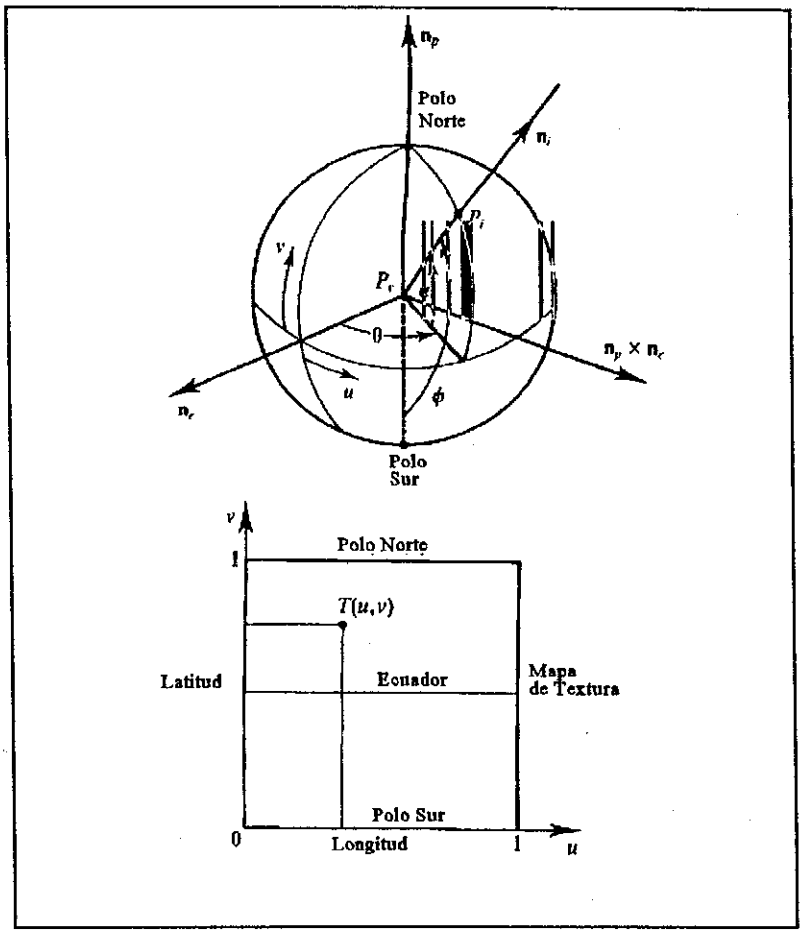



**Figura 4.5**: cuando la textura es mapeada sobre la esfera, es necesaria una función de mapeo para relacionar un punto $P_i(x_i,y_i,z_i)$ para $T(u,v)$.

El Algoritmo 4.1 de mapeo de textura, muestra el proceso de proyección a lo largo de un polígono cuando este es cortado dentro del cuadro. Los vértices del polígono son transformados en un objeto en el espacio, luego la función de proyección es aplicada a estos vértices transformados a coordenadas u,v, entonces el algoritmo pasa una lista (vertex_list), de número de vértices (vertex_no) del polígono y la función uvproject() que muestra las coordenadas u,v sobre la superficie. La función transform(fromspace,tospace,ptfrom,ptt) transforma un punto ptfrom en el espacio fromspace a un punto ptto en el espacio tospace. La función uvproyect(point,u,v) regresa las coordenadas de un punto dado point. El vértice es transformado de un espacio en pantalla a un espacio, en el cual es invocada la función de proyección. El valor promedio y de compresión de las coordenadas u,v son calculados y pasados a la función mip_map().

**Algoritmo 4.1**: texture_map.c

```
#include "types.h"

/* Esta función regresa el color desde el polígono mapa de
   textura texture_map */

void texture_map(map, vertex_no, vertex_list, uvproject, color)
char map[1024][1024];   /* para el mip_map */
```

```
int vertex_no;       /* número de vértices que conforman un polígono */
point *vertex_list;      /* lista de vértices */
void (*uvproject();      /* función de proyección que muestra las
                            coordenads u,v sobre un objeto */
float color[3];

{
      point *screen_vertex,object_vertex;
      float last_u,last_v,u,v,uav,vav,d,texture_area=0;
      int i;

  /* Se obtienen las coordenadas u,v del último vértice en la lista */
      screen_vertex = vertex_list + vertex_no - 1;
      transform("screen,"object",screen_vertex,&object_vertex);
      uvproject(objetc_vertex,&last_u,&last_v);

      for (i=0;i<vertex_no;i++) {
      /* Calculo del área del espacio de textura, sumando las áreas
               del trapecio bajo cada lado */

            screen_vertex = vertex_list + i;
            transform("screen,"object",screen_vertex,&object_vertex);
            uvproject(object_vertex,&u,&v);
            texture_area += (u + last_u) * (v - last_v);
            uav += u;
            vav += v;
            last_u = u;
            last_v = v;
      }
      texture_map *= 0.5;
      uav /= vertex_no;
      vav /= vertex_no;
      d = sqrt(fabs(texture_area)); /* Aproximación de la compresión */
      mip_map(map,uav,vav,d,color);
}
```

# 4.3 BUMP MAPPING

Este método fue desarrollado por James Blinn (Blinn, 1978c), e incrementa el realismo introduciendo un modelo de efecto de pseudo-superficie que da una apariencia que la superficie está cubierta con una textura de superficie 'bumpy', que es como cuero o concreto. El bump map es introducido en el sistema de computación de una fuente de video o la salida de un programa de coloreado. Durante la etapa de interpretación, los niveles de intensidad de esta imagen son usados para desequilibrar la superficie normal cuando se hace el cálculo de la luz especular. Esto es el resultado de una superficie que aparece cubierta por una textura moteada la cual crea luz y sombras, y realza el efecto 'bumpy'. La Lámina 1 ilustra el nivel de detalle que puede ser logrado usando mapeo de texturas. (ver apéndice).

# 4.4 SOMBRAS

Las sombras son encontradas donde haya fuentes de luz y objetos, y proporciona una señal visual importante para ayudar a la interpretación física del mundo 3-D. La falta de sombras en muchas imagenes generadas por computadora

es la razón por la cual dichas imagenes no son convincentes; es difícil determinar cuándo un objeto está en el suelo o flotando sobre él, porque no existe sombra.

La razón de este descuido es debido al tiempo necesario para su cálculo, especialmente cuando la escena es compleja y existen muchas fuentes de luz activas. No obstante, se ha visto que el trayecto del rayo puede ocasionar algunas sombras, y la radiocidad desarrolla zonas de sombra causados por objetos muy próximos. Estas técnicas no siempre están disponibles, aunque, a veces no proporcionan los tipos de imagenes necesarios para el animador. Consecuentemente, los algoritmos de sombreado han sido desarrollados para agregar efectos que pueden ser incorporados con otros algoritmos de interpretación de imagenes.

Considere la siguiente solución para el cálculo de sombras. La Figura 4.6 muestra una escena donde una simple fuente de luz ilumina un objeto que proyecta una sombra sobre el suelo. Si se imagina estar colocado en la fuente de luz, resulta posible construir un mapa de sombra de superficies visibles desde este punto de vista. Desde la posición de la cámara, se puede investigar cada pixel de la escena para ver como está iluminada o sombreada. Esto puede ser descubierto seleccionando un pixel y proyectar su trazo sobre la superficie más cercana. Esta posición en el espacio es probado para ver si está visible usando el mapa de intensidades de la fuente de luz: si no lo está, entonces el pixel está en estado de iluminación o de sombra. Esta técnica es usada para hacer los cálculos de sombreado más rápido. Esta técnica es simple de implementar; solo falla si la cámara está demasiado cercana a los objetos.

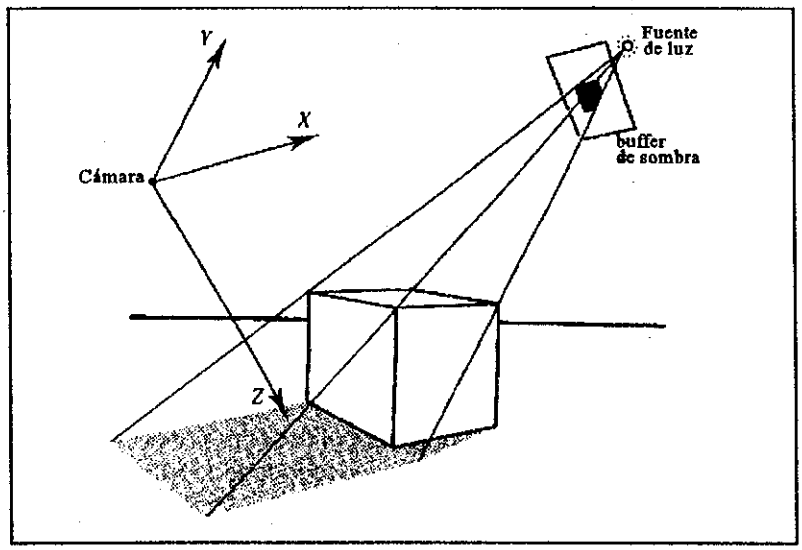



Figura 4.6

# 4.5 MOTION BLUR

El motion blur es una técnica más conocida, y es usada frecuentemente por fotógrafos. Un ejemplo para ilustrar esta técnica, es que si se tiene un carro, que viaja a 50 kph y recorre aproximadamente 0.46 m/seg., si el obturador de la cámara está operando a 0.01 seg., durante este período sólo recorrerá una distancia de 0.446 cm., lo que no sería tan notorio. Si alguien se encuentra agitando los brazos podría exceder la velocidad de 0.46 m/seg., y crea una foto obscura. Esto puede ser visto en películas de cine y televisión, y recientes desarrollos en la tecnología de la fotografía y cine sean producido vídeo cámaras con obturadores rápidos en el orden de 0.001 seg., lo que elimina la imagen

obscura.

En la animación tradicional, donde cada imagen es dibujada, pintada y fotografiada para una proyección de 24 cuadros/seg., lo obscuro no aparecerá en cada imagen estacionaria cuando es fotografiada. Esto significa que el animador desea acentuar el movimiento de un caracter, si se desea que la imagen sea obscura se hará por medio de técnicas de dibujo y pintura.

En la Animación por Computadora, la cámara virtual no tiene una velocidad de obturador físico, registra un estado del espacio en un instante en el tiempo, y además no se podrá desarrollar lo obscuro de la imagen. Consecuentemente, cuando estas imagenes son animadas, se pierde el sentido del movimiento asociado con la animación en cine y video. Para saber las desventajas, las técnicas han sido desarrolladas según el cual el movimiento obscuro es deliberadamente introducido durante la representación. Una técnica desarrollada por Andrew Glassner (Glassner, 1988) usa 'spacetime ray tracing' para obscurecer objetos en movimiento. La interpretación está basada en el algoritmo del trayecto del rayo para crear un volumen en 3D, y recorrer un objeto en movimiento; esto es muestreado con rayos extra sobre pequeños períodos de tiempo para simular un obturador abierto.

# 4.6 INTENSIDAD DE CAMPO

La intensidad de campo es otro atributo asociado con la tecnología de cámara, pero asociado con el sistema de lentes de la cámara; aquí el lente tiene que estar enfocado sobre un objeto para obtener una imagen nítida. Aunque algunas cámaras utilizan lentes de enfoque fijo, que sólo es capaz de realizar imagenes nítidas sobre una distancia útil para la mayoría de las imagenes tomadas.

La intensidad de campo de unos lentes describe la distancia en el espacio del objeto sobre la cual la imagen enfocada está razonablemente nítida. Es una función de la abertura del lente incrementar la intensidad del campo. Esta característica es usada por el fotógrafo para aislar objetos dentro de una escena para mantenerlos nítidos en lugar de obscuros o borrosos (fuera de enfoque).

Se puede crear un efecto de intensidad de campo simulando la acción de los lentes que enfocan un punto en el espacio del objeto sobre una pantalla de imagenes, donde se satisface la siguiente relación:

$$1/d_o + 1/d_i = 1/f,$$

donde $d_o$ es la distancia del objeto, $d_i$ es la distancia de la imagen, y $f$ es la longitud focal de los lentes. Aunque, si $f$ es constante, entonces la diferencia de valores de $d_o$ será necesario para complementar los valores de $d_i$ para satisfacer la ecuación de la lente. Pero como la distancia de la pantalla de imagenes es fija, los puntos en el espacio del objeto enfocarán perfectamente sobre la pantalla de imagenes o crear círculos de confusión.

Potmesil y Chakravarty (Potmesil, 1982) proponen que el diámetro del círculo de confusión $C_d$ puede ser calculado por la siguiente fórmula:

$$C_d = (1 - d/d_i)f/n,$$

donde $d$ es la distancia del plano de la imagen desde el lente y $n$ es el número de abertura. Los valores de $f$ y $d_i$ están en la ecuación de la lente. Cuando es cálculado el diámetro del círculo de confusión, la intensidad de luz para este punto muestra, es propagado alrededor del pixel. Por ejemplo, si el diámetro del círculo cubre un cuadro de 3 x 3 de pixels, las intensidades de luz podrían ser distribuidas como se muestra en la Figura 4.7

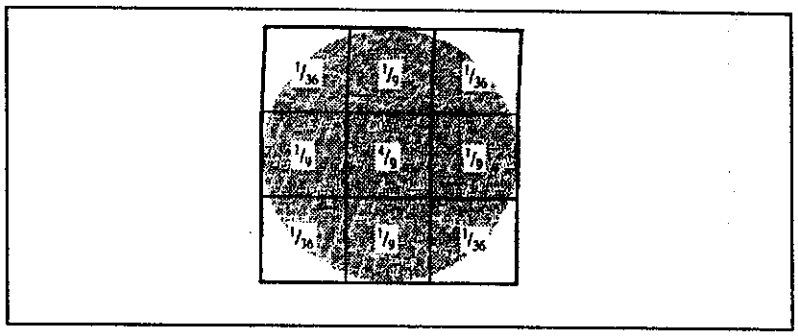



**Figura 4.7**: si un círculo de confusión cubre una matriz de pixels de 3 x 3, las intensidades pueden ser distribuidas como se muestra en la figura.

# 4.7 TEXTURAS SÓLIDAS

Una desventaja en el mapeo de textura es la distorsión que ocurre cuando está envuelta alrededor de una superficie compleja, lo que resulta en modelos que podrían permanecer espacialmente consistentes, resultando estiradas. La textura sólida resulta de este problema definiendo el modelo de textura en términos de funciones matemáticas o procedimientos que definen que textura existe en cualquier punto del espacio 3-D. Naturalmente, la técnica no puede ser usada para mapear fotografías y otras imagenes sobre objetos, pero sí es efectiva para crear texturas de superficie como de mármol, piedra, fibra, modelos abstractos y compuestos.

El algoritmo ilustra mejor si se describe cómo un objeto podría ser cubierto con puntos o lunares de color aleatorios. Para empezar, es necesario un mecanismo para localizar la posición y el tamaño de los puntos a través de un volumen de espacio. Esto se puede lograr con una tabla de valores-*xyz* aleatorios que referencien el radio localizado en este punto. Cuando es representada, la superficie geométrica proporciona el lugar 3-D preciso visto como un pixel, que es usado para buscar la tabla y ver cuál pixel está cubierto por un punto a color o por un color de fondo. Esto es mostrado en la Figura 4.8. No importa qué complejo sea el objeto, la naturaleza volumétrica de la textura asegura que los puntos aparecen correctamente distribuidos sobre la superficie.

# 4.8 MAPEO DEL MEDIO AMBIENTE

Aunque los reflejos de las superficies púlidas pueden ser distorsionados, son vitales para corregir la representación de objetos cromados o plateados como cuchillos y tenedores, y otros objetos pulidos. El trazado del rayo se puede calcular con reflexiones ópticos precisos. Una tecnología alternativa es usar mapas de medio ambiente, que ayudan a modelar el mundo alrededor de un objeto pulido para crear reflejos visualmente aceptables.

Para ilustrar esta técnica, hay que imaginar que una cuchara tiene que ser animada con imagenes de video tomadas en una cocina real. Se podrían ver diferentes reflejos en la cocina, como en la cuchara, pero más que el modelo completo de la cocina dentro de la computadora, un mapa del medio ambiente puede ser creado para que realice esta operación. El mapa consiste de seis fotografías tomadas desde el centro de la cocina con la cámara dirigida hacia las cuatro paredes, el techo y el piso. La cámara debe estar fija con un amplio ángulo de lente para capturar cada superficie. Si estas seis imagenes son ingresadas en el sistema de animación y direccionadas para encerrar el espacio usado por la cuchara, la técnica de 'ray-casting' puede determinar que parte del mapa es vista por el observador. La Lámina 2 muestra una escena representada usando esta técnica. (ver apéndice)

Cuando un modelo tiene que reflejar otros modelos sintéticos dentro del sistema de animación, el medio ambiente deberá ser construido a base de datos fuente. Esto implica capturar los mapas desde el centro del objeto reflejando el medio ambiente, aunque el objeto no es parte de esta operación.

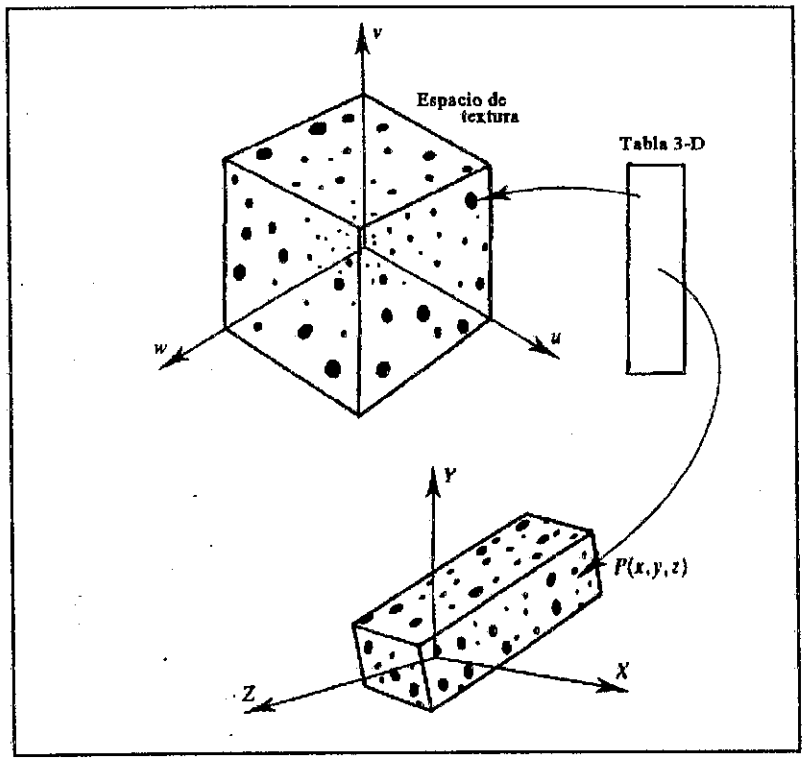



**Figura 4.9**: en este ejemplo, una tabla localiza la posición y tamaño esférica y permanece como una unidad de volumen de espacio. Cuando un objeto es representado con textura sólida, los puntos sobre su superficie son usados para accesar la tabla para descubrir si es cubierto por un punto o por un color de fondo. Si el objeto tiene partes cortadas, la superficie expuesta revelará un efecto de textura sólida continua.

Existe una técnica llamada mapeo de cromado, que utiliza un mip-mapping parecido a la técnica anti-aliasing; esta técnica requiere de una pequeña región -un área de textura- que será trazada fuera del espacio u-v. Esto se hace pixel por pixel por medio del Algoritmo 4.2, el cual se usa para reflejar vectores del polígono que va a ser mapeado. Los vectores reflejados forman un haz que se proyecta al interior de una superficie cerrada. El valor del polígono es pasado a la rutina como una lista de vértices (list_vert) del número de vértices (nu_vert) y la lista de normales (nor_vert). A continuación, se detalla el algoritmo 4.2 llamado Map_cromo().

Algoritmo 4.2 Map_cromo()

```
#include "types.h"

void normalizar();
void reflejar();
void cromo();

/* Map_cromo(): regresa el color del polígono mapeado */

void map_cromo(map,nu_vert,list_vert,list_normal,color)
char mapa[1024][1024];          /* mip map */
int nu_vert;              /* número de vértices que forman el polígono */
point *list_vert;         /* lista de vértices */
point *list_normal;       /* lista de normales */
float color[3];
{
      point *vert_pantalla,vector_ojo,*normal,R;
      float ult_u,ult_v,u,v,uav,vav,d;
      float uv_long_borde,compresion_borde = -1;
      int i;
```

```
        vert_pantalla = list_vert + nu_vert - 1;
        normal = list_normal + nu_vert - 1;
        transformar("pantalla,"camara",vert_pantalla,&vector_ojo);
        vector_ojo.x *= -1;
        vector_ojo.y *= -1;
        vector_ojo *= -1;
        normalizar(&vector_ojo);
        reflejar(vector_ojo,normal,&R);
        cromo(R,&ult_u,ult_v);

        for (i=0; i< nu_vert; i++){

              vert_pantalla = list_vert + i;
              normal = list_normal + i;
              transformar("pantalla,"camara",vert_pantalla,&vector_ojo);
              vector_ojo.x *= -1;
              vector_ojo.y *= -1;
              vector_ojo *= -1;
              normalizar(&vector_ojo);
              reflejar(vector_ojo,normal,&R);
              cromo(R,&ult_u,ult_v);

              uv_long_borde = (u - ult_u)*(u-ult_u) + (v-ult_v)*(v-ult_v);
              if (uv_long_borde > compresion_borde)
                    compresion_borde = uv_long_borde;

              uav += u;
              vav += v;
              ult_u = u;
              ult_v = v;
        }

        uav /= nu_vert;
        vav /= nu_vert;
        compresion_borde = sqrt(compresion_borde);
        mip_map(mapa,uav,v,compresion_borde,color);

}
```

El algoritmo 4.3 muestra el mapeo del ambiente; éste utiliza los mismos argumentos que el Algoritmo 4.2, además del mapa mapa_amb que es de seis mip-maps. Por ejemplo se tienen las caras de un cubo, a las cuales se le asigna un bit a cada una, lo que permite definir los bordes del cubo que será la suma de las dos caras que forman dicho borde. A continuación, se muestra el Algoritmo 4.3 llamado Mapeo_Ambiente().

```
Algoritmo Mapeo_ambiente()

#include "types.h"

#define nu_max 30

/* numeración de las caras y bordes del cubo */

#define cara0 1
#define cara1 2
#define cara3 8
#define cara4 16
#define cara5 32
```

```
#define borde01 3
#define borde02 3
#define borde03 9
#define borde04 17
#define borde12 6
#define borde23 12
#define borde34 24
#define borde41 18
#define borde51 34
#define borde52 36
#define borde53 40
#define borde54 48

void normalizar();
void reflejar();
void obtener_normal_reflejado();
void obtener_intersección_cara();
void obtener_intersección_borde();
void proyectar();

point cubo[nu_max]; /* almacena la proyección del haz reflejado sobre
                       el cubo */
int nu_cubo;       /* almacena el número de puntos que se forman en la
                      proyección*/
float uv[nu_max][2] /* almacena los valores uv de la proyección para una
                       cara dada */

/*mapeo_ambiente();regresa el color del polígono el mapa de ambiente */

void mapeo_ambiente(mapamb,nu_vert,list_vert,list_normal,color)
char mapamb[6][1024][1024];   /*mapa de ambiente que consiste de seis
                                mip-maps*/
int nu_vert;       /*número de vértices que generan el polígono */
point *list_vert;  /*lista de vértices */
point *list_normal; /* lista de normales */
float color[3];
{

      point ult_R,R,pt;
      float ult_u,ult_v,u,v,uav,vav,d,área_textura,área_total;
      int i,j;
      int bit_proyección; /*almacena todas las caras sobre las cuales
                            el haz se refleja */
      int borde,ult_cara,cara_act,cara_proy;
      float c_color[3];
      nu_cubo = 0;
      bit_proyección = 0;
      area_total = 0;

/*encontrar la intersección del vector reflejado desde el último */
      /* vértice en la lista */

      i = nu_vert - 1;
      obtener_normal_reflejada(list_vert+i,list_normal+i,&ultR);
      obtener_intersección_cara(&ultR,&cubo[nu_cubo],&ult_cara);
      nu_cubo++;
      bit_proyección |= ult_cara;
      for (i=0; i< nu_vert; i++) {

            obtener_normal_reflejada(list_vert+i,list_normal+i,&R);
            obtener_intersección_cara(&ultR,&cubo[nu_cubo],&ult_cara);
```

```
        /* si el último rayo reflejado una cara diferente para el */
              /* rayo actual, se encuentra la intersección del haz */
              /* reflejado */

              if (cara_act != ult_cara) {
              borde = cara_act | ult_cara;
              obtener_intersección_borde(&ultR,&R,borde,&cubo[nu_cubo]);
              nu_cubo ++;
              bit_proyección |= cara_act;
              }

              cubo[nu_cubo] = pt;
              nu_cubo++;
              ult_cara = cara_act;
              ultR = R;
        }

        color[0] = color[1] = color[2] = 0;
        for (i=0, cara_proy = 1; i<6; i++,cara_proy+=cara_proy){
              if (cara_proy & bit_proyeccción) {

                    /* obtener la proyección en el espacio uv sobre la */
                    /* cara proyectada */
                    proyectar(cara_proy);
                    ult_u = uv[0][0];
                    ult_v = uv[0][0];
                    uav = 0;
                    vav = 0;
                    area_textura = 0;

                    for (j=1; j<nu_cubo; j++) {

                          u = uv[j][0];
                          v = uv[j][1];
                          area_textura += (u+ult_u) * (ult_v - v)/2;
                          ult_u = u;
                          ult_v = v;
                          uav += u;
                          vav += v;
                    }
                    uav /= (nu_cubo - 1);
                    vav /= (nu_cubo - 1);
                    d = sqrt(fabs(area_textura));
                    mip_map(mapamb[i],uav,vav,d,c_color);

                    color[0] += c_color[0] * area_textura;
                    color[1] += c_color[1[ * area_textura;
                    color[2] += c_color[2] * area_textura;
                    area_total += area_textura;
              }
        }


        /* normalizar los pesos */
        color[0] /= area_total;
        color[1] /= area_total;
        color[2] /= area_total;

}

/*Dado un vert_pantalla y su normal la función obtener_normal_reflejada()*/
```

```
/*regresa el vector reflejado */

void obtener_normal_reflejada(vert_pantalla,normal,wR)
point *vert_pantalla,*normal,*wR;
{
      point vector_ojo,camaraR;

      transformar("pantalla,"camara",vert_pantalla,&vector_ojo);
      vector_ojo.x *= -1;
      vector_ojo.y *= -1;
      vector_ojo *= -1;
      normalizar(&vector_ojo);
      reflejar(vector_ojo,normal,&camaraR);
      transformar_vector("camara,"world",&camaraR,wR);
      normalizar(wR);
}


/*La función obtener_intersección_cara regresala cara del cubo donde */
/*intersecta la normal, y el punto de intersección */

void obtener_intersección_cara(normal,pt,cara)
point *normal,*pt;
int *cara;
{
      point n = *normal;
      float t;

      /* prueba de la dirección nz */
      if (n.z < 0) {    /* pruebar intersección en la cara 5 */
            t = -0.5/n.z;
            pt->x = n.x * t;
            pt->y = n.y * t;
            pt->z = n.z * t;
            if (fabs(pt->x) < 0.5 && fabs (pt->y) < 0.5) {
                  *cara = cara5;
                  return;
            }
      }
      else if (n.z > 0) {     /* pruebar intersección en la cara 5 */
                  t = -0.5/n.z;
                  pt->x = n.x * t;
                  pt->y = n.y * t;
                  pt->z = n.z * t;
                  if (fabs(pt->x) < 0.5 && fabs (pt->y) < 0.5) {
                        *cara = cara5;
                        return;
                  }
            }

      /* prueba de la dirección ny */
      if (n.y < 0) {    /* pruebar intersección en la cara 5 */
            t = -0.5/n.y;
            pt->x = n.x * t;
            pt->y = n.y * t;
            pt->z = n.z * t;
            if (fabs(pt->x) < 0.5 && fabs (pt->z) < 0.5) {
                  *cara = cara4;
                  return;
            }
      }
      else if (n.z > 0) {     /* pruebar intersección en la cara 5 */
```

```
            t = -0.5/n.y;
            pt->x = n.x * t;
            pt->y = n.y * t;
            pt->z = n.z * t;
            if (fabs(pt->x) < 0.5 && fabs (pt->z) < 0.5) {
                  *cara = cara2;
                  return;
            }
      }

/* prueba de la dirección nx */
if (n.x < 0) {    /* pruebar intersección en la cara 5 */
      t = -0.5/n.x;
      pt->x = n.x * t;
      pt->y = n.y * t;
      pt->z = n.z * t;
      if (fabs(pt->y) < 0.5 && fabs (pt->z) < 0.5) {
            *cara = cara3;
            return;
      }
}
else if (n.z > 0) {     /* pruebar intersección en la cara 5 */
            t = -0.5/n.x;
            pt->x = n.x * t;
            pt->y = n.y * t;
            pt->z = n.z * t;
            if (fabs(pt->y) < 0.5 && fabs (pt->z) < 0.5) {
                  *cara = cara1;
                  return;
            }
      }


/* la función obtener_intersección_borde regresa la intersección*/
/* del borde del cubo con el rayo reflejado */

void obtener_interseccion_borde(n1,n2,borde,pt)
point *n1,*n2,*pt;
int borde;
{
      float a,b,c;
      float x0,y0,z0,f,g,h;
      float denom,t;

/* obtener el plano eqn: ax+by+cz=0 de dos normales n1 y n2*/
      a = n1->y*n2->z - n1->z*n2->y;
      b = n1->z*n2->x - n1->x*n2->z;
      a = n1->x*n2->y - n1->y*n2->x;

      /* Establecer la línea eqn del borde */
      x0 = y0 = z0 = 0;
      f = g = h = 0;
      switch (borde) {
            case borde01:x0 = z0 = 0.5; g = 1; break;
            case borde02:y0 = z0 = 0.5; f = 1; break;
            case borde03:x0 = -0.5; z0 = 0.5; g = 1; break;
            case borde04:y0 = -0.5; z0 = 0.5; f = 1; break;
            case borde12:x0 = z0 = 0.5; h = 1; break;
            case borde23:x0 = -0.5; y0 = 0.5; h = 1; break;
            case borde34:x0 = y0 = -0.5; h = 1; break;
            case borde41:x0 = 0.5; y0 = -0.5; h = 1; break;
            case borde51:x0 = 0.5; z0 = -0.5; g = 1; break;
```

```
            case borde52:y0 = 0.5; z0 = -0.5; f = 1; break;
            case borde53:x0 = z0 = -0.5; g = 1; break;
            case borde54:x0 = z0 = -0.5; f = 1; break;
      }

      /*regresar la intersección del plano con el borde */
      denom = a*f + b*g + c*h;
      t = -(a*x0 + b*y0 + c*z0)/denom;
      pt->x = x0 + f*t;
      pt->y = y0 + g*t;
      pt->z = z0 + h*t;
}

/*  proyectar(), proyecta el polígono descrito en el arreglo
      cubo[]*/
/* sobre una cara dada del cubo y convertirlo en el espacio UV, */
/* almacenándolo en el arreglo uv[] */

void proyectar(cara)
int cara;
{
      int i;


      switch (cara) {
            case cara0:
                  for(i=0;i<nu_cubo;i++) {
                        uv[i][0] = -cubo[i].x + 0.5;
                        uv[i][1] = cubo[i].y + 0.5;
                  }
                  break;
            case cara1:
                  for(i=0;i<nu_cubo;i++) {
                        uv[i][0] = cubo[i].z + 0.5;
                        uv[i][1] = cubo[i].y + 0.5;
                  }
                  break;

            case cara2:
                  for(i=0;i<nu_cubo;i++) {
                        uv[i][0] = cubo[i].x + 0.5;
                        uv[i][1] = cubo[i].z + 0.5;
                  }
                  break;
            case cara3:
                  for(i=0;i<nu_cubo;i++) {
                        uv[i][0] = -cubo[i].z + 0.5;
                        uv[i][1] = cubo[i].y + 0.5;
                  }
                  break;
            case cara4:
                  for(i=0;i<nu_cubo;i++) {
                        uv[i][0] = cubo[i].x + 0.5;
                        uv[i][1] = -cubo[i].z + 0.5;
                  }
                  break;
            case cara5:
                  for(i=0;i<nu_cubo;i++) {
                        uv[i][0] = cubo[i].x + 0.5;
                        uv[i][1] = cubo[i].y + 0.5;
                  }
                  break;
```

```
        }
    }
```

# 4.9 MAPEO DE LA OPACIDAD

Los efectos de transparencia se pueden implementar dentro del medio ambiente del trazado del rayo, como objetos que pueden ser declarados transparentes y asignar un índice de refraxión.

No obstante, dentro de otros esquemas de representación, el mapeo de opacidad proporciona un método efectivo para crear efectos variables de transparencia. Básicamente, consiste en hacer un mapa de opacidad (o mapa de transparencia, lo que es lo inverso) que contiene valores de control de grados de opacidad dados, para la parte de superficie asociada. Si está sujeto a las mismas funciones de mapeo inverso para determinar el trazo del pixel, y cuando el valor de opacidad es calculado controlará el nivel de opacidad asignado para la superficie en este pixel. Cualquier cosa detrás de la superficie será visible, según el grado de opacidad.

# 4.10 RENDERMAN

Bill Reeves y Pat Hanrahan, como parte del grupo de Lucasfilm, diseñaron un software de interface, para representar la alta complejidad de las imagenes. Este software de interface en su primera versión fue llamado RenderMan.

El RenderMan proporciona una metodología para descripción de la escena donde los objetos, escenas, luces y cámaras pueden ser definidas y manipuladas coherentemente a través de llamadas de procedimientos. También proporciona al usario una herramienta de interface poderosa para construir objetos 3-D, curvas de Bézier y superficies B-spline. Una de las características más importanes de RenderMan es la distinción que hace entre el modelar y la representación de dominios.

Dentro de la técnica de RenderMan, existe lo que se llama árboles de textura que son usados para especificar y manejar las propiedades de sombreado de una superficie. Estos árboles tienen un máximo de dos ramas, la izquierda y la derecha. Los únicos valores que se encuentran en las ramas son valores que definen a los colores. Los nodos y hojas del árbol contienen datos, que son parámetros que son especificados por el usuario, además que se guardan punteros a funciones. El esquema de árboles de textura se puede implementar usando una rutina recursiva llamada eval_árbol(), que evalúa el árbol; esta función utiliza un polígono y un árbol y produce un color. Si la rutina pasa por un nodo, evalúa la función que está definida por el puntero en dicho nodo y luego evalúa las ramas izquierza y derecha en forma recursiva.

En esta función, el puntero *árbol_p sirve para recorrer el árbol. La estructura Arbol está definida como la unión de dos estructuras. Ambas estructuras del nodo y la hoja contienen un entero. A continuación, se muestra la rutina 4.3 llamada eval_árbol().

Eval_árbol()

```
#include "types.h"

void  eval_arbol(pol,árbol,color)
struct polygon *pol
union TREE árbol;
float color[3];
{
      float mask_no;
      floag colorl[3],colorr[3]
```

```
    if (árbol->node.type == NODE) {
          mask_no = (*(árbol->node.func)) (pol,árbol);

          if (mask_no == 0)
                eval_árbol(pol,árbol->node.left,color);
          else if (mask_no >= 1)
                      eval_árbol(pol,árbol->node.right,color);
                else {
                      eval_árbol(pol,árbol->node.left,colorl);
                      eval_árbol(pol,árbol->node.right,colorr);
                      color[0]=colorl[0] + mask_no*(colorr[0] -
                                                    colorl[0]);

                      color[1]=colorl[0] + mask_no*(colorr[1] -
                                                    colorl[1]);
                      color[2]=colorl[0] + mask_no*(colorr[2] -
                                                    colorl[2]);
                }
          }
          else eval_hoja(pol,árbol,color);
}
```

# CAPITULO 5

# HERRAMIENTAS PARA LA ANIMACIÓN POR COMPUTADORA

## INTRODUCCIÓN

En la animación tradicional, cualquier cosa puede ser representada usando un lápiz, papel, aunque algunas veces el trazado de imágenes de película o video proporciona una ayuda para lograr algunos movimientos complejos. En la Animación por Computadora, muchas de las técnicas y experiencias del animador pueden ser medidas de cómo puede controlar los programas para simular el efecto deseado. Por ejemplo, en la animación por celdas, el cambio de los contornos es una simple operación. El primero y último cuadro se dibujan inicialmente en hojas separadas y son intercambiadas por un instante. Si el movimiento entre la primera y la última es lineal, entonces se deben calcular las distancias iguales movidas por cada cuadro y dibujar los cuadros intermedios de acuerdo a dichos cáculos.

En el cálculo de los cuadros intermedios, el animador utiliza un trazo para controlar la velocidad a la cual la animación se moverá; esto es muy importante para crear ciclos de caminado, movimientos de ojos y otros movimientos sutiles. En el presente capítulo, se examinarán algunas herramientas que se encuentran disponibles dentro del ambiente de animación por computadora.

## 5.1 REPRESENTACIÓN DE ATRIBUTOS EN FORMA NUMÉRICA

Un atributo es cualquier característica o cualidad asociada a las cosas que incluye el ambiente de Animación por Computadora. Algunos ejemplos, matiz, saturación, valor, altura, intensidad, anchura, brillo, velocidad angular, coordenadas, transparencia, traslucencia, longitud focal, campo de visión, tensión y niveles de absorción atmosférica. Todos estos pueden ser usados para describir las cámaras, objetos y luces asociadas con una secuencia de animación; todas ellas, sin excepción, están almacenadas numéricamente dentro de la computadora. Algunos de los números pueden ser decimales, como en el caso de las coordenadas $x$, $y$ e $z$, de un vértice, los cuales también pueden ser negativos. También los números que representan niveles de matices, saturación y los valores son siempre positivos con un límite superior de 1, mientras que los niveles de rojo, verde y azul en un cuadro almacenado son representados por enteros positivos, con un rango entre 0 a 255.

El software de animación es capaz de asegurar que estos números retengan su integridad, si son valores erróneos pueden causar problemas en el sistema. Por ejemplo, la distancia entre dos puntos puede ser sólo un número positivo; una distancia negativa no tiene significado físico (aparte de su uso en las coordenadas Cartesianas). Similarmente, las dimensiones como la altura, ancho y profundidad son siempre cantidades positivas, y aunque un programa de modelación pueda aceptar un valor negativo para una profundidad, no existe garantía que resultará un valor válido.

Los valores negativos no deben exceder de sus rangos, ya que resultarían efectos no deseados. Por lo que se necesita que se esté monitoreando contínuamente los cálculos de las intensidades de rojo, verde y azúl para prevenir condiciones de fuera de rango.

El traslado de atributos físicos a números puede parecer una desventaja pero, por el contrario, es una notación fácil de interpretar por la computadora. Para cuando los programas de animación tengan que accesar las descripciones

numéricas, cualquier atributo pueda ser modificada a velocidades tan rápidas que hace que la Animación por Computadora sea posible y útil.

Analizando, por un momento, cualquier pieza dentro del ambiente de la Animación por Computadora. No importa cómo ha sido producida la animación, si fue con un sistema de animación comercial o con un sistema de programas FORTRAN o C++, cada cuadro resulta del estado de diferentes parámetros localizados dentro de la memoria de la computadora. La totalidad de la secuencia de animación fue producida cambiando dichos parámetros en una forma ordenada para lograr algún efecto visual deseado. Los atributos pueden ser cambiados en una gran cantidad de formas, incluyendo el uso de fórmulas, interpolación, datos obtenidos externamente o aleatoriamente; algunos de estos métodos se examinarán a continuación.

# 5.2 NÚMEROS DE LA ANIMACIÓN

La matemática es la forma en que se puede describir algún comportamiento físico. La matemática puede ser considerado como un lenguaje de reglas para manejar números, los cuales son relacionados frecuentemente con sistemas físicos. Por ejemplo, cuando un objeto pesado es suspendido de un resorte, es posible que la tensión del resorte y la masa del objeto permita al sistema oscilar libremente en alguna frecuencia resonante. Las técnicas matemáticas proporcionan un mecanismo para calcular esta frecuencia sin tener que construir el sistema. De hecho, en investigaciones recientes dentro de los sistemas físicos de animación, se utilizan técnicas matemáticas.

La matemática proporciona soluciones para problemas donde dos o más valores numéricos son conocidos, por lo cual es posible determinar algún patrón fundamental. Por ejemplo, se dice tener la posición de tres puntos diferentes sobre un plano que se encuentra sobre un arco circular; una simple fórmula permite calcular el radio y el centro del círculo.

Si, por el momento, no se sabe cómo el animador se relaciona con el sistema de animación, entonces el cambiar los números resulta la clave del éxito de la animación. Si solamente los valores que representan los atributos que describen los objetos, luces y cámaras pueden ser cambiados correctamente, entonces el problema está resuelto.

## 5.2.1 INTERPOLACIÓN LINEAL

En la Animación por Computadora, se encuentra frecuentemente con el problema de conocer que un objeto está localizado en el punto $P_1(x_1,y_1,z_1)$ en el cuadro $F_1$, y en el punto $P_2(x_2,y_2,z_2)$ en el cuadro $F_2$, y el objeto ha sido movido de $P_1$ a $P_2$ como el contador del cuadro procede de $F_1$ a $F_2$. El cálculo de estos valores intermedios es llamado interpolación, y cuando tienen espacios iguales, es usado el término de interpolación lineal.

Si se introduce un nuevo concepto en la forma de parámetro $t$, el cálculo de los valores interpolados es muy fácil. El parámetro $t$ varía entre 0 y 1, y las posiciones intermedias son especificadas por $P(x,y,z)$ de la siguiente forma:

$$x = (1 - t)x_1 + tx_2$$
$$y = (1 - t)y_1 + ty_2$$
$$z = (1 - t)z_1 + tz_2$$

donde $0 <= t <= 1$.

Una revisión rápida de estas funciones confirma que cuando $t$ es igual a cero, $P(x,y,z)$ es igual a $P_1$, y cuando $t$ es igual a 1, el punto $P(x,y,z)$ es colocado en la correspondiente posición lineal interpolada. De cualquier modo, de alguna forma se tiene que controlar el valor de $t$, y en este ejemplo será el número de cuadro $F$ actual, que empieza con un valor de $F_1$ y finaliza con $F_2$. La siguiente regla para el cálculo $t$ asegura que cuando $F$ es igual $F_1$, $t$ es igual cero, y cuando $F$ es igual a $F_2$, $t$ es igual a 1:

$$t = \frac{F - F_1}{F_2 - F_1}$$

La gráfica de esta relación es mostrada en la Figura 5.1.

Los pasos en la secuencia de animación pueden ser resumidos de la siguiente forma:

(1) Calcular t usando $F$, $F_1$ y $F_2$.
(2) Calcular $P(x,y,z)$ usando $t$, $P1$ y $P2$.
(3) Desplegar el objeto en $P(x,y,z)$.

Como el contador de cuadros $F$ aumenta, al igual que $t$, además el objeto se moverá de $P1$ a $P2$, como se deseaba.

Obviamente esta forma de interpolación lineal puede ser aplicada a cualquier par de cantidades, ya sean de color, velocidad, posiciones, ángulos o coordenadas. Otro aspecto de la interpolación es el tipo no lineal que permite alterar el espacio entre los valores interpolados; este tipo de movimientos es importante para la animación.

# 5.2.2 INTERPOLACIÓN NO LINEAL

La primera ley de movimento de Newton establece que 'cada partícula permanece en reposo o se mueve con movimiento uniforme en línea estrecha a menos o hasta que actúe sobre ella una fuerza externa', ya que en la naturaleza, el movimiento uniforme raramente sucede. Una bola que rueda llega al reposo por fuerzas de fricción, y una bala de rifle tiene que caer por la fuerza de gravedad y por el arrastre de la resistencia del viento. Aun y cuando se desea mover un objeto, no cambia rápidamente a la velocidad deseada; su masa resiste la fuerza de propulsión, lo que hace que el objeto sea acelerado a una nueva velocidad.

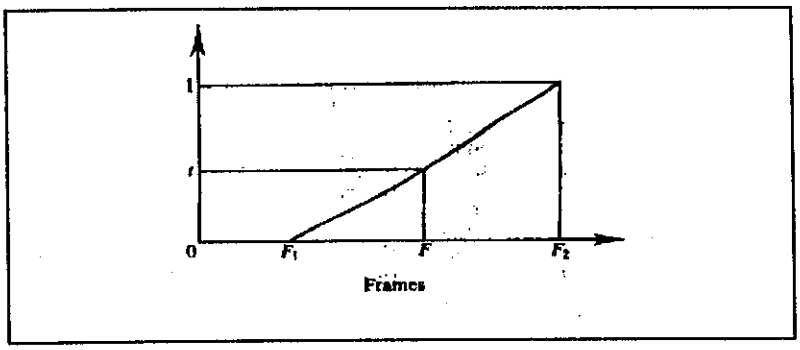



**Figura 5.1**: la línea gruesa representa una relación lineal entre $F$, de la forma que cambia de $F_1$ a $F_2$, y como cambia t de 0 a 1.

Estos movimientos son no lineales y están definidas en la forma de leyes, lo que les permite ser incorporadas dentro de un programa de computación para tener un control exacto de los movimientos de los objetos. Como es imposible simular cualquier cosa usando leyes matemáticas y físicas, se deben usar otras técnicas para aproximarse a su comportamiento.

Un efecto no lineal muy útil es creado por la función seno cuya gráfica está en la forma de ondas. Cuando el parámetro de la función es alterado en formas lineales, crea una secuencia de valores no lineales. Esto puede ser usado para mezclar un número con otro. Por ejemplo, se necesita interpolar entre $N_1$ y $N_2$ tal que la velocidad inicial es rápida, pero disminuye hasta que al final de la secuencia la velocidad es cero. Esto puede ser logrado usando la función seno sobre el período de 0° a 90° y puede ser expresado como:

$$N(t) = (1 - t)N_1 + tN_2,$$

donde $t = sen(\beta)$ y $\beta$ varía de 0° a 90°. La tabla 5.1 muestra los valores interpolados entre $N_1 = 10$ y $N_2 = 20$ sobre 10 cuadros.

La última columna indica el cambio en $N$ y muestra que entre los cuadros 1 y 2 hay un cambio relativo de 1.736, mientras que entre los cuadros 9 y 10, se

ha reducido a 0.152, confirmando el efecto no lineal. Los valores de la Tabla 5.1 se muestran gráficamente en la Figura 5.2.

Si se desea usar otra función que no sea seno, entonces se podría utilizar la relación cuadrática, la cual puede ser implementada declarando a *N* de la siguiente forma:

$$N(t) = N_1 + (N_2 - N_1)t^2$$

Esta función crea los valores de *N* como se muestra en la Tabla 5.2, con *t* que varía de 0 a 1 en pasos iguales.

**Tabla 5.1**

| *Cuadro* | *ß* | *N* | *Cambio* |
|---|---|---|---|
| 1 | 0.0 | 10.000 | - |
| 2 | 10.0 | 11.736 | 1.736 |
| 3 | 20.0 | 13.420 | 1.684 |
| 4 | 30.0 | 15.000 | 1.580 |
| 5 | 40.0 | 16.428 | 1.428 |
| 6 | 50.0 | 17.660 | 1.232 |
| 7 | 60.0 | 18.660 | 1.000 |
| 8 | 70.0 | 19.397 | 0.737 |
| 9 | 80.0 | 19.848 | 0.451 |
| 10 | 90.0 | 20.000 | 0.152 |

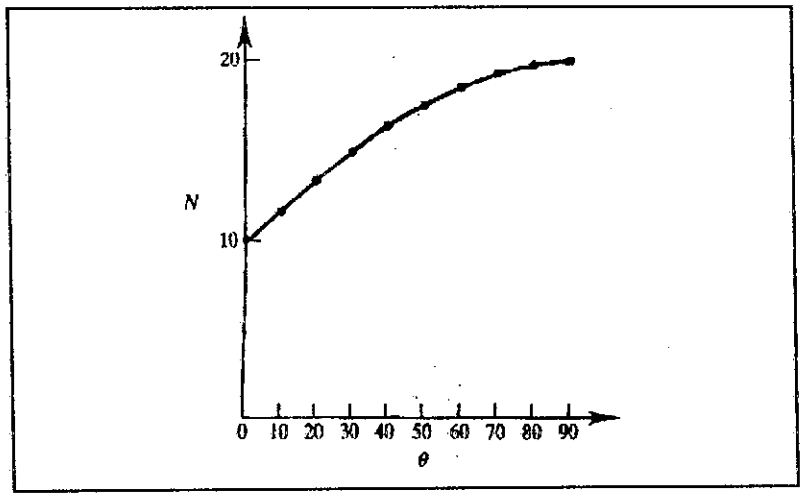



**Figura 5.2**: esta gráfica muestra como una interpolación no lineal se puede lograr entre *N = 10*, y *N = 20*, usando una interpolación senoidal. Los valores se muestran en la Tabla 5.1.

Nótese que la velocidad de cambio en *N* empieza lentamente en 0.1, y termina con un incremento final de 1.9. La gráfica de estos datos se muestra en la Figura 5.3.

# 5.2.3 FÓRMULAS

Las fórmulas matemáticas son importantes para controlar los movimientos que tienen una regularidad precisa o una forma ya definida. Por ejemplo, las manecillas de un reloj rotan a una velocidad angular constante; el brazo de oscilación de un metrónomo tiene un ritmo constante; una bola que rebota traza parábolas que disminuyen conforme la bola pierde su energía a través de la fricción, debido a la resistencia del aire y creando ruido y calor. Este tipo de comportamiento puede ser animado utilizando las ecuaciones usadas en mecánica, física y ciencia.

**Tabla 5.2**

| *Cuadro* | *ß* | *N* | *Cambio* |
|---|---|---|---|
| 1 | 0.0 | 10.0 | - |
| 2 | 0.1 | 10.1 | 0.1 |
| 3 | 0.2 | 10.4 | 0.3 |
| 4 | 0.3 | 10.9 | 0.5 |
| 5 | 0.4 | 11.6 | 0.7 |
| 6 | 0.5 | 12.5 | 0.9 |
| 7 | 0.6 | 13.6 | 1.1 |
| 8 | 0.7 | 14.9 | 1.3 |
| 9 | 0.8 | 16.4 | 1.5 |
| 10 | 0.9 | 18.1 | 1.7 |
| 11 | 1.0 | 20.0 | 1.9 |

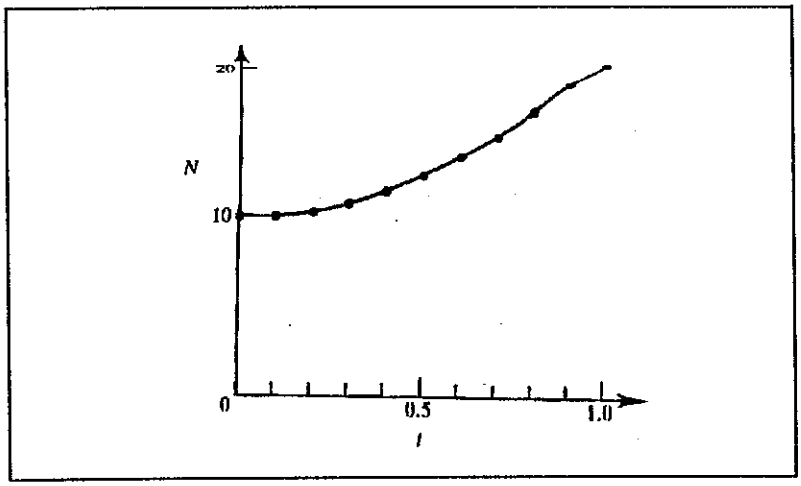



**Figura 5.3**: esta gráfica muestra como una interpolación no lineal se puede lograr entre *N = 10*, y *N = 20*, usando una interpolación cuadrática. Los valores se muestran en la Tabla 5.2.

Como un simple ejemplo, se considera el problema de animar un objeto que rota alrededor del origen para un radio fijo, y al mismo tiempo ondula hacia arriba y abajo. En vez de asociar valores absolutos con el radio, velocidad y altura de oscilación, se declaran parámetros que son ajustados para proporcionar un rango de animaciones. Algunos parámetros útiles podrían ser:

*F* es el número de cuadro actual.
*R* es el radio de rotación.
ß es el ángulo rotado por el objeto por cuadro de animación.
*H* es la amplitud de las ondulaciones senoidales.
*N* es el número de ondulaciones por rotación.

El movimiento del objeto es ilustrado en la Figura 5.4, y puede ser descrita definiendo las tres coordenadas, así:

x = Rsen(Fß)
z = Rcos(Fß)
y = Hsen(NFß)

Las coordenadas *x* e *z* controlan la rotación alrededor del origen sobre el plano, mientras que la coordenada *y* asegura que el objeto ondula hacia arriba y abajo con un movimiento ondular de amplitud *H*. El término *Fß* asegura que como el contador de cuadros *F* aumenta, el ángulo rotado por el objeto es generado automáticamente. El término *NFß* para la coordenada *y* puede ser tomada de la siguiente forma: si *N = 1*, entonces durante una rotación el objeto se levantará y caerá exactamenta una vez, y si *N = 2*, entonces se levantará y caerá dos veces durante una revolución. La rotación será en el sentido contrario de las agujas

del reloj alrededor del eje-y, pero puede ser al revés haciendo a $\beta$ negativo.

El objeto es animado calculando $x$, $y$ e $z$ por cada cuadro. A la animación se le pueden agregar detalles permitiendo al objeto rodar alrededor de su propio centro mientras realiza una rotación. La acción de rodar se realizará antes de que el objeto sea trasladado al punto $(x,y,z)$

El ejemplo anterior muestra una técnica importante, la cual es de dividir el problema en varios componentes y resolverlos individualmente. La geometría de coordenadas es una herramienta importante para determinar estos componentes, y en donde se tratará de analizar cualquier problema geométrico pensando en términos de expresiones paramétricas que generan las coordenadas separadas de $x$, $y$ e $z$.

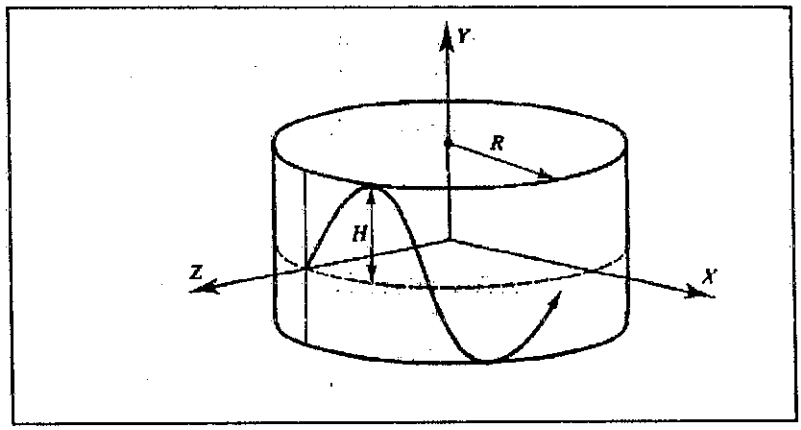



**Figura 5.4**: la trayectoria de este objeto es tal que, mientras rota alrededor del eje-y, también está sujeto a un desplazamiento vertical senoidal.

## 5.2.4 DATOS EXTERNOS

Existirán ocasiones cuando los datos externos tienen que ser ingresados para controlar una secuencia de animación; esto puede ser en forma de una curva especial que la cámara debe seguir, o la variación en tamaño asociada con un objeto. Un buen ejemplo, es la simulación del latido del corazón humano. Después de haber modelado el corazón con base en secciones transversales de datos geométricos, algunos datos pueden describir como ciertas partes del corazón se mueven en el espacio. Estos datos pueden ser ingresados en la computadora y usados para manipular en modelo en 3-D. Básicamente, no existen límites para el tipo de datos que pueden ser ingresados a un programa de animación, siempre y cuando exista una trayectoria de entrada en el sistema de animación.

## 5.2.5 NÚMEROS ALEATORIOS

La generación de números aleatorios proporciona una fuente útil de datos numéricos: cuando se instruye, los programas proporcionan un número aletario dentro de un rango especificado, teniendo una distribución de probabilidad, por ejemplo: uniforme. De hecho, estos programas son llamados generadores de números pseudo-aleatorios, ya que generalmente tienen un ciclo que se repite una y otra vez.

Una aplicación de los números aleatorios es el de asignar rotaciones aleatorias que podrían ser asociadas con un gran número de objetos. Se tiene la siguiente situación, donde 1000 cubos tienen que ser mostrados rotando en el espacio alrededor de sus propios centros, y cada cubo debe tener un movimiento aleatorio dentro de un rango dado. Una solución simple podría ser almacenar para cada cubo tres números aleatorios que representan las rotaciones angulares alrededor de los ejes $x$, $y$ e $z$ en un arreglo -en total 3000 números. La escena

final es creada tomando cada cubo a la vez, y el cubo está sujeto a tres rotaciones individuales acumuladas. Aun las posiciones de los cubos podrían ser determinadas aleatoriamente, pero existe la posibilidad de que dos o más cubos se crucen uno con otro.

## 5.3 TÉCNICAS PARAMÉTRICAS DE ENTREMEZCLADO

Una facilidad importante en cualquier sistema de animación es la habilidad de hacer ajustes a un parámetro sin tener que introducir alguna discontinuidad, o perturbar otras partes de la secuencia que si están bien. Ahora se verá cómo las técnicas paramétricas de entremezclado pueden ser usadas para proporcionar técnicas de inteporlación alternativas.

### 5.3.1 MEZCLA CUADRÁTICA Y CÚBICA

Aquí se examinarán las mismas técnicas paramétricas, pero para el uso de mezclar cualquier cantidad numérica, ya sea que representen coordenadas cartesianas o una intensidad de luz.

Si se necesita una mezcla cuadrática (ley del cuadrado), esto se puede lograr usando la siguiente expresión para realizar la mezcla:

$$N(t) = (1 - t)^2N_1 + t^2N_2$$

No obstante, cuando los términos en $t$ son expandidos, se encuentra que no son más que la unidad; un término extra $2t(1 - t)$ es incluido en la expresión anterior de la siguiente forma:

$$N(t) = (1 - t)^2N_1 + t^2N_2 + 2t(1 - t)$$

Ahora la parte interesante respecto al término extra es que cuando $t$ es igual a cero o uno, los valores de los términos también son cero, además no tiene influencia en los extremos del proceso de mezclado. Obviamente debe tener algún impacto sobre la forma en que se lleva a cabo la mezcla de $N_1$ a $N_2$, el impacto podría ser controlado multiplicando la expresión por un valor extra $N_c$, de la siguiente forma:

$$N(t) = (1 - t)^2N_1 + t^2N_2 + 2t(1 - t)N_c$$

El efecto de $N_c$ se puede ver en la Figura 5.5, la cual muestra las mezclas entre dos números para diferentes valores de $N_c$. Nótese que ciertos valores de $N_c$ permiten al valor mezclado caer hasta $N_1$, y aún pasarse de $N_2$, lo que hace, o no, una característica útil. Existen límites para el tipo de mezcla creada para diferentes valores de $N_c$, pero para mayor flexibilidad puede ser agregado utilizando una función cúbica, de la siguiente forma:

$$N(t) = (1 - t)^3N_1 + t^3N_2$$

e igual a la forma cuadrática, el término $t$ debe sumar la unidad, lo cual pasará si se incluyen dos términos más, de la siguiente forma:

$$N(t) = (1 - t)^3N_1 + t^3N_2 + 3t(1 - t)^2 + 3t^2(1 - t)$$

De igual forma, los términos extra no tienen influencia en los límites extremos del proceso de mezcla, y también pueden ser multiplicados por dos valores extra para controlar su influencia, de la siguiente forma:

$$N(t) = (1 - t)^3N_1 + t^3N_2 + 3t(1 - t)^2N_c + 3t^2(1 - t)N_d$$

Está demás decir, que existen ahora muchas combinaciones para valores de $N_c$ y $N_d$ que tienen un efecto sobre las curvas mezcladas entre $N_1$ y $N_2$.

## 5.3.2 MEZCLA DE HERMITE

En la mezcla de Hermite, las funciones cúbicas pueden ser usadas para mezclar dos valores $N_1$ y $N_2$ junto con otro par $S_1$ y $S_2$ que controla la velocidad a la cual empieza y termina la mezcla respectivamente. El proceso puede ser representado en una forma de matriz, de la siguiente forma:

$$N(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ S_1 \\ S_2 \end{bmatrix}$$

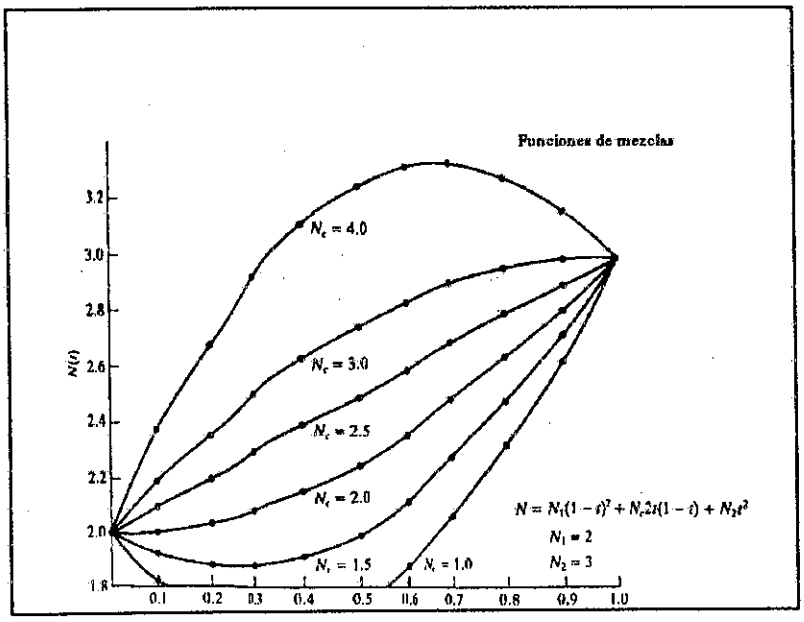



**Figura 5.5:** esta familia de curvas se levantan alterando el valor de $N_c$ en la ecuación paramétrica: $N = N_1(1 - t)^2 + N_c 2t(1 - t) + N_2 t^2$, donde $N_1 = 2$, $N_2 = 3$ y $0<=t<=1$.

La mejor forma de visualizar lo que está sucediendo es construir una gráfica en donde el parámetro $t$ sea representado por un eje horizontal, y la escala numérica para el eje vertical. Si ambos ejes tienen la misma escala, entonces $S_1$ y $S_2$ serán la inclinación de la función de mezcla en $t = 0$ y $t = 1$, respectivamente.

Esto funciona interpolando $N_1 = 1$ y $N_2 = 2$ con diferentes combinaciones de $S_1$ y $S_2$. En la Tabla 5.3, $N$ ha sido calculado con diferentes valores de $S_1$, $S_2$ y $t$. Estas mezclas se muestran en la Figura 5.6. La gráfica muestra que cuando las inclinaciones $S_1$ y $S_2$ son iguales a 1, ocurre una interpolación lineal. Cuando ambas inclinaciones son cero, la función de mezcla se mueve hacia el valor inicial, interpolando valores intermedios, y al mismo tiempo anticipando de que $N_2$ tendrá la misma velocidad con la cual se queda $N_1$.

La mezcla de Hermite proporciona una técnica útil para un ambiente de Animación por Computadora, que permite simular movimientos lentos que son importantes en la animación. No obstante, la animación no siempre concierne a la interpolación entre un par de valores. Un animador necesitará mezclar entre una serie de valores asociados con algún cuadro principal, y es importante que la continuidad se mantenga sobre la secuencia.

**Tabla 5.3**

| | Valores de N | | |
|---|---|---|---|
| $t$ | $S_1=1, S_2=1$ | $S_1=0, S_2=0$ | $S_1=2, S_2=0$ |
| 0.0 | 1.0 | 1.00 | 1.00 |
| 0.1 | 1.1 | 1.03 | 1.19 |
| 0.2 | 1.2 | 1.10 | 1.36 |
| 0.3 | 1.3 | 1.22 | 1.51 |
| 0.4 | 1.4 | 1.35 | 1.64 |
| 0.5 | 1.5 | 1.50 | 1.75 |
| 0.6 | 1.6 | 1.65 | 1.84 |
| 0.7 | 1.7 | 1.78 | 1.91 |
| 0.8 | 1.8 | 1.90 | 1.96 |
| 0.9 | 1.9 | 1.97 | 1.99 |
| 1.0 | 2.0 | 2.00 | 2.00 |

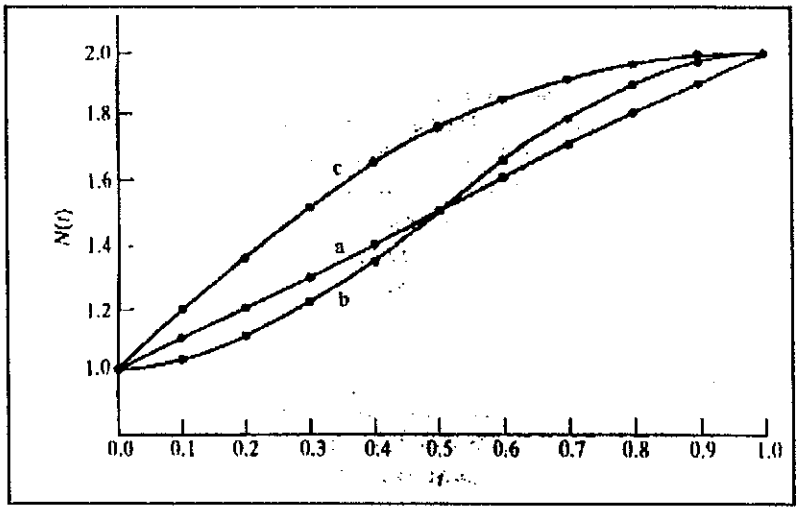



**Figura 5.6:** estas tres curvas muestran la función de mezcla de Hermite aplicada entre los valores de 1 y 2 con diferentes condiciones de inclinación en los dos extremos. La curva (a) tiene inclinaciones en sus extremos de 1 y 1; la curva (b) tiene inclinaciones en sus extremos de 0 y 0; la curva (c) tiene inclinacinoes en sus extremos de de 2 y 0. Los valores se muestran en la Tabla 5.3.

# 5.3.3 MEZCLA PARABÓLICA

La mezcla parabólica es una técnica que permite un gran número de valores para ser interpolados continuamente, el control de los valores está incluída en la secuencia interpolada. Por ejemplo, se necesita interpolar entre una secuencia de valores $N_1$, $N_2$, $N_3$, $N_4$, $N_5$, etc., usando una función cúbica. El valor interpolado $N(t)$ está dado por:

$$N(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1.0 & -2.5 & 2.0 & -0.5 \\ -0.5 & 0.0 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix}$$

donde $0 <= t <= 1$.

El rango de $N(t)$ solamente estará entre $N_2$ y $N_3$ y para obtener la siguiente porción de la mezcla, el proceso se debe repetir para $N_2$, $N_3$, $N_4$ y $N_5$, con $t$ variando entre 0 y 1. Una consecuencia de esta técnica es que los valores entre $N_1$ y $N_2$ no son generados.

Por ejemplo, se desea interpolar los valores 1, 3, 6, 2, 1. Primero, los valores se extenderán para incluir el primero y el último valor dos veces, entonces se tendrá 1, 1, 3, 6, 2, 1, 1. Aplicando la operación de matriz anterior para los primeros cuatro dígitos, resulta:

$$N(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1.0 & -2.5 & 2.0 & -0.5 \\ -0.5 & 0.0 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 3 \\ 6 \end{bmatrix}$$

Esta proporciona los valores de la Tabla 5.4 y la Figura 5.7 como $t$ varía de 0 a 1.

**Tabla 5.4**

| $t$ | $N(t)$ |
|---|---|
| 0.0 | 1.000 |
| 0.1 | 1.115 |
| 0.2 | 1.256 |
| 0.3 | 1.421 |
| 0.4 | 1.608 |
| 0.5 | 1.813 |
| 0.6 | 2.032 |
| 0.7 | 2.264 |
| 0.8 | 2.504 |
| 0.9 | 2.751 |
| 1.0 | 3.000 |

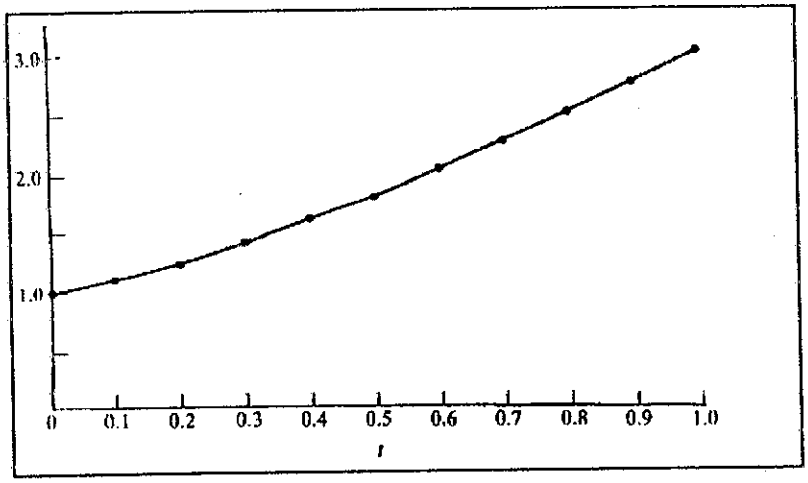



**Figura 5.7**: esta gráfica muestra el efecto de una función de mezcla parabólica que fue creada de los datos de la Tabla 5.4.

La siguiente secuencia de mezcla puede ser calculada reemplazando los valores 1, 1, 3, 6 por 1, 3, 6, 2, y para la mezcla restante 3, 6, 2, 1 y 6, 2, 1, 1. Estos han sido calculados y tabulados en la Tabla 5.5. La secuencia de mezcla puede ser vista en la segunda columna, seguida por la tercera, etc. Estos datos están representados gráficamente por la curva de la Figura 5.8, en la cual se trazan los valores contra un parámetro lineal horizontal. La duplicación del primer y último valor fue arbitrario, pero se pueden agregar otros valores que proporcionarán un mecanismo para controlar la inclinación de los valores inteporlados al inicio y al final de la secuencia.

**Tabla 5.5**

| $t$ | Secuencias de $N_1$, $N_2$, $N_3$, $N_4$ 1,1,3,6 | 1,3,6,2 | 3,6,2,1 | 6,2,1,1 |
|---|---|---|---|---|
| 0.0 | 1.000 | 3.000 | 6.000 | 2.000 |
| 0.1 | 1.115 | 3.291 | 5.870 | 1.774 |
| 0.2 | 1.256 | 3.648 | 5.600 | 1.592 |
| 0.3 | 1.421 | 4.047 | 5.220 | 1.448 |
| 0.4 | 1.608 | 4.464 | 4.760 | 1.336 |
| 0.5 | 1.813 | 4.875 | 4.250 | 1.250 |
| 0.6 | 2.032 | 5.256 | 3.720 | 1.184 |
| 0.7 | 2.264 | 5.583 | 3.200 | 1.132 |
| 0.8 | 2.504 | 5.832 | 2.720 | 1.088 |
| 0.9 | 2.751 | 5.979 | 2.310 | 1.046 |
| 1.0 | 3.000 | 6.000 | 2.000 | 1.000 |

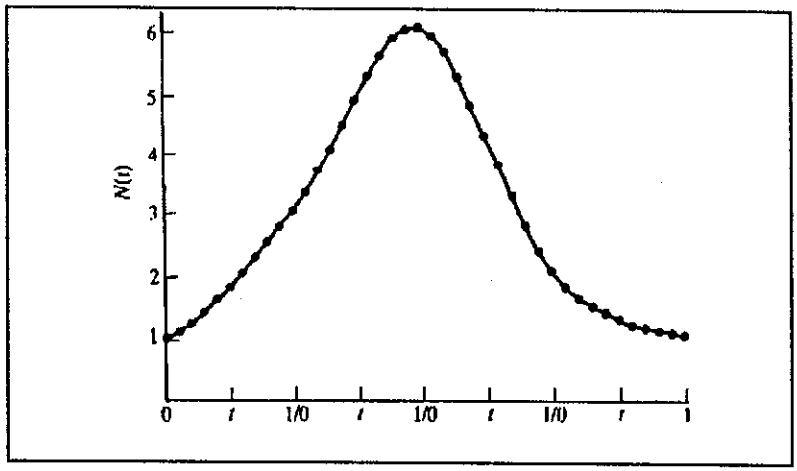



**Figura 5.8**: esta mezcla parabólica puede ser usada sobre un rango de números, y en este diagrama se puede ver como la técnica interpola la secuencia 1, 3, 6, 2 y 1.

Una extensión de esta técnica permite el concepto de 'tensión'; éste controla la tangente dela curva cuando pasa a través de los valores de control. Esto es incorporado introduciendo un parámetro de tensión $T$ en la matriz, de la siguiente forma:

$$\begin{bmatrix} -T & 2-T & T-2 & T \\ 2T & T-3 & 3-2T & -T \\ -T & 0 & T & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

La matriz original es equivalente para un valor de $T$ igual a 0.5, y cuando $T$ es igual a 1, se crea la siguiente matriz:

$$\begin{bmatrix} -1 & 1 & -1 & 1 \\ 2 & -2 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

A continuación, se muestra el Algoritmo 5.1 que es utilizado para describir un esquema de interpolación para un vector no periódico.

Algoritmo 5.1: tridiag_circular()

```
tridiag_circular(a,b,c,d,x,n)
float a[],b[],b[],d[],x[];
int n;
{
      int i,j;
      float beta,*gamma,*mu,*delta;
      float bn,cn,dn,M[3][3],D[3][3],X[3];

      gamma = (float *)malloc(n*sizeof(float));
      delta = (float *)malloc(n*sizeof(float));
      mu = (float *)malloc(n*sizeof(float));

      /* eliminación hacia adelante */
      gamma[0] = c[0]/b[0];
      mu[0] = a[0]/b[0];
      delta[0] = d[0]/b[0];
      for (j=1; j<n-2; j++) {
            beta = b[j] - a[j]*gamma[j-1];
            gamma [j] = c[j]/beta;
            mu[j] = -a[j]*mu[j-1]/beta;
            delta[j] = (d[j] - a[j]*delta[j-1])/beta;
      }

      /*eliminación sucesiva del valor más a la izquierda de la fila */
      /*inferior de la matriz */
      cn=c[n-1];
      bn=b[n-1];
      dn=d[n-1];
      for (j=1; j<n-2; j++) {
            bn = bn - mu[j-1]*cn;
            dn = dn - delta[j-1]*cn;
            cn = -gamma[j-1]*cn;
      }

      /* resolución del sistema simultáneo de tres ecuaciones */
      M[0][0] = 1; M[0][1] = gamma[n-3]; M[0][2] = mu[n-3];
      M[1][0] = a[n-2]; M[1][1] = b[n-2]; M[1][2]= c[n-2];
      M[2][0] = cn; M[2][1] = a[n-1]; M[2][2] = bn;
      D[0] = delta[n-3]; D[1] = d[n-2]; D[2] = dn;
      Resolver(M,D,X);

      /* substitución hacia atrás */
      for (j=0; j<3; j++) x[n-3+i] = X[j];
      for (j=n-4; j>=0; j--) x[j] = delta[j] - gamma[j]*x[j+1] -
                                    mu[j]*x[n-1];

      free(gamma);
      free(delta);
      free(mu);
}

Resolver(M,D,X)
float M[3][3],X[3][3],D[3][3];
{
      float m[2][2],d[2],piv;
      piv = -M[1][0]/M[0][0];
      m[0][0] = M[1][1] + piv*M[0][1];
      m[0][1] = M[1][2] + piv*M[0][2];
      d[0] = D[1] + piv*D[0];
      piv = -M[2][0]/M[0][0];
```

```
        m[1][0] = M[2][1] + piv*M[0][1];
        m[1][1] = M[2][2] + piv*M[0][2];
        d[1] = D[2] + piv*D[0];
        piv = -m[1][0]/m[0][0];
        X[2] = (d[1] + piv*d[0])/(m[1][1] + piv*m[0][1]);
        X[1] = (d[0] + -m[0][1]*X[2])/(m[0][0];
        X[0] = (D[0] + M[0][2]*X[2] - M[0][1]*X[1])/M[0][0]);
}
```

# 5.4 CURVAS DE ESPACIO

Como la animación se relaciona con la posición espacio de los objetos, luces y cámaras, es importante que existan técnicas para efectuar y controlar las maniobras espaciales. Dichas técnicas proporcionan diferentes métodos para definir curvas de espacio 3-D, que pueden ser usadas como trayectorias de vuelo para objetos, luces y cámaras.

Las curvas tridimensionales tienen una forma circular, elíptica, parabólica o cúbica, pueden ser representadas utilizando formulas matemáticas, pero las curvas libres de forma describen alguna trayectoria arbitraria en el espacio, requiere de un estudio diferente.

## 5.4.1 CURVAS DE BEZIER

Para empezar, se usará una curva de Bézier cuadrática en 2-D, cuya forma está especificada en términos de dos expresiones paramétricas que describen las coordenadas $x$ e $y$ de cualquier punto sobre la curva. Si los dos puntos finales de la curva son $P_1$ y $P_2$, con coordenadas $(x_1,y_1)$ y $(x_2,y_2)$ respectivamente, entonces cualquier punto $P(t)$, donde $t$ es un parámetro entre los límites 0 y 1, tiene coordenadas $(x(t),y(t))$ definida de la siguiente forma:

$$x(t) = (1 - t)^2x_1 + t^2x_2 + 2t(1 - t)x_c$$
$$y(t) = (1 - t)^2y_1 + t_2y_2 + 2t(1 - t)y_c$$

donde $(x_c,y_c)$ son las coordenadas de un punto de control el que afecta la forma de la curva entre $P_1$ y $P_2$. La Figura 5.9 muestra la figura de una curva de Bézier cuadrática con dos puntos de control diferentes. Los términos en el parámetro $t$ fueron derivados de una evaluación algebráica de expresiones teniendo la siguiente forma:

$$(a + b)^2 = a^2 + b^2 + 2ab$$

y si $a = (1 - t)$, y $b = t$, se obtiene las expresiones anteriores para $x(t)$ y $y(t)$.

Una expresión natural es introducir una tercera expresión para definir el valor de $z(t)$, la cual crea una curva de espacio Bézier cuadrática en 3-D. El papel del punto de control tiene ahora una interpretación espacial, en que actúa como un punto en el cual la curva es atraída.

La acción de un simple punto de control limita el rango de curvas que pueden ser creadas, no obstante, usando funciones de mezcla cúbicas, dos puntos de control resultan disponibles, y da una flexibilidad extra. Tomando un ejemplo en 3-D, se consideran dos puntos $P_1$ y $P_2$ con coordenadas $(x_1,y_1,z_1)$ y $(x_2,y_2,z_2)$ respectivamente, con puntos de control $(x_c,y_c,z_c)$ y $(x_d,y_d,z_d)$. Entonces, cualquier punto $P(t)$ sobre la curva tendrá coordenadas $(x(t),y(t),z(t))$ definidas por las expresiones:

$$x(t) = (1 - t)^3x_1 + t^3x_2 + 3t(1 - t)^2x_c + 3t^2(1 - t)x_d$$
$$y(t) = (1 - t)^3y_1 + t_3y_2 + 3t(1 - t)^2y_c + 3t^2(1 - t)y_d$$
$$z(t) = (1 - t)^3z_1 + t_3z_2 + 3t(1 - t)^2z_c + 3t^2(1 - t)y_d$$

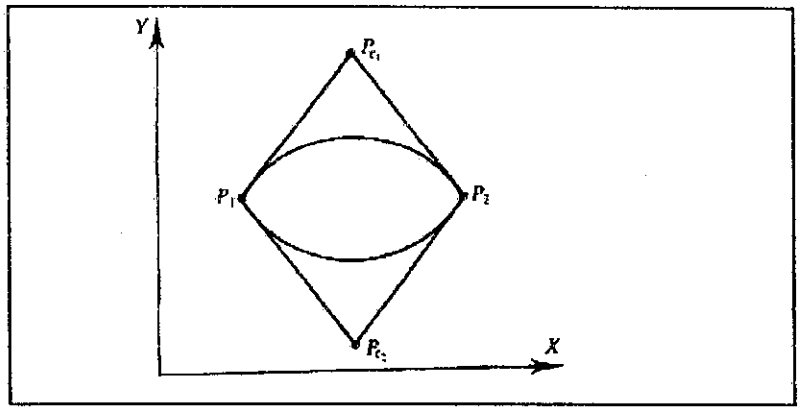



**Figura 5.9**: para construir una curva de Bézier cuadrática entre dos puntos $P_1$ y $P_2$, el punto de control central $P_c$ está localizado en una posición adecuada para obtener la curva deseada. En este diagrama, las posiciones de $P_{c1}$ y $P_{c2}$ muestran su influencia para determinar la forma de la curva.

Los términos en $t$ fueron derivados de una evaluación algebráica de la siguiente expresión:

$$(a + b)^3 = a^3 + b^3 + 3a^2b + 3ab^2$$

donde $a = (1 - t)$ y $b = t$.

Una vez más la acción de los puntos de control es atraer la curva sobre su recorrido entre los puntos límite, con el punto extra proporcionando la habilidad para doblar la curva y crear un amplio rango de trayectorias.

Existe un número de características interesantes de esta curva, la primera de las cuales se refiere a la dirección de la curva en los puntos límite. Cuando la curva primero deja el punto incial $P_1$, que sirve de encabezado para el primer punto de control, pero es inmediatamente desviado de esta meta por la influencia del segundo punto de control y $P_2$. Sin embargo, la inclinación inicial de la curva es igual a la inclinación de la línea que conecta $P_1$ con el primer punto de control. Similarmente, como la curva llega a $P_2$, resulta coincidente con la línea que conecta a $P_2$ con el punto de control adyacente. Es capaz de controlar la inclinación inicial y final que permite cualquier número de dichos segmentos de curva para ser unidos, donde la inclinación final de un segmento iguala la inclinación inicial de un segmento contiguo. Así, las curvas de espacio Bézier pueden ser creadas usando la técnica anterior.

Una segunda característica relacionada con la naturaleza del punto de control que tiene sobre la curva, en que si un punto de control es movido a otra posición, entonces toda la figura es modificada.

Una tercera observación es que la curva no pasa a través de sus puntos de control, además podría si están alineados tal que los puntos estén sobre una línea que conecte $P_1$ y $P_2$. Cuarto, el polígono convexo creado por los puntos de control, siempre estará contenido en la curva de Bézier.

El Algoritmo 5.2 produce una subdivisión de Bézier y divide el corte en las direcciones u y v respectivamente.

Algoritmo 5.2: Dividir()

```
void dividir_u(P,Q,R)
float P[4][4][3],
      Q[4][4][3],
      R[4][4][3];
{
      int i,iu;

      for (iu=0; i<4; iu++) {
            for (i=0; i<3; i++) {
                  Q[0][iu][i] = P[0][iu][i];
                  Q[1][iu][i] = (P[0][iu][i]+P[1][iu][i])/2;
                  Q[2][iu][i] = Q[1][iu][i]/2+
                                (P[1][iu][i]+P[2][iu][i])/4;
```

```
                R[3][iu][i] = P[3][iu][i];
                R[2][iu][i] = (P[2][iu][i]+P[3][iu][i])/2;
                R[1][iu][i] = R[2][iu][i]/2+
                              (P[1][iu][i]+P[2][iu][i])/4;
                Q[3][iu][i] = (Q[2][iu][i]+R[1][iu][i]/2;
                R[0][iu][i] = Q[3][iu][i];
            }
        }
}

void dividir_v(P,Q,R)
float P[4][4][3],
      Q[4][4][3],
      R[4][4][3];
{
      int i,iv;

      for (iv=0; i<4; iv++) {
          for (i=0; i<3; i++) {
                Q[iv][0][i] = P[iv][0][i];
                Q[iv][1][i] = (P[iv][0][i]+P[iv][1][i])/2;
                Q[iv][2][i] = Q[iv][1][i]/2+
                              (P[iv][1][i]+P[iv][2][i])/4;
                R[iv][3][i] = P[iv][3][i];
                R[iv][2][i] = (P[iv][2][i]+P[iv][3][i])/2;
                R[iv][1][i] = R[iv][2][i]/2+
                              (P[iv][1][i]+P[iv][2][i])/4;
                Q[iv][3][i] = (Q[iv][2][i]+R[iv][1][i]/2;
                R[iv][0][i] = Q[iv][3][i];
          }
      }
}
```

## 5.4.2 CURVAS B-SPLINE

Las curvas B-Spline son una generalización de las curvas Bézier que permiten que una curva sea construída a partir de cualquier número de puntos de control tal que; cuando éstos son ajustados, solamente una arcada de la curva es influenciada. El algoritmo B-Spline requiere una lista de parámetros que trazan la curva.

Para interpolar linealmente un par de valores $N_1$ y $N_2$, se usa la siguiente expresión:

$$N(t) = (1 - t)N_1 + tN_2$$

donde t está restringido entre 0 y 1, pero se necesita interpolar linealmente entre los valores $N_0$, $N_1$, $N_2$, $N_3$,..., $N_n$. Un método para lograrlo es crear una secuencia de parámetros que puedan ser asociados con los números mostrados en la Tabla 5.6.
donde $t_0<=t_1<=t_2<=t_3<=...t_n$.

**Tabla 5.6**

| | | | | | |
|---|---|---|---|---|---|
| *valores* | $N_0$ | $N_1$ | $N_2$ | $N_3$ | - |
| *parámetros* | $t_0$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ |

Ahora se necesita un mecanismo para calcular cualquier valor $N(t)$ utilizando funciones de mezcla $B_0(t)$, $B_1(t)$, $B_2(t)$, $B_3(t)$, etc., tal que:

$$N(t) = B_0(t)N_0 + B_1(t)N_1 + B_2(t)N_2 + B_3(t)N_3 + \ldots B_n(t)N_n$$

No obstante, no se desea una gran cantidad de funciones separadas, sino una sola función que se ajuste según el valor de su parámetro $t$.

Se sabe que en la interpolación lineal el valor interpolado es obtenido de dos partes: uno cuyo valor es reducido, y otro cuyo valor es aumentado; ésta es la acción de los términos $(1 - t)$ y $t$. En la situación anterior durante la arcada de los parámetros $t_1$ a $t_2$, el efecto de $N_0$ se debe reducir, mientras que $N_1$ debe aumentar.

La definición final de una función de mezcla está dada por la siguiente prueba:

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t),$$

donde $B(t)$ es una función recursiva, con parámetros $i$, $k$ y $t$, donde:

- $i$ referencia los valores de control y valores de parámetros.
- $k$ es el orden de la función mezcla,
  - = 2 para una mezcla lineal,
  - = 3 para una mezcla cuadrática, y
  - = 4 para una mezcla cúbica.
- $t$ es el parámetro de mezcla.

La acción recursiva de $B_{i,k}(t)$ es terminada por la siguiente condición:

$$B_{i,1}(t) \begin{cases} = 1, & \text{si } t_i <= t <= t_{i+1} \\ = 0, & \text{de otra forma} \end{cases}$$

La lista de parámetros es llamado el knot vector, que tendrá un número suficiente para igualar los valores de control y permitir la acción de una mezcla lineal cuadrática o cúbica. De hecho, si existieran $m + 1$ valores de control, entonces el knot vector debe contener $m + k + 1$ knots. Por ejemplo, si 5(m + 1) valores están en la mezcla cúbica $(k = 4)$, entonces el knot vector tendrá nueve entradas.

En realidad, los valores de control son coordenadas en dos o tres dimensiones, los cuales se pueden mezclar para crear la curva B-Spline. Como se ha visto, cuando es movido un punto de control, solamente dicha parte es afectada. Además, los valores del knot vector permiten al usuario influenciar en la forma de la curva.

Aquí se muestra el método de la matriz:

$$P(t) = 1/6[t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \end{bmatrix}$$

donde para un valor de $t$ dado, restringido entre 0 y 1, y por cuatro valores $N_0$, $N_1$, $N_2$ y $N_3$ que están sobre la curva cúbica.

Cuando los valores $N_0$, $N_1$, $N_2$ y $N_3$ representan las coordenadas $x$, $y$ o $z$ de los puntos de control en el espacio, los valores calculados por $P(t)$ forman un segmento de la curva cúbica B-Spline. Gráficamente, los valores de control resultan ser los puntos de control, y si están conectados para crear el polígono convexo, entonces la curva B-Spline existirá dentro de sus límites.

Para ilustrar la acción de la matriz anterior, se considera la siguiente secuencia de puntos de control en 2-D (1,1), (1,2), (2,2), (2,1) y (1,0), como

se muestra en la Figura 5.10. Se despliegan dos segmentos del B-Spline creados usando los puntos de control (1,1), (1,2), (2,2) y (2,1), y luengo usando (1,2), (2,2), (2,1) y (1,0). Los segmentos inicial y final no pueden ser calculados por lo insuficiente de su geometría, pero si la curva pasa a través del primero y último punto de control, entonces esto puede ser repetido tres veces para crear la secuencia (1,1), (1,1), (1,1), (1,2), (2,2), (2,1), (1,0), (1,0) y (1,0). Si la matriz es aplicada a esta secuencia seis veces, tomando en cuentra los cuatro puntos de control contiguos, la curva completa es mostrada en la Figura 5.10.

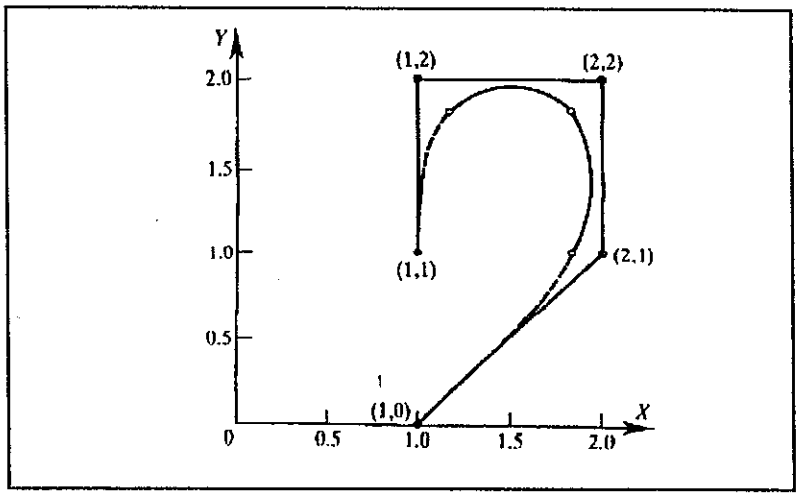



**Figura 5.10**: estos cinco puntos de control son necesarios para construir las dos porciones centrales de la curva cúbica B-Spline; pero si el primer y último punto se repiten tres veces en la matriz de evaluación, entonces los segmentos extremos se pueden desarrollar y resultar parte de la curva.

Las coordenadas inicial y final entre estos segmentos están tabulados en la Tabla 5.7; como se puede ver, los segmentos de curva inicial y final son muy cortos, por lo que se requiere que el parámetro *t* varíe entre 0 y 1. De hecho, para cada segmento, como *t* aumenta en pasos lineales, la longitud de cuerda de la curva varía en una forma no-lineal, lo que puede o no ser lo deseado.

El Algoritmo 5.3 obtiene el punto de control dado un conjunto de puntos y dado un punto del llamado vector knot. Esta función llama a otra función que resuelve los puntos X[ ][3] dado un conjunto lineal en forma diagonal de una matriz; una matriz tridiagonal es aquella en la que los elementos de su diagonal y los dos números más próximos no son cero.

La parte básica de este algoritmo es el siguiente, ya que cada fila de la matriz representa una ecuación lineal del sistema, se podrá alterar una fila de la matriz con sólo agregar múltiplos de otra fila.

Algoritmo 5.3: Interpolación_Bspline()

```
Interpolación_bspline(dato,control,knots,nodato)
float dato[][3]],
      control[][3],
      knots[];
int nodato;
{
      int i;
      float *alfa,*beta,*gamma;
      float t1,t2,t3,t4,t5;

      alfa = (float *)malloc(nodato*size(float));
      beta = (float *)malloc(nodato*size(float));
      gamma = (float *)malloc(nodato*size(float));

      alfa[0] = 0;
      beta[0] = 1;
      gamma[0] = 0;
```

```
        alfa[nodato-1] = 0;
        beta[nodato-1] = 1;
        gamma[nodato-1] = 0;

        for (i=1; i<nodato; i++) {
              t1 = knots[i+1];
              t2 = knots[i+2];
              t3 = knots[i+3];
              t4 = knots[i+4];
              t5 = knots[i+5];

              alfa[i] = (t4 - t3)*(t4 - t3)/(t4 - t1);
              beta[i] = (t3 - t1)*(t4 - t1)+(t5 - t3)*(t3 - t2)/(t5 - t2);
              gamma[i] = (t3 - t2)*(t3 - t2)/(t5 - t2);
              alfa[i] /= (t4 - t2);
              beta[i] /= (t4 - t2);
              gamma[i] /= (t4 - t2);
        }
        tridiag(alfa,beta,gamma,dato,control+1,nodato);

        for (i=0; i<3; i++) {
              control[0][i] = control[1][i];
              control[nodato+1][i] = control[nodato][i];
        }
        free(alfa);
        free(beta);
        free(gamma);
}
```

El Algoritmo 5.4 PuntoLongArc() regresa un punto pt sobre el spline, el cual tiene una longitud de arco s donde s pertenece a [0,1]. La rutina SetLong(), que es llamada inicialmente, establece la tabla de longitudes de segmento llamada seg_long[] de las longitudes acumuladas. La función LongArc() regresar la longitud del arco del segemento de la curva actual a través del intervalo [uinic,ufin]. La función SegCoef() establece los coeficientes relevantes para el segemento actual, indexado por i, para ser usado por la función ArcIntegra().

Algoritmo 5.4: PuntoLongArc()

```
float long_total;
float seg_long[nu_max];

void puntolongarc(spline,s,pt);
float spline[3];
float s;
float pt[3];
{
        float longarc();
        float ul=0;
              ur=1;
              ua;
        float segmento_dist,
              sa;
        int segmento_nu=0;

        /*Encuentra el segmento de curva donde se está s*/
        do segmento_nu++;
        while (seg_long[segmento_nu] < s);
        segmento_nu--;
        segmento_dis = long_total*(s-seg_long[segmento_nu]);
```

```
        /*Establecer los coeficientes para el segmento */
        segcoef(spline,segmento_nu);

        /*Buscar el segmento requerido*/
        do {
              ua = (ul + ur)/2;
              sa = longarc(ul,us);

              if (segmento_dist < sa+total_s)
                    ur = us;
              else {
                    ul = us;
                    segemento_dist -= sa;
              }
        } while ((segmento_dist>sa+total_s)||(segmento_dist<sa -
                    total_s));
        ObtenerPuntoSp(us,pt);
}

float longarc(us,ue)
float us,ue;
{
        float arcintegra();
        float h,sum,u;
        int i;

        /*Cálculo de la longitud de arco usando la regla extendida de
          Simpson */
        h = (ue-us)/(float)iter_cont;
        sum=0;
        u = us + h;

        for (i=2; i<iter_cont; i++) {
              if (!(i&1)) sum += 4.0*arcintegra(u);
              else sum += 2.0*arcintegra(u);
              u += h;
        }
        return(h*arcintegra(us)+sum+arcintegra(ue))/3.0;
}

void setlong(spline,npuntos)
float spline[3];
int npuntos;
{
        int i,nu_segmentos;
        float arclong,arclongi();

        nu_segmentos = npuntos - 3;
        arclong = 0;
        seg_long[0]=0;
        for (i=0; i<nu_segmentos; i++) {
              segcoef(spline,i);
              arclong += arclongi(0,1);
              seg_long[i+1] = arclong;
        }
        long_total = arclong;
        for (i=0; i<=nu_segmentos; i++) {
              seg_long[i] /= long_total;
}
```

# 5.4.3 CATMULL-ROM SPLINE

Un método para clasificar las curvas spline es dividirlas en dos categorías: aquellas que pasan a través de sus puntos de control, y las que no. La primera es conocida como interpolación spline y la segunda como aproximación spline. El Catmull-Rom spline es una interpolación spline que utiliza una técnica de mezcla parabólica.

Como una ilustración, considere los puntos de control (1,0), (0,1), (1,3), (2,1) y (3,2) en 2-D. Estos pueden ser substituidos en la siguiente matriz para generar coordenadas *x* e *y* del spline cúbico. Hay que recordar que el primer segmento de la curva es producido tomando los primeros cuatro puntos de control, y el segundo segmento es desarrollado dejando afuera el primer valor de control, y seleccionando los siguientes cuatro puntos.

**Tabla 5.7**

| *x* | *y* |
|---|---|
| 1.000 | 1.000 |
| 1.000 | 1.167 |
| 1.167 | 1.833 |
| 1.833 | 1.833 |
| 1.833 | 1.000 |
| 1.167 | 0.167 |
| 1.000 | 0.000 |

Como una ilustración, considere los puntos de control (1,0), (0,1), (1,3), (2,1) y (3,2) en 2-D. Estos pueden ser substituidos en la siguiente matriz para generar coordenadas *x* e *y* del spline cúbico. Hay que recordar que el primer segmento de la curva es producido tomando los primeros cuatro puntos de control, y el segundo segmento es desarrollado dejando afuera el primer valor de control, y seleccionando los siguientes cuatro puntos.

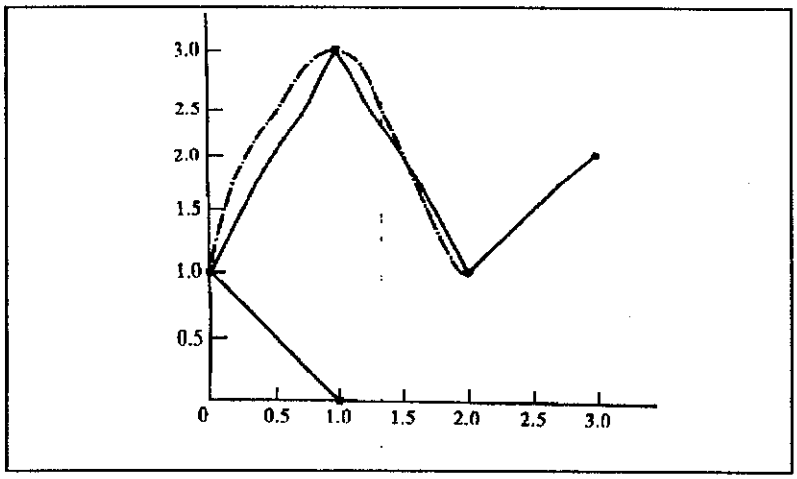
$$P(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1.0 & -2.5 & 2.0 & -0.5 \\ -0.5 & 0.0 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix},$$

donde $P(t)$ tendrá coordenadas $(x,y)$ dado que $P_1$, $P_2$, $P_3$ y $P_4$ que son cuatro puntos de control en 2-D. Como antes, $t$ varía entre 0 y 1 para cada segmento de curva.

La Tabla 5.8 muestra los valores correspondientes de $P(t)$ para los dos segmentos de curva para diferentes valores de $t$, y la curva se muestra en la Figura 5.11. Para una curva en el espacio 3-D, solamente se agrega la coordenada-z.

**Tabla 5.8**

| | *Segmento 1* | | *Segmento 2* | |
|---|---|---|---|---|
| *t* | *x* | *y* | *x* | *y* |
| 0.0 | 0.000 | 1.000 | 1.000 | 3.000 |
| 0.1 | 0.019 | 1.175 | 1.100 | 2.949 |
| 0.2 | 0.072 | 1.400 | 1.200 | 2.808 |
| 0.3 | 0.153 | 1.653 | 1.300 | 2.600 |
| 0.4 | 0.256 | 1.920 | 1.400 | 2.344 |
| 0.5 | 0.375 | 2.188 | 1.500 | 2.063 |
| 0.6 | 0.504 | 2.440 | 1.600 | 1.776 |
| 0.7 | 0.637 | 2.663 | 1.700 | 1.506 |
| 0.8 | 0.768 | 2.840 | 1.800 | 1.272 |
| 0.9 | 0.891 | 2.958 | 1.900 | 1.097 |
| 1.0 | 1.000 | 3.000 | 2.000 | 1.000 |




Dados cuatro puntos de control, el Catmull-Rom spline es capáz de derivar a spline entre los dos puntos internos. En este diagrama cinco puntos de control han sido usados para crear los dos segmentos internos con inclinación contínua.

# CAPITULO 6

# AMBIENTE DE ANIMACIÓN EN 3-D

## INTRODUCCIÓN

La parte central de una secuencia de animación, ya sea usando técnicas tradicionales o gráficas por computadora, es la necesidad de contar una historia que consiste de una secuencia de dibujos. También contiene otra información relacionada con los tiempos de movimientos y tiempos de duración de una escena, además de las ilustraciones necesarias para el animador con imágenes claras desde cualquier ángulo. En el caso de las técnicas tradicionales donde las imágenes son dibujadas a mano, el animador debe ser capaz de anticipar el efecto visual final. En la animación por computadora, los cambios de última hora en movimientos de cámara, niveles de luz y cambios de color, son posibles de hacer.

En las siguientes secciones de este capítulo, se verá que la animación tradicional es diferente a la creada en el mundo virtual de 3-D utilizando gráficas por computadora. Una característica importante de los dibujos hechos a mano es su técnica rápida -algunas marcas hechas sobre el papel son suficientes para crear un dibujo animado, y repitiendo este proceso rápidamente varias veces generará un dibujo animado con movimiento.

Comparado lo anterior con las gráficas por computadora, antes de realizar un modelo en 3-D, este tiene que ser construido; debe ser analizado dentro del sistema mundial de coordenadas (WCS, por sus siglas en inglés), el uso de una cámara virtual para observar la animación. De alguna forma, el modelo debe ser manipulado para realizar los movimientos deseados, y finalmente, las imágenes deben ser guardadas en una cinta de video o película para ser vistas.

La naturaleza virtual de las gráficas por computadora, donde los objetos, atributos, posiciones, movimientos, luces y cámaras son controlados por parámetros. La computadora proporciona un mundo imaginario donde cualquier cosa puede suceder, si algún programa lo hace. Para crear este mundo imaginario, hay que revisar el concepto de las coordenadas cartesianas y construir un frame de referencia para todo trabajo posterior.

## 6.1 SISTEMAS CARTESIANO Y POLAR

Tal vez el método más simple para definir la posición y orientación de una cámara es especificando la posición visual y un punto de enfoque usando las coordenadas cartesianas, donde $(x_c, y_c, z_c)$ fijan la posición de visión C, y $(x_f, y_f, z_f)$ localiza el punto de enfoque F. Estos dos puntos identifican cómo el eje es posicionado en el espacio, y no el ángulo de rotación. Una forma simple de introducir un rodaje es aplicar una rotación para las coordenadas de la cámara cuando estos son calculados.

Una de las grandes desventajas con la definición cartesiana es que los movimientos rotacionales de la cámara, que son muy comunes, resultan difíciles de especificar, por lo tanto, el uso de coordenadas polares proporcionan dos formas útiles de operación. La primera alínea el CCS con el WCS, y está expuesto a dos rotaciones para proporcionar un giro e inclinación, y es trasladada por $(x_c, y_c, z_c)$. El rodar puede ser realizado rotando la proyección perspectiva final, y en la misma forma será rotada, o puede ser incluida como en eje básico de rotación. Este método de localización es útil cuando se necesita que la cámara esté animada a lo largo de una trayectoria conocida, y tenga que moverse a través de rotaciones angulares específicas.

Es importante que la cámara esté dirigida a punto conocido, y por eso se usa un segundo método, el que requiere que el CCS esté inicialmente alineado con WCS, y luego expuesto a dos rotaciones para proporcionar un grio e inclinación. Cuando el primer método necesita de cinfo parámetros: $(x_c, y_c, z_c)$ y dos ángulos, el segundo necesita de seis parámetros: $(x_f, y_f, z_f)$, la distancia y dos ángulos.

La última técnica permite a una cámara rastrear un objetivo en movimiento por medio de una distancia fija, como el punto de enfoque puede ser cualquier punto en el objetivo; la cámara es forzada a seguir al objetivo. La Figura 6.1 ilustra ambos métodos.

## 6.1.1 ANIMACIÓN DE LA CÁMARA

Las técnicas del cine moderno son restringidas por el tamaño físico de la cámara y los métodos manuales para ajustar el enfoque. Cada cierto tiempo la cámara es movida a una posición nueva para volver a enfocar, el recálculo de los niveles de luz, las áreas de sombras ajustadas para asegurar que la película esté correcta. Para empezar, no hay necesidad de enfocar; no hay lentes asociados con la cámara de la computadora -además el efecto puede ser simulado. Segundo, la cámara no tiene masa ni tamaño, y puede ser colocada en cualquier parte y movida a cualquier velocidad.

En la definición cartesiana de la cámara, es necesario saber la localización inicial y final y el punto de enfoque de la cámara. Si la posición de la cámara está definida por el punto C con coordenadas $(x_c, y_c, z_c)$, y punto de enfoque F con coordenadas $(x_f, y_f, z_f)$, entonces puede ser movida de la posición $C_1(x_{c1}, y_{c1}, z_{c1})$ a $C_2(x_{c1}, y_{c1}, z_{c1})$ con un punto de vista en común con coordenadas interpoladas linealmente de la siguiente forma:

$$x_c = (1 - t)x_{c1} + tx_{c2}$$
$$y_c = (1 - t)y_{c1} + ty_{c2}$$
$$z_c = (1 - t)z_{c1} + tz_{c2}$$

Donde *t* es un parámetro que varía de 0 a 1 como el número de frame actual cambia de un key-frame a otro. Este movimiento hace que la cámara permanezca dirigida al mismo punto durante toda la maniobra, y como la cámara se mueve en línea de $C_1$ a $C_2$. De hecho, las posiciones relativas de $C_1$ y $C_2$ determinan dónde se hará el movimiento de rastreo. La Figura 6.2 ilustra dos movimientos usando esta técnica.

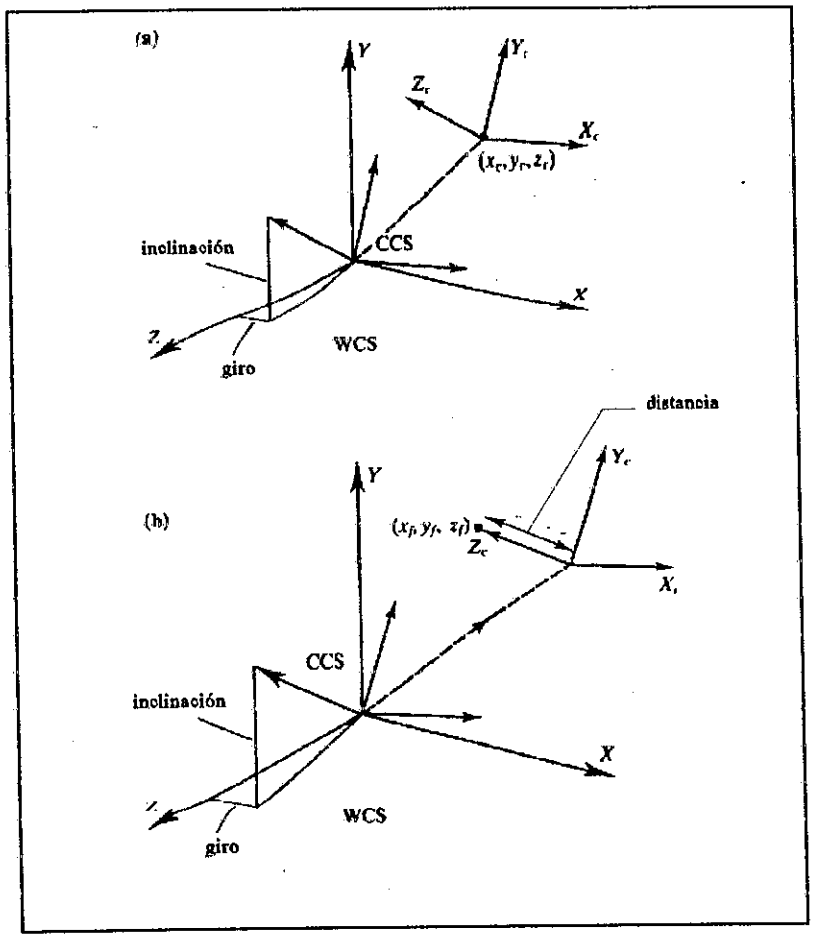



**Figura 6.1:** una cámara virtual puede ser colocada en el WCS usando una variedad de técnicas. En (a) el sistema de ejes de la cámara está inicialmente alineada con el WCS y luego es rotada para introducir un giro e inclinación; finalmente es trasladada por $(x_c, y_c, z_c)$ a un lugar conveniente. Cuando se necesita que a la cámara se le especifique una distancia específica de algún punto de enfoque, las operaciones de giro e inclinación son seguidas de una traslación, como se muestra en (b).

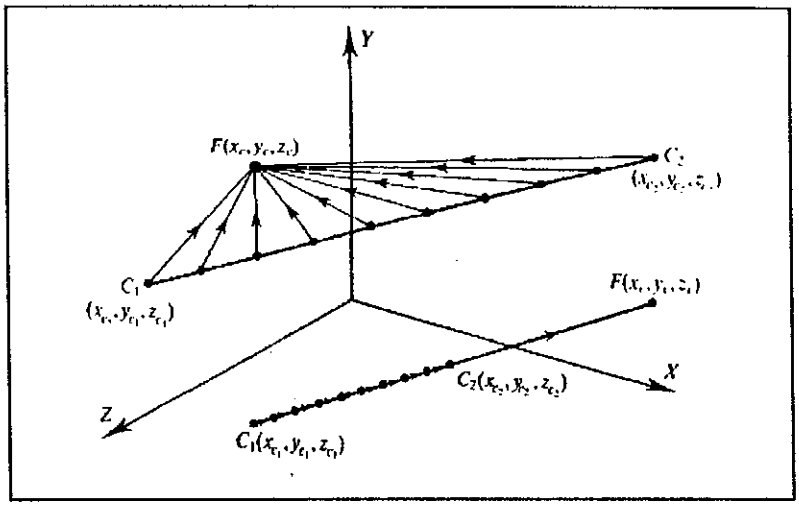



**Figura 6.2**: estos dos movimentos de cámara pueden ser simulados manteniendo el punto de enfoque fijo en $(x_c, y_c, z_c)$ y mover la cámara de $C_1$ a $C_2$ usando interpolación lineal.

El movimiento rotacional se controla mejor usando coordenadas polares; se considera el caso cuando la cámara necesita un pivote alrededor de ella de 360°. Mientras está rotando en el plano horizontal, se inclina hacia arriba en un ángulo de 30°.

Una forma efectiva para mezclar valores ya sea el inicial y el final, es usar una función senoidal, o la técnica Hermite, descrita en el capítulo anterior. Analizando esta técnica, el valor mezclado es dado por:

$$N(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

donde la última columna de números (0,1,0,0) está compuesta de (0,1), que es el rango numérico de la mezcla, y (0,0) es la inclinación al inicio y al final. Esta operación podría ser representada por la función:

```
N(t) = hermite(0,1,0,0,t)
```

Esta regresa el valor de N(t). La tabla 6.1 muestra estos valores para diferentes valores de *t*

**Tabla 6.1**

| *t* | *hermite* |
|---|---|
| 0.0 | 0.000 |
| 0.1 | 0.028 |
| 0.2 | 0.104 |
| 0.3 | 0.216 |
| 0.4 | 0.352 |
| 0.5 | 0.500 |
| 0.6 | 0.648 |
| 0.7 | 0.784 |
| 0.8 | 0.896 |
| 0.9 | 0.972 |
| 1.0 | 1.000 |

Una revisión de los valores de hermite(0,1,0,0,t) confirma que se mueven desde cero muy despacio; pasa de 0.5 cuando *t* es igual a 0.5; y se mueve despacio hacia 1 cuando *t* se acerca a 1.

Si el ángulo horizontal de la cámara es controlado por alpha, y el ángulo de elevación por beta, la orientación de la cámara puede ser descrito por:

```
alpha = 360hermite(0,1,0,0,t)
beta  = 30hermite(0,1,0,0,t)
```

El parámetro *t* puede ser calculado por el actual frame F y los valores del key-frame $F_1$ y $F_2$ usando la siguiente expresión:

$$t = \frac{F - F_1}{F_2 - F_1}$$

Substituyendo F, $F_1$ y $F_2$ para obtener *t*, lo cual produce hermite(0,1,0,0,t), permite calcular a alpha y beta; éstos son substituidos en una definición polar de la cámara para ocasionar que la cámara se mueva como se muestra en la Figura 6.3.

Otra situación en que son útiles las coordenadas polares es cuando el punto de enfoque es conocido, y la cámara deber permanecer a una distancia fija de él, para realizar una rotación horizontal y elevacional. Esto hace que la cámara rastree un objeto en movimiento a una distancia dada, y girar a su alrededor.

Se puede ver cuán importante es para el animador relacionarse con el ambiente de animación a través de gráficas interactivas por computadora; esto permite comunicar toda la información al sistema por medio de operaciones de menú y algunos comandos gráficos. El siguiente ejemplo puede mostrar cómo una curva en el espacio puede ser usada para actuar como un trayecto de vuelo para la cámara -puede ser cualquier clase de curva ya sea B-splines y Bézier.

Si se elige una curva de Bézier cúbica, entonces los dos puntos centrales se pueden utilizar para 'encorvar' la curva a una forma deseada. Se necesita una función para obtener las coordenadas *x*, *y* e *z* de los puntos de control usando un parámetro *t*. Dados los puntos de control $C_1(x_{c1},y_{c1},z_{c1})$, $C_2(x_{c1},y_{c1},z_{c1})$, $C_3(x_{c3},y_{c3},z_{c3})$ y $C_4(x_{c4},y_{c4},z_{c4})$, y la función de bezier($v_1,v_2,v_3,v_4$,t), donde $v_1$, $v_2$, $v_3$ y $v_4$ son los valores de control y *t* es el parámetro general, entonces las coordenadas de cualquier punto C(x,y,z) sobre la curva pueden ser especificados por:

$$\begin{aligned} x &= \text{bezier}(x_{c1},x_{c2},x_{c3},x_{c4},t) \\ y &= \text{bezier}(y_{c1},y_{c2},y_{c3},y_{c4},t) \\ z &= \text{bezier}(z_{c1},z_{c2},z_{c3},z_{c4},t) \end{aligned}$$

Del punto C(x,y,z) puede resultar la posición de la cámara, pero se necesita un punto de enfoque, que puede ser una posición fija, o un valor interpolado entre dos posiciones. Podría ser otra posición sobre una segunda curva de Bézier; aunque, algo importante es cuando el punto de enfoque está localizado en cualquier parte a lo largo de la misma curva. Esta posición puede ser determinada incrementando *t* en alguna cantidad, 0.1 por ejemplo; esto significa que el punto de enfoque $F(x_f,y_f,z_f)$ puede ser calculado por:

$$\begin{aligned} x &= \text{bezier}(x_{c1},x_{c2},x_{c3},x_{c4},t + 0.1) \\ y &= \text{bezier}(y_{c1},y_{c2},y_{c3},y_{c4},t + 0.1) \\ z &= \text{bezier}(z_{c1},z_{c2},z_{c3},z_{c4},t + 0.1) \end{aligned}$$

Esto implica que la posición de la cámara puede existir para valores de parámetros entre 0.0 y 0.9, y el punto de enfoque puede existir para valores de parámetros entre 0.1 y 1.0, por lo que no se pueden imponer restricciones a esta técnica. Esta geometría es mostrada en la Figura 6.3.

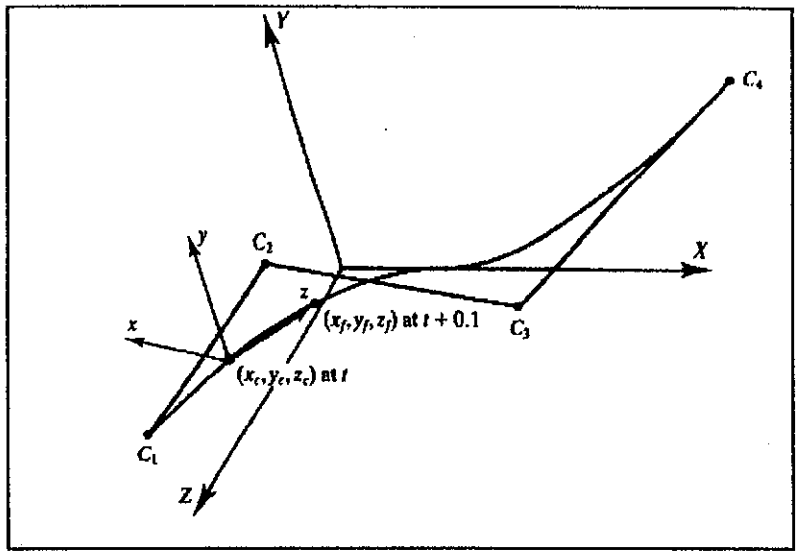



**Figura 6.3**: la curva de Bézier cúbica localiza la posición de la cámara virtual y el punto de enfoque para los valores de $t$ y $t+0.1$ respectivamente. La cámara tiene posiciones válidas entre los rangos $0 <= t <= 0.9$; mientras que el punto de enfoque está entre $0.1 <= t <= 1.0$. La cámara puede moverse fuera de éstos límites, pero se perderá el control directo sobre su velocidad.

# 6.2 ANIMACIÓN DE LOS ATRIBUTOS DEL OBJETO

Se tiene el ejemplo de animar un cubo rotando en el espacio: primero, se necesita la geometría que forman los ocho vértices, seis lados y doce bordes. Esto se puede desarrollar interactivamente, creado manualmente, o generado por un sistema; depende del ambiente de animación. Cuando se crea el cubo, otro número de factores deben ser suministrados para que se realice la animación.

Por ejemplo, ¿dónde se pondrá el cubo en el WCS? ¿Dónde se colocará la cámara, y ¿cuál será su punto de enfoque? ¿Cuál es el color del cubo? ¿Dónde se encuentran las fuentes de luz y cuáles son sus colores? ¿Es el cubo un reflector difuso o especular? ¿Se rotará el cubo alrededor de los ejes *x*, *y* e *z*? ¿Cuál es la rotación angular aplicada al cubo por cada imagen? ¿Aumentará la rotación angular del cubo y luego mantener una velocidad angular?. Estas y muchas otras preguntas se hace el animador. El sistema de animación puede entonces permitir al animador explorar un infinito número de variaciones.

Para crear la animación, el sistema utilizará las instrucciones proporcionadas por el animador y crear la primera vista del cubo. Entonces rotará las coordenadas de los ocho vértices por una cantidad especificada y representar otra vista. Modificando constantemente la geometría del cubo y representar una nueva imagen, se creará una secuencia diferente de imágenes. Cuando la representación no permite un despliegue en tiempo real, las imágenes podrán ser transferidas a una cinta de video o disco de video, y desplegarlo a la velocidad del video.

Pero si el color del cubo ha cambiado de rojo luego a verde y azul, y regresa a color rojo en una rotación. Esto se puede lograr instruyendo al sistema para asociar un valor inicial del matiz, saturación y evaluar el cubo para modificarlo en cada frame. Hay que recordar que los números decimales codifican los rangos de parámetros de color de cero a uno, con lo que el matiz 0.0 es igual a rojo, 0.333 es igual a verde y 0.666 igual a azúl, y 1.0 regresa a rojo. El primer frame empieza con un matiz de 0.0, entonces se representará un cubo rojo. Antes de que se produzca un frame, el parámetro de matiz del cubo es aumentado por un valor, tal que en una rotación el valor haya llegado a 1.0. Cuando la representación cambia de color en la siguiente imagen, existirá un nuevo valor de matiz, lo que asegura que cada imagen representada aparece con un nuevo color. Pero, si se permite que el cubo rote contínuamente, el valor del matiz excederá

de 1.0.

El ejemplo anterior muestra que cualquier atributo puede ser cambiado mientras el cubo está rotando.

Así que alterando los atributos de un objeto, no parece ser una operación compleja, aunque puede ser difícil cambiarlos en tal forma que la animación sea natural. El mundo virtual de las gráficas por computadora básicamente dice 'crear un mundo de objetos, luces y cámaras, y al manipularlo con la ayuda de varias herramientas, trucos y engaños, se tendrá un buen resultado.

El principal problema de la animación de objetos no es tanto lo que es animado, sino qué técnicas son usadas para realizar la animación, y cómo el animador se relaciona con ellas. Otro problema que se presenta es si éstas técnicas pueden ser usadas para cada tipo de técnica para modelar.

## 6.2.1 FORMAS DE INTERPOLACIÓN

La interpolación entre un par de números es un ejercicio común y puede ser realizado con una interpolación linea o una interpolación no lineal como la parabólica o la interpolación de Hermite. Estos números representan coordenadas cartesianas, entonces los valores interpolados deberán codificar alguna forma geométrica. Esta es la idea de la interpolacíon de formas u objetos que permite figuras en 2-D u objetos en 3-D que sufren cambios de forma o de objeto. La técnica se entiene mejor con un ejemplo en 2-D que se traslada a un dominio en 3-D agregando una coordenada-z.

La tabla 6.2, muestra las coordenadas $x$ e $y$ de dos figuras A y B, y por conveniencia tienen el mismo número de vértices. Si se utiliza una interpolación lineal con el parámetro $t$, entonces cualquier figura interpolada $I$ puede ser calculada por:

$$I(t) = (1 - t)A + tB,$$

donde esto es aplicado para las coordenadas $x$ e $y$. En la misma tabla, se muestra la figura interpolada calculada por I(0.5), y las tres figuras A, B e I(0.5) se muestran en la Figura 6.4.

En la figura, se puede ver que los vértices interpolados están situados sobre las líneas que conectan los correspondientes vértices, y que cambiando el valor de $t$ se puede crear cualquier figura interpolada. Se puede ver que la secuencia de vértices de las figuras A y B es importante; por si una secuencia es revertida, entonces las figuras interpoladas sufrirán un desvío, lo cual no es deseable. Además, como la longitud de la figura A es de 2.4142 y la de la figura B es de 2.0, entonces la longitud de la figura intermedia será, en general, diferente.

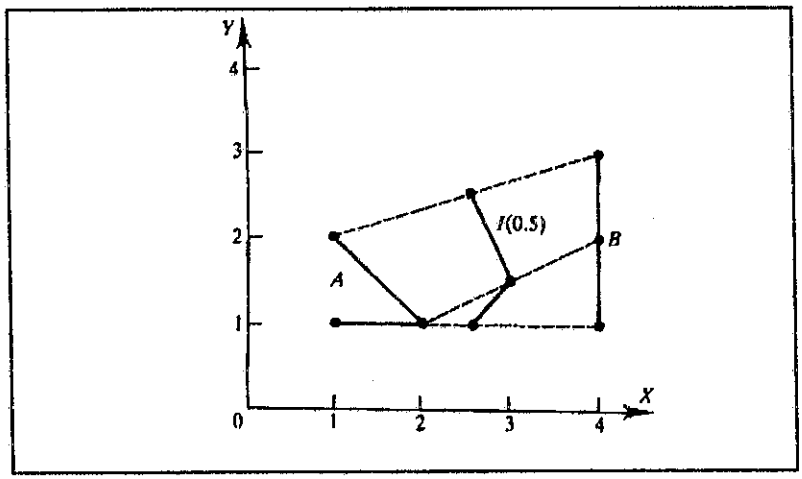



**Figura 6.4**: las figuras A y B han sido interpoladas linealmente para producir la figura I, esto sucede cuando el parámetro de control es igual a 0.5.

**Tabla 6.2**

| Vértice No. | A | | I(0.5) | | B | |
|---|---|---|---|---|---|---|
| | x | y | x | y | x | y |
| 1 | 1.0 | 1.0 | 2.5 | 1.0 | 4.0 | 1.0 |
| 2 | 2.0 | 1.0 | 3.0 | 1.5 | 4.0 | 2.0 |
| 3 | 1.0 | 2.0 | 2.5 | 2.5 | 4.0 | 3.0 |

Cuando las figuras principales no tienen el mismo número de vértices, por ejemplo, si una figura tiene 10 vértices y la otra 120 vértices, se necesita una estrategia para decidir cómo un vértice en una figura es mapeado dentro de una posición determiada en la otra figura. Esto se puede basar en la posición relativa del vértice o de la longitud de la cuerda en las figuras. Surge otra complicación, cuando dos figuras principales están compuestas de diferentes números de líneas de segmentos separadas. En este caso, se necesita una estrategia para decidir cuál segmento se partirá en pequeños segmentos para satisfacer los requerimientos de la figura final. Estos problemas se deben tomar en cuenta cuando se crea el sistema de computación.

El ejemplo anterior trabaja en dos dimensiones, pero ¿trabajará para tres dimensiones? La respuesta es sí, si sólo se considera una cádena de vértices; lo único es agregar la tercera coordenada-z para los cálculos.

## 6.2.2 INTERPOLACIÓN DE LA POSICIÓN

Se sabe que cualquier conjunto de datos puede ser colocado en el espacio agregando un valor a las coordenadas *x*, *y* e *z*, y dado un objeto A que ha sido modelado, tal que sus coordenadas implican un origen en el centro de su volumen, puede ser movido de $P_1(x_1,y_1,z_1)$ a $P_2(x_2,y_2,z_2)$ por la siguiente operación de matrices:

$$\begin{bmatrix} A'_x \\ A'_y \\ A'_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x(t) \\ 0 & 1 & 0 & y(t) \\ 0 & 0 & 1 & z(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \\ 1 \end{bmatrix}$$

donde:

$(A'_x,A'_y,A'_z)$ son las coordenadas trasladadas de A por cualquier valor de t;
$(A_x,A_y,A_z)$ son las coordenadas originales de A;
$(x(t),y(t),z(t))$ son las traslaciones que se aplicarán a A; y
$x(t) = (1 - t)x_1 + tx_2$
$y(t) = (1 - t)y_1 + ty_2$
$z(t) = (1 - t)z_1 + tz_2$,

donde $0 <= t <= 1$. Esta técnica mantiene las coordenadas originales de A y calcula y traslada la versión A', la cual mueve de $P_1$ a $P_2$ en igual número de pasos como *t* es cambiado de 0 a 1. Una alternativa es modificar los datos de las coordenadas del objeto como sus coordenadas son modificadas en cada frame de la animación por ciertos incrementos. Cuando el movimiento es lineal, el incremento se calcula fácilmente, pero un movimiento no lineal significa que son diferentes para cada frame, y se debe asegurar que el objeto sea adaptado a su posición final. Esto se puede ver, examinando la coordenada-x de un movimiento de traslación.

# 6.2.3 INTERPOLACIÓN DE LA ROTACIÓN

La rotación en dos dimensiones es fácil de imaginar y ocurre alrededor de un punto, en cambio un objeto en tres dimensiones puede estar sujeto a cualquier combinación de los tres ejes de rotación, lo que dificulta su visualización.

Como ejemplo, se considera un cubo modelado tal que su centro esté localizado en el origen, el que debe ser animado con diez frames, tal que en su frame diez su velocidad angular alrededor del eje-y vertical da una revolución cada segundo. Se asume que la velocidad del frame es de 25 Hz y debe rotar 360° en un segundo, entonces la velocidad angular por frame debe ser:

$$14.4 = 360/25[^\circ/\text{frame}]$$

Una vez más, la función de Hermite puede ser usada para generar los pasos no lineales para las rotaciones angulares teniendo una inclinación inicial de cero grados, y una inclinación final de 40° (hermite(0,14.4,0,40,t). Esto produce los diferentes valores de la Tabla 6.3.

Una revisión de los valores de la tabla muestra que como $t$ se incrementa en pasos lineales, la función Hermite aumenta en pequeños incrementos hasta un incremento final de 3.643 (14.4 - 10.757). Si esto es usado para rotar las coordenadas de un objeto en cada frame, la secuencia animada mostrará al objeto inferir de los demás, ganando cierto tiempo hasta que el décimo frame ha inferido en 14.4° por frame, lo que resulta en una velocidad angular constante.

**Tabla 6.3**

| *t* | *hermite* |
|---|---|
| 0.0 | 0.000 |
| 0.1 | 0.043 |
| 0.2 | 0.218 |
| 0.3 | 0.590 |
| 0.4 | 1.229 |
| 0.5 | 2.200 |
| 0.6 | 3.571 |
| 0.7 | 5.410 |
| 0.8 | 7.782 |
| 0.9 | 10.757 |
| 1.0 | 14.400 |

La animación final que empieza en el frame $F_{start}$ puede ser representada por la siguiente operación de matrices:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(h) & 0 & \text{sen}(h) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(h) & 0 & \cos(h) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

donde:

(x,y,z) es cualquier vértice del objeto
F es el actual frame dentro de un rango permitido

y

$t = (F - F_{start})/10$
$h = \text{hermite}(0,14.4,0,40,t)$

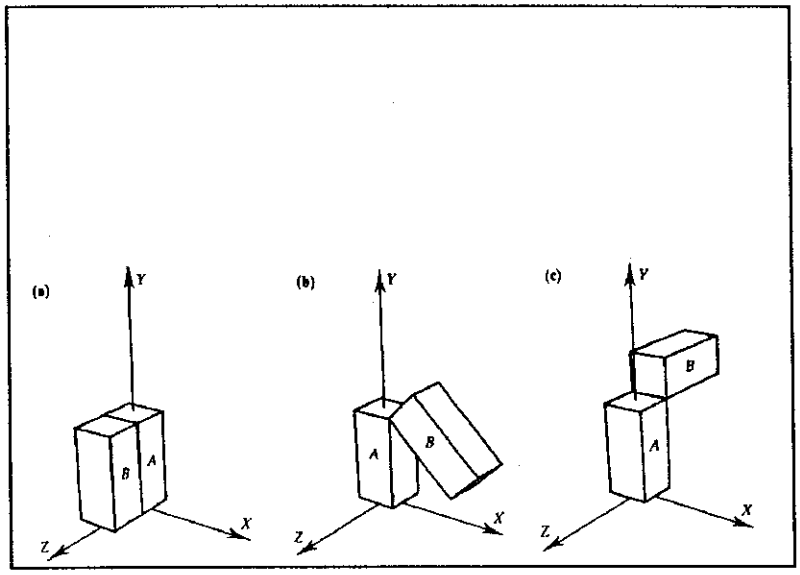
Para otro ejemplo, considérense los problemas de animar una simple estructura articulada compuesta de dos miembros A y B, tal que A es libre de

rotar alrededor de el eje-y vertical, y B está acoplado A, para permitir movimientos verticales elevacionales. Esta configuración se muestra en la Figura 6.5. La animación necesita que el bloque B rote desde su posición original como se muestra en la Figura 6.5a a su posición final mostrada en la Figura 6.5e, al mismo tiempo que todo el conjunto está rotando alrededor del eje-y.

## 6.2.4 INTERPOLACIÓN DE LOS ÁNGULOS DE DESVIACIÓN, INCLINACIÓN Y ONDULACIÓN

La orientación de un objeto, cámara o fuente de luz puede ser especificada con los ángulos de desviación, inclinación y ondulación; estos son ángulos individuales rotados alrededor de los ejes *x*, *y* e *z* para crear alguna rotación angular compuesta. Como se muestra en la Figura 6.6, la desviación es una rotación alrededor del eje-y; la inclinación es una rotación alrededor del eje-x, y la ondulación es una rotación alrededor del eje-z.

Se sabe que las rotaciones no son conmutativas, es decir, que el objeto está sujeto a tres matrices de rotación en la secuencia de A, B seguido de C entonces, en general, esto no será equivalente a C, B seguido de A. El orden para desarrollar una rotación en 3-D adoptará la costumbre de aplicar matrices de rotación en la secuencia de ondulación, inclinación y por desviación. Las matrices para estas rotaciones son las siguientes:



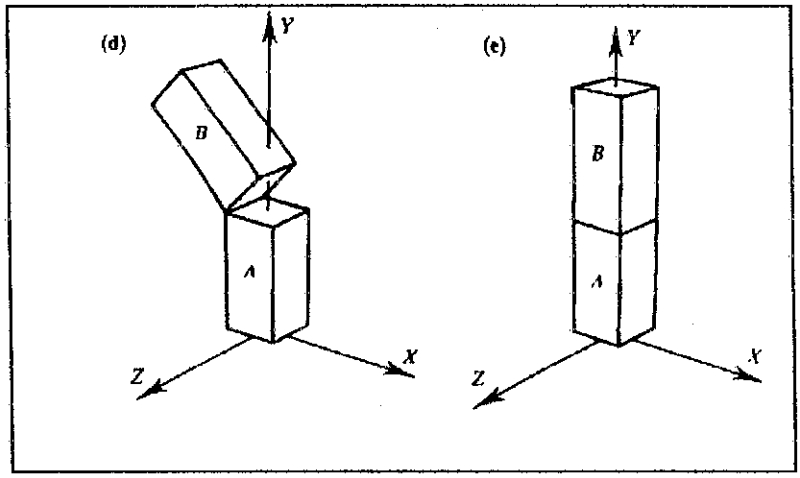



**Figura 6.5:** estos cinco frames de animación muestran dos objetos A y B en (a), los cuales son rotados simultáneamente alrededor de un eje R en el eje-y vertical.

$$\begin{bmatrix} \cos(roll) & -\text{sen}(roll) & 0 & 0 \\ \text{sen}(roll) & \cos(roll) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(pitch) & -\text{sen}(pitch) & 0 \\ 0 & \text{sen}(pitch) & \cos(pitch) & 0 \\ 0 & 0 & 0 \quad 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos(yaw) & 0 & \text{sen}(yaw) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(yaw) & 0 & \cos(yaw) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

y cualquier punto (x,y,z) puede ser rotado a otra posición (x',y',z') aplicando las matrices de la siguiente forma:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = [yaw][pitch][roll] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

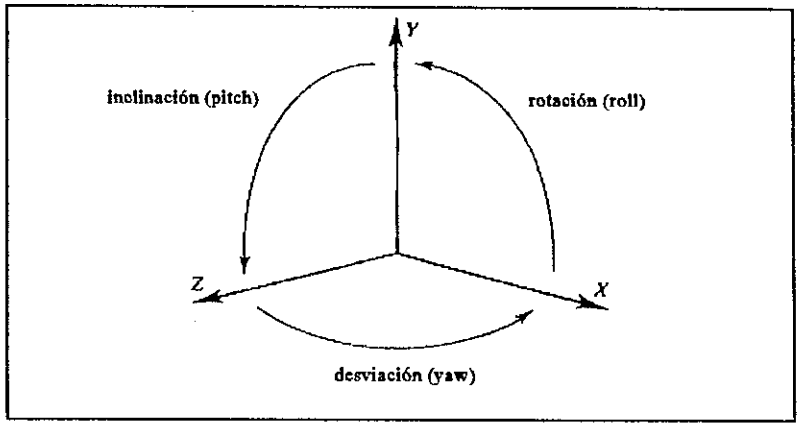



**Figura 6.6**: existen diversas formas para obtener la inclinación, ondulación y desviación para rotaciones de ejes.

# 6.2.5 INTERPOLACIÓN DE QUATERNIONS

En el siglo XIX, Sir William Rowan Hamilton empezó su investigación para descubrir la forma de representar el radio de dos vectores 3-D como un tercer vector: si existiera dicha técnica, se proporcionaría una forma de rotar un vector multiplicándolo por otro. Por 15 años investigó dicho sistema, y en 1843 se imaginó algo parecido a cuatro dimensiones, para lo cual le dio el nombre de 'quaternions'. Los matemáticos actuales saben que dicho sistema sólo trabaja con dos, cuatro u ocho componentes, y que es imposible un equivalente de 3-D.

Se sabe que un vector 3-D posee tres escalares que representan los tres componentes espaciales cartesianos, aunque un quaterinion combina un vector 3-D con una cantidad escalar extra. Esto puede ser usado para describir una 4-tupla.

En animación, la orientación de un objeto o cámara es descrita usando ángulos de ondulación, inclinación y desviación que tienen algunas desventajas. La forma quaternion proporciona un mecanismo simple para rotar objetos y cámaras alrededor de un eje; además, dada una secuencia de quaternions, es posible interpolarlos para crear rotaciones complejas controladas. A continuación, se muestra cómo los quaternions pueden ser usados para rotar puntos en el esapcio 3-D e interpolar los ángulos de orientación.

En general, un queternion $\mathbf{q}$ está definido por un escalar y un vector 3-D donde $\mathbf{v}$ tiene tres componentes:

$$\mathbf{q} = [s,\mathbf{v}]$$

Dado dos quaternions $\mathbf{q}_1$ y $\mathbf{q}_2$:

$$\mathbf{q}_1 = [s_1,\mathbf{v}_1]$$
$$\mathbf{q}_2 = [s_2,\mathbf{v}_2]$$

Entonces, sumados y multiplicados entre sí se obtiene:

$$\mathbf{q}_1 + \mathbf{q}_2 = [s_1 + s_2,\mathbf{v}_1 + \mathbf{v}_2]$$
$$\mathbf{q}_1\mathbf{q}_2 = [(s_1s_2 - \mathbf{v}_1.\mathbf{v}_2),[s_1\mathbf{v}_1 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]]$$

donde $\mathbf{v}_1.\mathbf{v}_2$ es el producto punto de $\mathbf{v}_1$ y $\mathbf{v}_2$, y $\mathbf{v}_1 \times \mathbf{v}_2$ es el producto cruz. Nótese que en ambos casos el resultado es siempre un quaternion.

La magnitud de un cuaternion $\mathbf{q}$ está definido como:

$$|\mathbf{q}| = \sqrt{(x^2 + y^2 + z^2 + s^2)},$$

donde $s$ es el término escalar, $x$, $y$ e $z$ son los componentes del vector. El quaternion inverso está definido como:

$$\mathbf{q}^{-1} = \frac{[s,-\mathbf{v}]}{|\mathbf{q}|_2}$$

y también se puede mostrar que:

$$\mathbf{q}\mathbf{q}^{-1} = [1,[0,0,0]]$$

Si esto es aceptable, entonces se puede mostrar que un vector **v** puede ser rotado alrededor de un eje por la operación:

$$\mathbf{v}' = \mathbf{q}\mathbf{v}\mathbf{q}^{-1}$$

Aunque se quieren rotar puntos alrededor de un eje, esto no es un problema, porque las coordenadas de un punto pueden ser interpretados como la representación de componentes de un vector que pasan a través del origen. Dicho vector es llamado posición de vector. Además, si el punto que va a ser rotado está especificado como una posición de vector **A**, el cual es representado en la forma de quaternion como [0,**a**], entonces la siguiente operación:

$$\mathbf{q}[0,\mathbf{A}]\mathbf{q}^{-1}$$

hacer rotar a **A** por un ángulo alrededor de un eje que está representado por **q**. Cuando son usados quaternions unitarios, el quaternion **q** es igual a:

$$\mathbf{q} = [s,\text{sen}(\beta/2)\mathbf{u}]$$

y tiene las siguientes condiciones:

$$s = \cos(\beta/2)$$
**u** = es un vector unitario,

donde ß es el ángulo de rotación, y **u** es la dirección de la rotación del eje pasando a través del origen.

Por ejemplo, una rotación de desviación alrededor del eje-y está especificado como:

$$\mathbf{q} = [\cos(\beta/2),\text{sen}(\beta/2)[0,1,0]],$$

mientras una rotación de inclinación alredeodr del eje-x es:

$$\mathbf{q} = [\cos(\beta/2),\text{sen}(\beta/2)[1,0,0]],$$

y una rotación de ondulación alrededor del eje-z es:

$$\mathbf{q} = [\cos(\beta/2),\text{sen}(\beta/2)[0,0,1]]$$

donde ß es el ángulo de rotación. Nótese que la magnitud de **q** es siempre unitario como:

$$\text{sen}^2(\beta) + \cos^2(\beta) = 1$$

Como un simple ejemplo, considérese el punto (0,1,1), el cual tiene que ser rotado 90° alrededor del eje-y vertical. Se puede predecir sin la ayuda de quaternions que la respuesta será (1,1,0).

Primero, se transforma el punto que va a ser rotado a un quaternion como **P**[0,[0,1,1]] con un componente escalar cero. Segundo, el quaternion **q** que codifica el eje y ángulo de rotación es:

$$\mathbf{q} = [\cos(90/2),[0,\text{sen}(90/2),0]],$$

además, el quaternion inverso es:

$$q^{-1} = \frac{[\cos(90/2),[0,-\text{sen}(90/2),0]]}{|q|_2}$$

Como se ha usado un quaternion unitario $|q| = 1$, el denominador en el quaternion inverso puede ser ignorado, como es igual a 1.

El cálculo del quaternion rotado **P'** es expresado por:

$$P' = qPq^{-1},$$

donde el vector componente del quaternion **P'** mantendrá el punto rotado. La evaluación se hace en dos etapas que pueden ser realizadas por un pequeño programa.

**Etapa 1**

$$qP = [\cos(90/2),[0,\text{sen}(90/2),0]] \times [0,[0,1,1]]$$
$$[-\text{sen}(45),[\text{sen}(45),\cos(45),\cos(45)]]$$

**Etapa 2**

$$qPq^{-1} = [-\text{sen}(45),[\text{sen}(45),\cos(45),\cos(45)]] \times$$
$$[\cos(45),[0,-\text{sen}(45),0]]$$
$$P' = [0,[\text{sen}(90),1,\cos(90)]]$$
$$P' = [0,[1,1,0]]$$

El vector componente de **P'** confirma que el punto **P**(0,1,1) es rotado a (1,1,0).

Lo anterior puede ser visto muy difícil, pero cuando son implementados en un programa de computadora se vuelve una herramienta muy útil. Además, el especificar el eje de rotación como un vector es muy conveniente. Cuando se desea rotar un objeto alrededor de un eje que no pasa a través del origen, entonces el objeto primero es trasladado, tal que el eje pase a través del origen; entonces es rotado alrededor del eje.

Los quaternions se relacionan con los ángulos de ondulación, inclinación y desviación. Pero se debe tener cuidado como las acciones de desviación, inclinación y ondulación son relacionados con el sistema de ejes en 3-D. Por conveniencia, la desviación es una rotación alrededor del eje-y vertical; la inclinación será una rotación alrededor del eje-x horizontal; y la ondulación será una rotación alrededor del eje-z. Además, el espacio 3-D será hacia la derecha para que las rotaciones angulares positivas aparezcan como en el sentido contrario a las manecillas del reloj.

Si las rotaciones de desviación, inclinación y ondulación son representadas como quaternions como siguie:

$$q_{yaw} = [\cos(yaw/2),\text{sen}(yaw/2)[0,1,0]]$$
$$q_{pitch} = [\cos(pitch/2),\text{sen}(pitch/2)[1,0,0]]$$
$$q_{roll} = [\cos(roll/2),\text{sen}(roll/2)[0,0,1]],$$

entonces, pueden ser multiplicados para crear un simple quaternion **q**:

$$q = q_{yaw}q_{pitch}q_{roll}$$

Se ha descubierto que un quaternion realiza una rotación codificando los ejes y ángulos de rotación dentro de sus cuatro componentes, y si éstos son modificados, entonces los grados y ejes de rotación pueden ser controlados. Un quaternion que rota en grado cero es el quaternion identidad [1,(0,0,0)], y si estos componentes son interpolados con un segundo quaternion, entonces el quaternion intermedio puede ser usado para rotar un punto.

Para ilustrar esta técnica, se toma el quaternion de desviación como el segundo quaternion y se interpolan sus componentes con el quaternion de identidad como sigue:

$$q'[s,[x,y,z]] = (1 - t)[1,[0,0,0]] + t[\cos(yaw/2),\text{sen}(yaw/2)[0,1,0]],$$

donde:

```
s = (1 - t)1 + tcos(yaw/2)
x = (1 - t)0 + t0
y = (1 - t)0 + tsen(yaw/2)
z = (1 - t)0 + t0
```

y *t* es el parámetro de interpolación lineal, y *s*, *x*, *y* e *z* son los cuatro componentes del quaternion interpolado. Simplificado es:

```
s = (1 - t) + tcos(yaw/2)
x = 0
y = tsen(yaw/2)
z = 0,
```

lo que no es un quaternion unitario. Además, si esto es usado para rotar el punto P(0,1,1), entonces las coordenadas obtenidas se muestran en la tabla 6.4.

**Tabla 6.4**

| *t* | *dist* | *x* | *y* | *z* |
|---|---|---|---|---|
| 0.0 | - | 0.000 | 1.000 | 1.000 |
| 0.1 | 0.145 | 0.145 | 1.000 | 0.989 |
| 0.2 | 0.153 | 0.294 | 1.000 | 0.956 |
| 0.3 | 0.159 | 0.441 | 1.000 | 0.897 |
| 0.4 | 0.163 | 0.581 | 1.000 | 0.814 |
| 0.5 | 0.165 | 0.707 | 1.000 | 0.707 |
| 0.6 | 0.165 | 0.814 | 1.000 | 0.581 |
| 0.7 | 0.163 | 0.897 | 1.000 | 0.441 |
| 0.8 | 0.159 | 0.956 | 1.000 | 0.294 |
| 0.9 | 0.153 | 0.989 | 1.000 | 0.145 |
| 1.0 | 0.145 | 1.000 | 1.000 | 0.000 |

Como el parámetro *t* cambia de 0 a 1 en pasos de 0.1, las coordenadas *x*, *y* e *z* se mueven de (0,1,1) a (1,1,0), y como la columna *dist* muestra, la distancia no es constante entre dos puntos cercanos, pero que se incrementa de 0.145 a 0.165, y regresa otra vez a 145. En términos de animación, esto podría resultar en un movimiento teniendo una aceleración seguida por una disminución de la velocidad, lo cual es a causa de la interpolación arbitraria de una estructura que tiene una geometría esférica de 4-D.

El Algoritmo 6.1 proporciona el código para un movimiento completo de una rotación de matrices a quaternios que usan la rutina mattoguat(mat,q), por medio de interpolación.. La función slerp(p,q,t,qt) regresa el quaternion *qt* interpolado, para *t* entre *p* y *q*. La rutina proporciona para los casos especiales donde los quaternions principales se encuentran muy cerca uno de otro, en dicho caso, se utiliza una interpolación lineal. La función quattomat calcula la conversión de un quaternion a una rotación matricial.

Algoritmo 6.1: quatlib

```
#define X 0
#define Y 1
#define Z 2
#define W 3
#define EPSILON 0.00001
#define HALFPI 1.570796326794895

int nxt[3] = {Y,Z,X};

void quattomat(q,mat)
float q[4];
```

```
	float mat[4][4];
	{
		double s,xs,ys,zs,wx,wz,xx,xy,xz,yy,yz,zz;

		s = 2.0/(q[X]*q[X] + q[Y]*q[Y] + q[Z]*q[Z] + q[W]*q[W]

		xs = q[X]*s; ys = q[Y]*s; zs = q[Z]*s;
		wx = q[W]*xs; wy = q[W]*ys; wz = q[W]*zs;
		xx = q[X]*xs; xy = q[X]*ys; xz = q[X]*zs;
		yy = q[Y]*ys; yz = q[Y]*zs; zz = q[Z]*zs;

		mat[0][0] = 1.0 - (yy + zz);
		mat[0][1] = xy + wz;
		mat[0][2] = xz - wy;

		mat[1][0] = xy - wz;
		mat[1][1] = 1.0 - (xx + zz);
		mat[1][2] = yz + wx;

		mat[2][0] = xz + wy;
		mat[2][1] = yz - wx;
		mat[2][2] = 1.0 - (xx + yy);

		mat[0][3] = 0; mat[1][3] = 0; mat[2][3] = 0; mat[3][3] = 1;
		mat[3][1] = 0; mat[3][1] = 0; mat[3][2] = 0;

}

void mattoquat(mat,q)
float mat[4][4];
float q[4];
{
	double tr,s;
	int i,j,k;

	tr = mat[0][0] + mat[1][1] + mat[2][2];
	if (tr > 0.0) {


		s = sqrt(tr + 1.0);
		q[W] = s*0.5;
		s = 0.5/s;

		q[X] = (mat[1][2] - mat[2][1])*s;
		q[Y] = (mat[2][0] - mat[0][2])*s;
		q[Z] = (mat[0][1] - mat[1][0])*s;
	}
	else {
		i = X;
		if (mat[Y][Y] > mat[X][X]) i = Y;
		if (mat[Z][Z] > mat[i][i]) i = Z;
		j = nxt[i]; k = nxt[j];

		s = sqrt((mat[i][i] - (mat[j][j] + mat[k][k])) + 1.0);

		q[i] = s*0.5;
		s = 0.5/s;
		q[W] = (mat[j][k] - mat[k][j])*s;
		q[j] = (mat[i][j] + mat[j][i])*s;
		q[k] = (mat[i][k] + mat[k][i])*s;
	}
}
```

```
void slerp(p,q,t,qt)
float p[4],q[4];
float t;
float qt[4];
{
      double omega,cosom,sinom,sclp,sclq;
      int i;

      cosom = p[X]*q[X] + p[Y]*q[Y] + p[Z]*q[Z] + p[W]*q[W];

      if ((1.0 + cosom) > EPSILON) {
            if ((1.0 - cosom) > EPSILON) {
                  omega = acos(cosom);
                  sinom = sin(omega);
                  sclp = sin((1.0 - t)*omega)/sinom;
                  sclq = sin((t*omega)/sinom;
            else {
                  sclp = 1.0 - t;
                  sclq = t;
            }
            for (i=0; i<4; i++) qt[i] = sclp*p[i] + sclq*q[i];
      }
      else {
            qt[X] = -p[Y]; qt[Y] = p[X];
            qt[Z] = -p[W]; qt[W] = p[Z];
            sclp = sin((1.0 - t)*HALFPI);
            sclq = sin(t*HALFPI);
            for (i=0; i<3; i++) qt[i] = sclp*p[i] + sclq*qt[i];
      }
}
```

# 6.3 ANIMACIÓN DEL AMBIENTE DE ILUMINACIÓN

Se sabe que la posición de un objeto y orientación pueden ser controlados por operaciones de matrices, y los frames de animación intermedios pueden ser calculados interpolando entre dos frames principales usando técnicas paramétriacas de ángulos de ondulación, inclinación y desviación y quaternions. Ahora aunque las fuentes de luz no necesitan un modelo físico en forma de polígonos, pueden ser interpretados como objetos virtuales y manipulados usando las mismas técnicas. Por ejemplo, la posición de una fuente de luz puede ser calculada de un lugar a otro interpolando los valores de sus coordenadas principales; de igual forma, su intensidad, color, dirección, etc. pueden ser modificados usando las mismas técnicas. Como los parámetros espaciales son cambiados, son reflejados en los niveles de iluminación animada en representación de escenas. Aunque hay que tener el cuidado de no modificar muchos de estos parámetros a la vez, es necesario un tiempo considerable para balancear los niveles de luz y mantener la continuidad de iluminación deseada a través de una secuencia de animación.

Aunque no existe un límite extremo del número de fuentes de luz utilizados en la Animación por Computadora. Además, se debe tomar en cuenta un tiempo extra para representar secuencia tras secuencia, para lograr un cierto efecto visual. De hecho, en escenas complejas, una buena práctica es detener la animación del ambiente de iluminación lo más posible, y ejecutar la línea de prueba usando la configuración de luces que minimice la representación.

En el mundo real, es imposible para dos objetos compartir el mismo espacio; aún en el mundo virtual de las gráficas por computadora algo puede ser posible, aunque puede causar problemas en el software, especialmente cuando algunos

movimientos no han sido anticipados por el animador.

Una escena puede ser iluminada por varias fuentes de luz, y no hay requerimiento para que cada objeto debe recibir la iluminación de cada luz, como esto puede ocasionar problemas de saturación, que es difícil de resolver. Una técnica conveniente que resuleve el problema de balancear los niveles de iluminación, es asociar cada objeto con una o más fuentes de luz específicas. Cuando un foco o fuente de luz direccional es necesario para seguir un objeto cuando se mueve en el espacio, el animador puede asociar un parámetro para controlar la dirección para la fuente de luz con algún vértice del objeto entonces; como el objeto está animado, la fuente de luz automáticamente sigue sus movimientos y garantiza que el objeto permanecerá iluminado.

Otro punto que hay que tomar en cuenta es que la fuente de luz es solamente un punto o dirección en el espacio de donde los rayos de luz se asumen que emanan -no existe un punto de brillo de iluminación. Consecuentemente, cuando la cámara está dirigida hacia la fuente, no habrá registro de algo luminoso. Si dicho brillo es necesario que aparezca, entonces se debe modelar como se desea que ilumine.

Para ilustrar esta idea, se considera el caso en el cual es necesario un resplandor de luz para crear un amanecer. Se puede modelar una esfera brillante. Más que modelar físicamente una esfera brillante, puede ser simulado con escribir en el buffer de intensidad de color para crear un efecto de resplandor. Esto se puede lograr almacenando las intensidades de resplandor como una matriz de valor, que es usada para reescribir el buffer de imágenes en la posición deseada. Las intensidades de resplador pueden ser representadas como una zona central de blanco puro, y en el perímetro se especifica otra intensidad más tenue. Si el buffer de imágenes todavía conserva el color de fondo del cielo, los cambios de intensidad de resplandor no parecerán naturales, y además estos valores iniciales deben resultar parte del resplandor.

La secuencia de animación podría ser la siguiente:

(1) Llenar el buffer de imágenes como algunos grados de color del cielo.
(2) Calcular el pixel central inicial del sol naciente.
(3) Leer el área requerida de pixels del buffer de imágenes.
(4) Modifica dichos pixels con una función que calcula el resplandor.
(5) Escribir de nuevo estos pixels modificados en el buffer de imágenes.
(6) Representar otros elementos de la escena.
(7) Registrar la imágen.
(8) Modificar levemente el color del cielo.
(9) Mover la posición del sol.
(10) Regresar al principio de la secuencia el número necesario de frames.

# CAPITULO 7

# MODELACIÓN Y ANIMACIÓN

## INTRODUCCIÓN

La modelación de objetos y su animación eventual son temas muy relacionados, y en muchos casos es imposible para el proceso de modelación proceder hasta conocer precisamente cómo será animado un objeto. Como una ilustración, se considera el modelar una cabeza humana: ésta puede ser modelada construyendo un modelo físico de yeso, la cual puede ser cortada en secciones horizontales. Esto se puede digitalizar e ingresar en un sistema de animación en donde se pueden conectar triangularmente con una imitación de piel. Este puede ser un método aceptable de modelación, lo cual no será fácil animar cuando el modelo esté hablando.

Si se sabe que la intención es manipular las coordenadas de la superficie para imitar el hablar, se podría, posiblemente, elegir una red de polígonos que corresponderían a los patrones musculares, así que las mejillas, cejas y labios podrían moverse independiente uno de otro.

Así que este es el tema del presente capítulo, además se mencionarán algunos de los problemas que enfrenta el animador en términos de qué proceso de animación puede ser usado para permitir ciertos efectos de animación.

## 7.1 MODELOS FÍSICOS

Con la construcción de un modelo físico de arcilla o yeso, se pueden diseñar objetos, y que con los datos obtenidos de este diseño, se puede tener información que sirva de datos de entrada a la computadora. Actualmente, digitalizadores láser y pruebas electromagnéticos en 3-D proporcionen un método conveniente para obtener o capturar la geometría de la superficie. En el caso del digitalizador en 3-D, un modelo particular es capaz de probar un espacio hemisférico de alrdedor de 2.36 mt. de radio para una resolución de 0.063 mm. El proceso digitzalizador empieza dibujando una red de triángulos o polígonos sobre una superficie, y luego probar los vértices comunes donde las coordenadas $x$, $y$ e $z$ son los datos de entrada para el computador. Una vez, dentro del ambiente de animación. estos datos pueden ser procesados para aclarar las necesidades del animador.

Cuando dicha tecnología no se encuentra disponible, la geometría del modelo puede ser ingresada por métodos fotográficos, o cortando el modelo en partes horizontales. Lo delgado de cada parte dependerá del nivel de detalle requerido a lo largo del corte de ejes, y del número de facetas necesarias en la superficie final. Antes de empezar la partición, debe de existir algún método para alinear las secciones que han sido cortadas; esto se puede lograr colocando dentro del modelo alguna forma de extrusion para actuar como un origen de referencia. Otro método es encerrar el modelo dentro de alguna caja de tamaño conveniente, la cual es llenada con una segunda substancia, tal vez teniendo un color diferente que la del modelo; las esquinas de las partes pueden resultar como marcas de referencia.

Cuando las secciones han sido cortadas, sus formas pueden ser trazadas en papel, junto con las posiciones de las marcas de referencia. Como dichas secciones eventualmente tendrán que ser conectadas como una superficie de red, es muy importante la posición de los puntos que van a ser digitalizados sobre cada sección. Un método para fijar estos puntos es elegirlos como posiciones angulares específicas alrededor del contorno, tal vez cada 5° a 10°; esto asegura que las secciones vecinas tendrán el mismo número de puntos, y les permita conectarse fácilmente. La Figura 7.1 muestra como una red triangular puede ser

creada entre cualquiera dos seciones uniendo los puntos como se ilustra. Obviamente, esta técnica de secciones no puede arreglárselas con objetos que tienen cinco características cóncavas como las que se encuentran en la oreja.

Dentro de la computadora, las secciones digitalizadas sólo tendrán dos valores de coordenadas, las coordenadas *x* e *y*. Para reconstruir el modelo con las secciones, se deberán trasladar a su posición *y* correcta y entonces enlazándolos para formar una piel en 3-D.

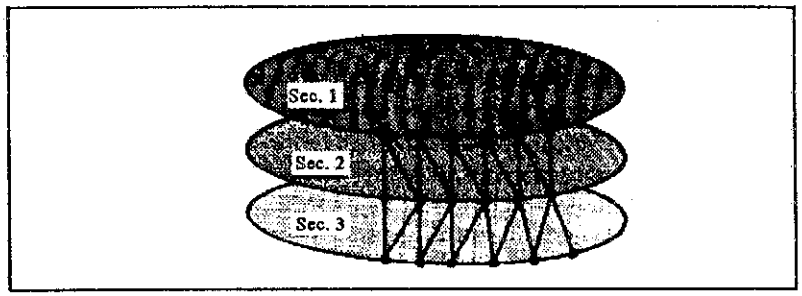



**Figura 7.1**: si estos tres contornos son secciones de una parte tridimensional, a través de un objeto, entonces ellos pueden ser usados para construir una red triangular 3-D uniendo los vértices como se ilustra.

# 7.2 FOTOGRAFÍAS ORTOGONALES

Esta técnica puede ser usada con una cabeza real o con un modelo, y empieza con trazar una red de triángulos sobre la superficie, anticipando los requerimientos de animación, y alterando el tamño de los triángulos para igualar la curvatura de la superficie.

La segunda etapa es fotografiar el objeto de frente y de lado, idealmente con un gran lente focal para minimizar el grado de perspectiva en la imagen. Sólo se necesita uno de los laídos y se asume que la cabeza es simétrica, y la otra mitad puede ser construida para formar una reflexión de uno de los lados modelados. Los fotografos pueden entonces ampliar a un formato conveniente, tal vez dos veces el tamaño orginal, y los triángulos trazados en el papel, donde son anotados con un conjunto de números únicos asociados con los vértices en cada una de las dos vistas.

La tercera etapa requiere ser montada sobre un digitizer y alineado para representar una elevación frontal y lateral del objeto. El software que construye la red de triángulos permitirá al operador probar el mismo vértice en ambas vistas, lo cual proporciona los tres valores de las coordenadas, y cruzando las vistas en secuencia triangular, la superficie 3-D puede ser capturada.

Una cuarta etapa desarrolla la otra mitad de la cabeza reflejando los triángulos originales alrededor del plano vertical central de la cabeza, pero este proceso de reflexión revertirá la secuencia de los vértices de estos triángulos y, esto será importante para la representación, estos triángulos nuevos tendrán que ser manipulados para corregir este efecto.

La Figura 7.2 muestra una vista frontal y lateral de la cabeza, la que ha sido usada por Keith Waters en su investigación sobre expresión facial. Como cada vértice existe en ambas vistas, sus coordenadas 3-D se pueden obtener fácilmente.

# 7.3 LASER SCANNING

Actualmente hay una gran variedad de laser-scanning, a través de los cuales ciertos modelos pueden ser colocados sobre una plataforma giratoria y, mientras rota, el objeto se digitaliza en líneas verticales por medio de laser-scanning y el punto de contacto óptico es medido por una cámara para revelar su posición 3-D en el espacio. Esto es posible si la posición de la fuente del láser es

conocida en relación al detector y esta geometría permite al punto iluminado a ser transformado en un objeto en el espacio. Estos puntos pueden ser ingresados en una computadora donde pueden ser usados para construir una réplica del modelo original. Dichos dispositivos son capaces de generar grandes archivos de coordenadas.

# 7.4 ESTRUCTURAS ENLAZADAS

Una estructura enlazada consiste de dos o más elementos que están conectados de alguna forma, normalmente a través de un mecanismo de unión que permite varios grados de libertad. Un simple ejemplo es utilizado en una caja donde la tapa se le permite rotar alrededor de un eje alineado con uno de los bordes de la caja. Un ejemplo más complejo es encontrado en los mecanismos asociados con pistones en una máquina de combustión. El cuerpo humano es también una estructura enlazada; su animación requiere de técnicas especiales para hacer que su movimiento sea real y expontáneo.

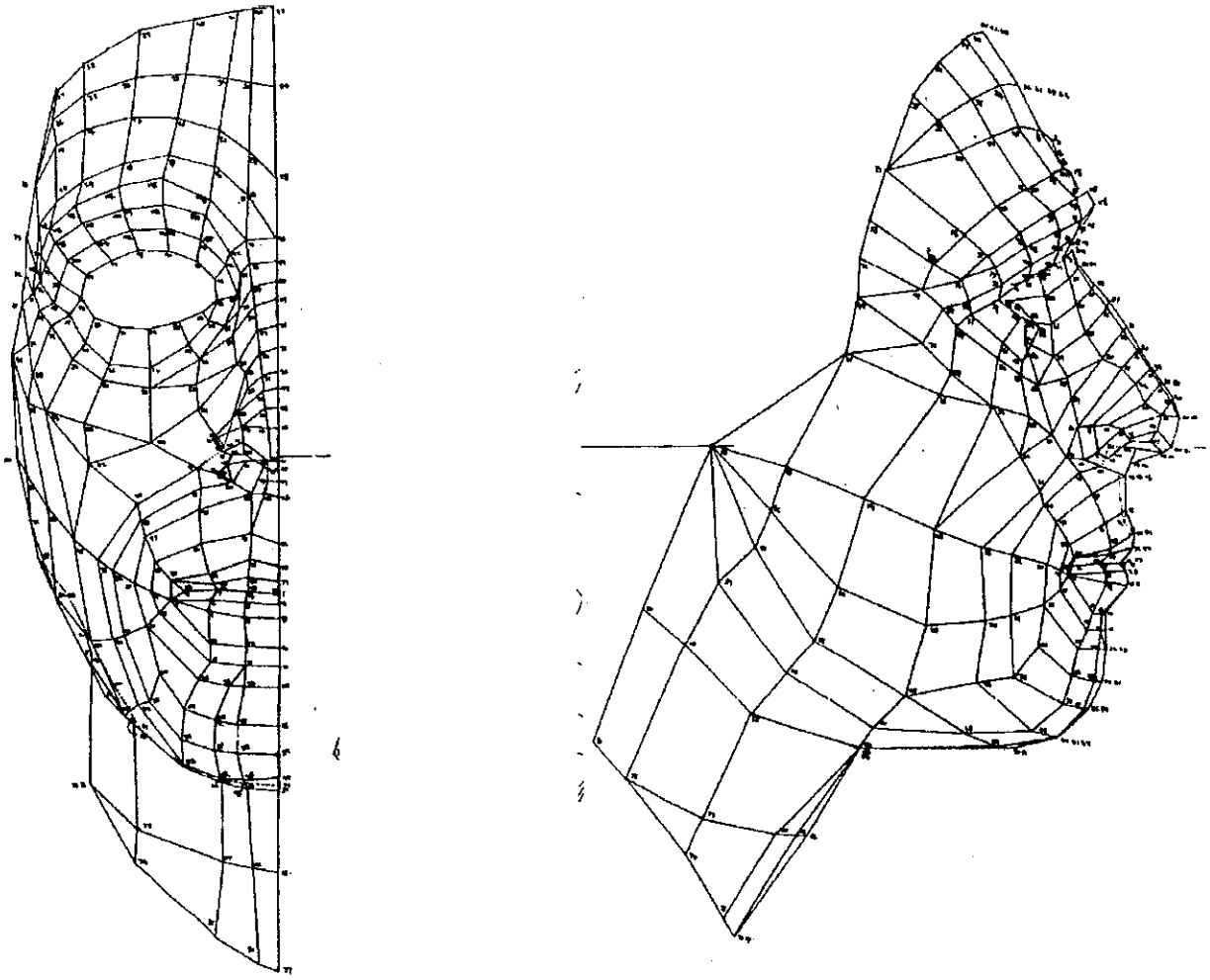


**Figura 7.2**: estos dos dibujos fueron preparados por Keith Waters usando dos fotografías de un objeto cubierto con polígonos. Luego fueron digitados para formar una red polinomial en 3-D y luego animarla.

Una característica importante de las estructuras enlazadas es la jerarquía que los une, más las limitaciones asociadas con sus grados de libertad, y de alguna forma esta conectividad debe ser construida dentro del sistema. En el caso de una caja con una tapa con bisagra, no importa cómo la caja es animada: la tapa debe ser una parte coherente de cualquier movimiento. Todavía la tapa posee su propio grado de libertad en que puede ser pivote alrededor de un eje asociado con el frame de referencia de la caja. En caso del frame humano, los brazos, piernas, hombros, manos, dedos y espina dorsal todos pueden moverse independientemente uno de otro, dentro de ciertos límites; no obstante, todos se encuentran juntos cuando el cuerpo está sometido a alguna acción global como caminar, correr o caer.

Sin la ayuda de algún sistema de animación, esta jerarquía de uniones debe encontrar su forma dentro del ambiente de animación y en las estructuras de datos que permiten la geometría del modelo, o como parte de los procedimientos de animación. Si están implementados dentro de una estructura de datos, entonces el animador debe permitir la emisión de comandos que manipulen cualquier parte de la jerarquía, sabiendo que el sistema aplicará automáticamente el mismo proceso para elmentos inferiores en la estructura. Esto se puede notar en el caso de la animación de una mano alrededor de su muñeca: cuando la mano es rotada, los dedos -que están compuestos de elementos pequeños- todos deberán ser rotados la misma cantidad. No obstante, cuando el primer dedo es animado con un movimiento de un golpe ligero, el sistema debe asegurarse que solo este dedo se moverá. Además, debe ser posible establecer el movimiento del dedo mientras la mano está rotando alrededor de la muñeca.

Este tipo de conectividad puede ser lograda por procedimientos que están diseñados desde un principio con esta jerarquía, y que son fácilmente implementados dentro de un lenguaje de programación orientado a objetos. Una estructura como HAND puede ser diseñada para referenciar seis elementos: una palma, el pulgar y cuatro dedos restantes. HAND conoce su eje de rotación, y mantiene el punto pivote para sus propios elementos. Cuando el animador emite un comando para rotar la mano; HAND es activado en un modo de rotación, lo que garantiza por consiguiente que la palma, el pulgar y el resto de los dedos son rotados. Esta rotación puede ser reflejada directamente como un valor nuevo de una coordenada para estos elementos, o trasladar a nuevas matrices que serán usadas para rotar los elementos cuando sean eventualmente desplegados.

Para ilustrar esto, se considera un pequeño problema de un mecanismo unido formado por un disco de rotación y el movimiento de una barra conectada al disco. Se requiere una secuencia de animación que rotará el disco a cualquier velocidad, y mostrar la posición correcta de la barra unida que está restringida en un extremo para una acción de deslizamiento a lo largo del eje-x. El disco y la barra están modeladas tal que sus posiciones iniciales se muestran en la Figura 7.3. Además, este ejemplo es para dos dimensiones; la misma técnica puede ser usada fácilmente en el caso de tres dimensiones.

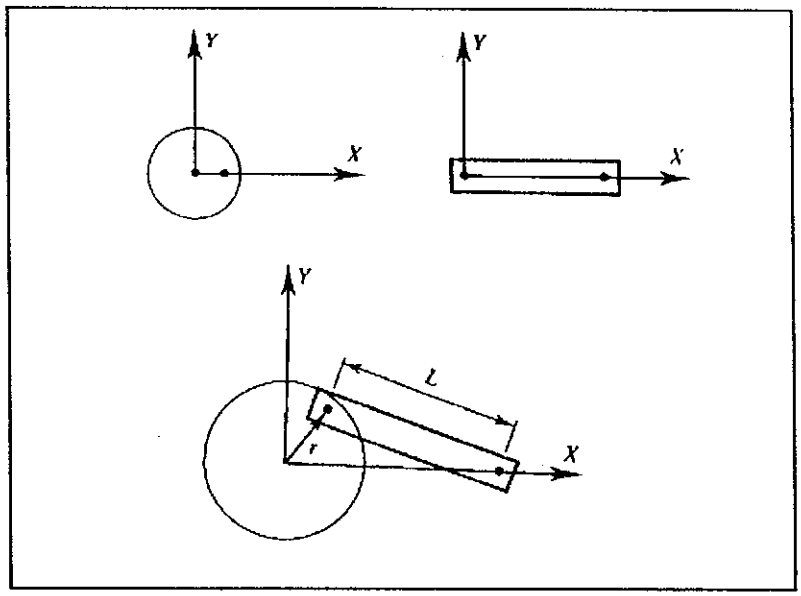



**Figura 7.3:** esta estructura enlazada muestra una barra conectada a un disco en rotación, donde el objetivo de la secuencia de animación es mostrar la barra en la posición correcta cuando el disco rota. Los dos diagramas muestran las posiciones iniciales de los dos elementos.

La animación puede ser lograda en forma procedural lo cual, dada una rotación angular del disco ß, el disco rotará en la posición correcta, y orientar la barra correctamente. Para resolver el problema, se necesita entender la geometría en términos del disco de rotación ß, el radio *r* de la conexión con el disco, y la longitud *L* de los puntos fijos de la barra. La Figura 7.4 muestra que para ciertos valores de *r*, *ß* y *L* será posible calcular el ángulo ß, el cual permite a la barra posicionarse automáticamente por un procedimiento de animación.

Las coordenadas *x* e *y* del punto *P* son ((rcos(ß),rsen(ß)), y:

$$\text{sen}(\alpha) = r\text{sen}(ß)/L$$

y:

$$\cos(\alpha) = \sqrt{(1 - r_2\text{sen}_2(ß)/L_2)}$$

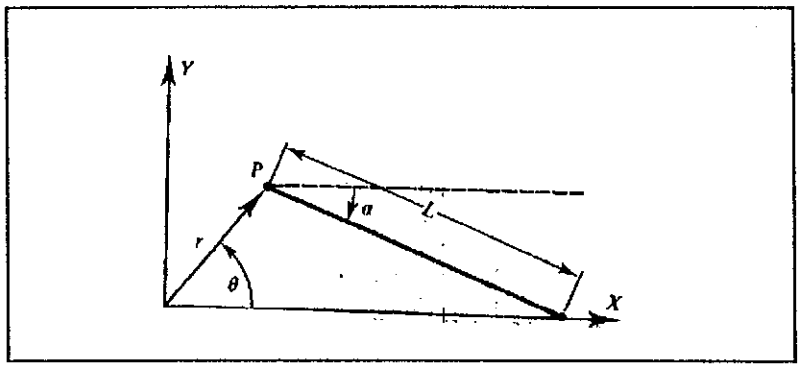



**Figura 7.4:** este diagrama ilustra las restricciones impuestas sobre el disco y la barra, donde, para cualquier *r*, *ß* y *L* es posible calcular ß, el cual permite a la barra posicionarse automáticamente por un procedimiento de animación.

Si la posición inicial de la barra es siempre horizontal con el eje-x, esto puede ser copiado y rotado -α alrededor del origen para proporcionar el ángulo correcto, y entonces trasladar (rcos(ß),rsen(ß)) para conectarlo a la barra. La primera operación se obtiene con la matriz siguiente:

$$\begin{bmatrix} \cos(-\alpha) & -\text{sen}(-\alpha) & 0 \\ \text{sen}(-\alpha) & \cos(-\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

simplificada:

$$\begin{bmatrix} \cos(\alpha) & \text{sen}(\alpha) & 0 \\ -\text{sen}(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

y la traslación es lograda por:

$$\begin{bmatrix} 1 & 0 & r\cos(\beta) \\ 0 & 1 & r\text{sen}(\beta) \\ 0 & 0 & 1 \end{bmatrix}$$

concatenando las dos matrices anteriores produce:

$$\begin{bmatrix} \cos(\alpha) & \text{sen}(\alpha) & r\cos(\beta) \\ -\text{sen}(\alpha) & \cos(\alpha) & r\text{sen}(\beta) \\ 0 & 0 & 1 \end{bmatrix}$$

Esta matriz final puede ser usada para rotar y trasladar una copia de la barra unida por cualquier rotación angular ß del disco. Si se asume que $r = 1$, $L = 2$, y las coordenadas de los extremos izquierdo y derecho de la barra son $(x_L, y_L)$ y $(x_R, y_R)$ respctivamente, entonces la tabla 7.1 resume algunos de los valores de las coordenadas. Estos valores de $(x_L, y_L)$ y $(x_R, y_R)$ confirman que la barra está rotada y trasladada cuando el disco rota.

**Tabla 7.1**

| ß | α | $x_L$ | $y_L$ | $x_R$ | $y_R$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 3 | 0 |
| 90 | 30 | 0 | 1 | 1.732 | 0 |
| 180 | 0 | -1 | 0 | 1 | 0 |
| 270 | -30 | 0 | -1 | 1.732 | 0 |
| 360 | 0 | 1 | 0 | 3 | 0 |

# 7.5 ESTRUCTURAS

Uno de los desarrollos más importantes en la Animación por Computadora ha sido la habilidad de modelar y animar objetos flexibles como banderas, alfombras y manteles. El modelar se logra a través del uso de redes de triángulos.

En el caso de una bandera, puede ser animada al alterar los vértices de una red de triángulos usando las ondas senoidales, la Figura 7.5 muestra cómo una bandera se fija al eje-y con sus vértices compensando las direcciones de $z$ positiva y negativa. Esto se obtiene con la función:

$$z\_offset = amp\ \text{sen}(\pi x/w)$$

donde:

*amp* es la amplitud de compensación
*x* es la posición a lo largo del eje-x
*w* es la longitud de onda de las ondulaciones

Esta función coloca la bandera dentro de una posición ondulada, pero se puede animar agregando un parámetro de tiempo para forzar un ciclo en distintos frames. Esto se obtiene con la función:

$$z\_offset = amp\ sen(2\pi(x/w + n/f))$$

donde:

*n* es el número de frame
*f* es el número de frames en un ciclo completo

Esto ocasiona un problema relacionado con el fijar la bandera al eje-y. Una posición fija evitará que la bandera se mueva en este punto, y la función z_offset hará que la red triangular se mueva. Esto se prodría resolver modificando la función para aumentar en un valor dependiendo sobre la coordenada-x, tal que:

$$z\_offset = x\ amp\ sen(2\pi(x/w + n/f))/con,$$

donde *con* es alguna constante conveniente, o introduciendo algún procedimiento donde esta condición sea detectada, y la función z_offset controlada.

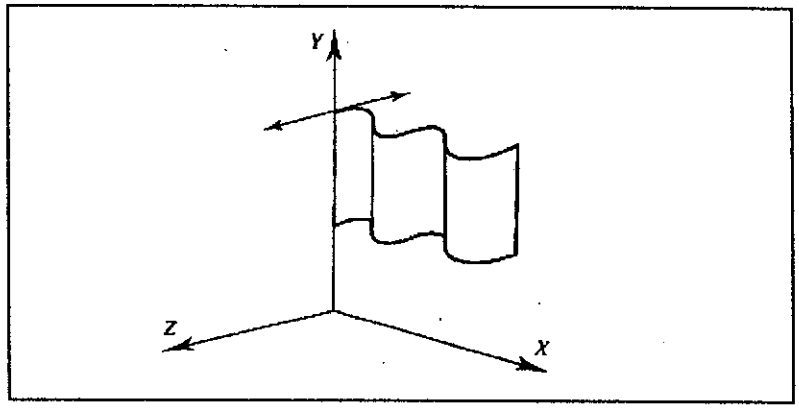



**Figura 7.5**: esta bandera ha sido modelada por alguna función senoidal.

A pesar de este análisis, la animación no parecerá real -parecerá que una hoja plástica ha sido distorcionada usando ondas senoidales. ¿Cómo se puede realizar el movimiento físico real? Para empezar, si se distorsiona la red de triángulos con algunas funciones que simulan la acción de ondas con diferentes amplitudes, frecuencias y fases, el movimiento será más real. Entonces, el observador notará que la bandera está siempre colgada en el espacio como si tuviera algún soporte invisible, pero se sabe que una bandera cae por la fuerza de gravedad cuando no ha sido movida por el viento.

Una técnica para modelar estructuras es representarla como una matriz de puntos de masa conectados en una red triangular, donde cada masa está conectada a su vecino usando resortes como se muestra en la Figura 7.6. Las fuerzas que actúan a través de los resortes pueden influenciar en la masa usando la segunda ley de Newton, la cual establece que el cambio de velocidad de momento es directamente proporcional a la fuerza aplicada. Por ejemplo, se toma un simple hilo fijo de 2-D en su extremo. Esto se puede representar por una secuencia de puntos de masa *m* conectados a sus vecinos por medio de resortes. Si los resortes tienen una longitud *L* en condición relajada, entonces la ley de Hooke establece que su extensión es proporcional a la fuerza aplicada. Para calcular esta fuerza, es necesario calcular primero la extensión del resorte.

La Figura 7.7 muestra las posiciones de los tres puntos de masa donde las posiciones están especificadas por:

$$(x_{i-1},y_{i-1}),(x_i,y_i) \text{ y } (x_{i+1},y_{i+1}),$$

entonces las longitudes izquierda y derecha $L_L$ y $L_R$ serán:

$$L_L = \text{SQRT}((x_{i-1} - x_i)^2 + (y_{i-1} - y_i)^2)$$
$$L_R = \text{SQRT}((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)$$

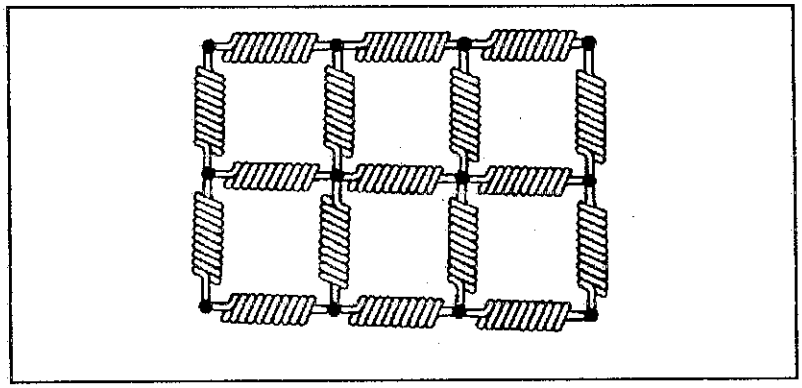


**Figura 7.6:** la estructura puede ser representada como una red de puntos de masa conectados uno con otro con resortes rígidos. Las fuerzas que actúan a través de estas fuerzas pueden ser calculadas y usadas para mover las posiciones de los puntos de masa para simular el comportamiento del material bajo fuerzas de gravedad.

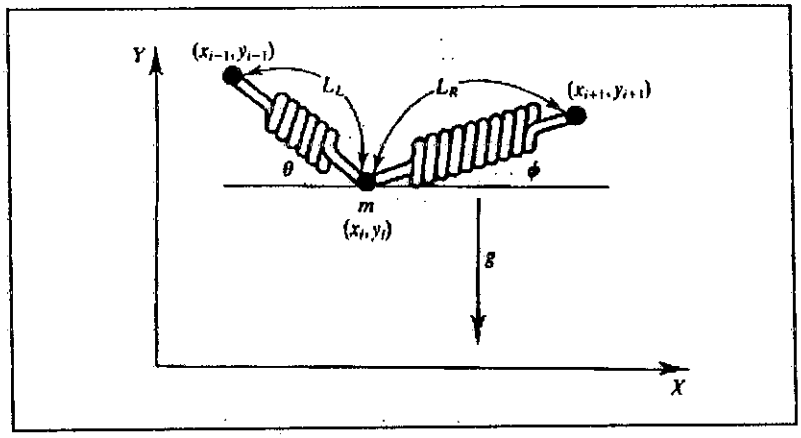



**Figura 7.7:** dados tres puntos de masa cayendo por la acción de la gravedad (g), sus posiciones de cambio en el espacio pueden ser determinados resolviendo los componentes de las fuerzas horizontal y vertical.

y las extensiones derecha e izquierda del resorte $E_L$ y $E_R$ serán:

$$E_L = L_L - L \quad \text{y} \quad E_R = L_R - L$$

Con las extensiones del resorte calculadas, se necesita identificar los componentes vertical y horizontal de las fuerzas, para ello se necesitan saber los ángulos del resorte. Estos son:

$$\cos(\beta) = (x_i - x_{i-1})/L_L \qquad \text{sen}(\beta) = (y_i - y_{i-1})/L_L$$

y

$$\cos(\phi) = (x_{i+1} - x_i)/L_R \qquad \text{sen}(\phi) = (y_{i+1} - y_i)/L_R$$

Ahora, como la fuerza empleada por un resorte es proporcional a su extensión y rigidez *S*, entonces las fuerzas horizontales actúan sobre la masa i-ésima usando la segunda ley de Newton donde la fuerza es igual a la masa por

la aceleración (F = ma), son:

$$ma_h = S(E_R\cos(\phi) - E_L\cos(\beta))$$

además:

$$a_h = S/m((L_R - L)(x_{i+1} - x_i)/L_R - (L_L - L)(x_i - x_{i-1})/L_L) \qquad (1)$$

y se puede obtener una expresión similar para la aceleración vertical:

$$a_v = S/m((L_R - L)(y_{i+1} - y_i)/L_R - (L_L - L)(y_i - y_{i-1})/L_L) - g/m \qquad (2)$$

Las dos ecuaciones anteriores describen las aceleraciones vertical y horizontal del punto de masa central, luego se determinan los valores de las coordenadas *x* e *y*. Un método que se presta a la animación es el método de Euler para integración numérica, que permite evaluar el desplazamiento del punto de masa asumiendo que se mueven en cantidades pequeñas en intervalos de tiempo cortos, sería mejor que estos intervalos de tiempo sean infinitesimalmente pequeños.

La aceleración es una medida del coeficiente del cambio de velocidad. Sobre una base incremental, la aceleración horizontal puede ser definida en términos de la velocidad horizontal; se tiene que:

$$a_h = \frac{v_h(t + Deltat) - v_h(t)}{Deltat},$$

donde *t* es cualquier punto en el tiempo, y Deltat es un incremento pequeño. La velocidad en el tiempo *t* + Deltat es:

$$v_h(t + Deltat) = v_h(t) + Deltata_h \qquad (3)$$

pero la velocidad también se puede definir como:

$$v_h(t) = \frac{x(t + Deltat) - x(t)}{Deltat}$$

la posición horizontal del punto de masa está dado por:

$$x(t + Deltat) = x(t) + Deltatv_h(t) \qquad (4)$$

En $t_0$ cuando $t = 0$, la velocidad horizontal y aceleración son cero:

$$v_h(t_0) = 0 \quad y \quad a_h(t_0) = 0$$

Por lo tanto, al evaluar la ecuación original (1) que relaciona las fuerzas horizontales con la aceleración para obtener la aceleración inicial. Esto es substituio en la ecuación (3) con un intervalo de tiempo pequeño Deltat y una velocidad inicial cero, para calcular la nueva velocidad, que luego es substituida en la ecuación (4) para determinar la nueva posición del punto medio de la masa. El mismo proceso se aplica a la ecuación (2) para obtener el desplazamiento vertical. Cuando se obtienen los dos desplazamientos *x* e *y*, la posición media del punto de la masa es actualizado, y el ciclo se vuelve a repetir. Los valores *S*, la fuerza de gravedad *g* y la masa *m* tendrán que ser asignados por el animador para crear el amibiente deseado.

# 7.6 LA FIGURA HUMANA

Se sabe que la figura humana es un objeto muy complejo que está compuesto de un esqueleto que soporta la carne, grasa y tejido muscular, todo cubierto de

piel, que se mueve de diferentes formas y que trata de mantener un estado de balance dentro de un campo gravitacional que continuamente atrae cada elemento al suelo. El torso, cabeza, brazos y piernas son grandes masas controladas por músculos que permiten una estructura articulada para moverse de una posición a otra, ya sea una actividad como caminar, bailar, correr o sentarse. Modelar y animar una estructura, parece un trabajo difícil, ya que también se deben tomar en cuentra los problemas de la expresión facial de hablar y respirar.

## 7.6.1 LA CABEZA HUMANA

Una forma de modelar una cabeza humana es utilizar polígonos que son obtenidos digitando un modelo físico. El modelo puede ser animado usando animación por desplazamiento, deformación de formas o simulando el control muscular.

El modelar un rostro humano siempre ha sido una gran tarea para animadores por computadora y el trabajo realizado por Fred Parke 'A parametric model for human faces' fue un éxito en esta área. Este trabajo se centra en el desarrollo de un rostro construído con base en una red poligonal, la cual fue manejada por medio de una interpolación de características faciales usando traslación, rotación y la técnica escalar.

Otro trabajo de investigación elaborado por Platt y Badler para la simulación de los movimientos del rostro humano, se basó en procedimientos para mover vértices en una forma pre-definida, Platt y Badler implementaron un modelo que simulaba la piel como limitaciones de arcos interconectados. Un trabajo más reciente sobre la animación facial fue desarrollado por Pat Hanrahan quien usó una representación caligráfica para evaluar los parámetros que controlan los atributos faciales. Más recientemente, Keith Waters creó un sistema de músculos formados por vértices específicos de una red polinomial para moldearlo con diferentes expresiones. La Figura 7.8 muestra una parte de una red polinomial en 3-D que es distorcionada moviendo los vértices *P* a un punto fijo *A* hacia arriba. Si el movimiento a lo largo de la dirección del músculo ilustrado es de una unidad, entonces la nueva posición *P* se puede calcular fácilmente, aunque el movimiento de un simple punto no tendrá un impacto real sobre toda la expresión. No obstante, si los vértices circunvecinos son movidos también al punto *A*, el efecto será más obvio; entonces se necesitará definir un radio de influencia para restringir la distorsión para una región del rostro.

Esto se puede lograr usando la función coseno para atenuar la cantidad a mover sobre el radio de influencia, y asegura que el vértice fijado al músculo está sujeto a un máximo movimiento. Las láminas 3 y 4 ilustran tres vistas de una cabeza, (ver apéndice).

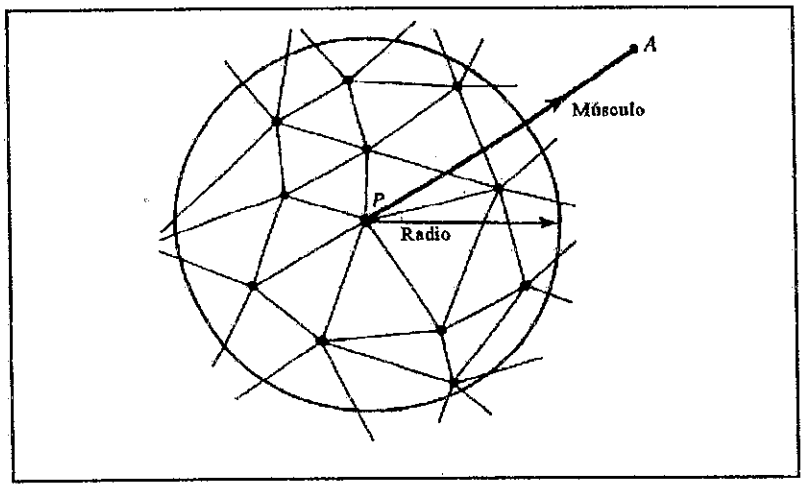



**Figura 7.8**: Keith Water utilizó una red polinomial para lograr una expresión facial la cual puede ser distorcionada moviendo los vértices dentro de un radio dado hacia un punto *A*. El vértice *P* está expuesto a un gran movimento, mientras otros movimientos son disminuidos por una función coseno.

# 7.6.2 EL CUERPO HUMANO

Cuando se está modelando una forma humana, se sabe exactamente qué tipo de animación se va a realizar, al menos que sea construido con las uniones correctas, y proporcionarle una piel que se adapte a los movimientos. El cuerpo humano ha sido modelado y animado usando una variedad de técnicas, desde un simple hombre hecho de 'palillos' hasta estructuras articuladas animadas por procedimietnos que introducen fuerzas, velocidades y aceleraciones, para crear movimientos naturales. No obstante, con una animación tradicional, la locomoción humana es muy difícil de imitar, quizá porque el cuerpo humano y su movimiento son muy familiares.

Los animadores tradicionales frecuentemente utilizan un rotoscopio como un método de hacer sus propias imágenes naturales: esto involucra el trazado de dibujos de una película o un video para capturar ciclos del caminado humano o de movimientos complejos, por ejemplo, en deportes, gimnasia o baile. No obstante, estas imágenes son en 2-D, mientras que el ambiente de Animación por Computadora es 3-D, y aunque el rotoscopio convencional puede ser usado para animar imágenes de dibujos animados asociados con sistemas de pintura, el rotoscopio 3-D es necesario para manejar modelos 3-D. Esto involucra obtener coordenadas 3-D de las uniones más importantes mientras el sujeto está en movimiento. Existen dos métodos para obtener las coordenadas: fotografía y rastreo 3-D. La primera técnica involucra marcas fijas de las uniones y registro del movimento con dos cámaras localizadas en posiciones conocidas. La geometría del ambiente registrado permite obtener las coordenadas 3-D de la imagen, aunque si bien manualmente, éstos pueden ser usados para fijar las posiciones de las uniones en cada frame.

Un desarrollo reciente en la Animación por Computadora es la realidad virtual: esto involucra técnicas de interface que permiten al usuario interactuar físicamente con los mundos virtuales 3-D creados por las gráficas en computadora. El usuario utiliza guantes interactivos y trajes con transductores para rastrear su posición y estado físico, y los datos obtenidos por estos dispositivos son usados para controlar directamente la posición y orientación en la computadora. Algunos localizadores 3-D y guantes de bajo costo se encuentran disponibles y su influencia aumentará la forma en que se interactúa con el ambiente de Animación por Computadora.

El modelo de un hombre hecho con palillos puede ser animado colocándolo en diferentes posiciones y ángulos o cualquier proceso que se utilice el modelo es animado, eventualmente tendrá un recubrimiento que de alguna forma lo hará un cuerpo sólido. Una forma estilizada es reemplazando los palillos por una secuencia de anillos representando secciones de las diferentes partes del cuerpo; no obstante, estas secciones harán que el muñeco parezca transparente.

La Figura 7.9 muestra dos elementos AB y BS que representan un brazo del modelo de palillos, con una unión en el codo B. Los elmentos de anillos pueden ser usados para obtener una red poligonal, la cual produce un efecto de piel. En esta ilustración, un polígono sombreado ha sido creado de dos secciones contiguas formando los cuatro vértices 1, 2, 3 y 4. Cuando los elementos del antebrazo son animados, las secciones asociadas con él también deben ser movidas, y como la capa de piel creada también debe ser animada. No obstante, la unión del codo presenta un problema, porque la sección asociada con la unión pertenece al brazo y antebrazo.

Una simple solución es orientar la sección, tal que hay que bisecar el ángulo 3-D en esta unión. De esta forma, como el ángulo de unión es interpolado, la capa de piel poligonal es desarrolada para cada frame de animación.

Vestir esta capa de piel con camisas, blusas, faldas o pantalones es otro problema; si el modelo está siempre vestido, entonces las secciones representarán a la ropa más que al cuerpo. Tal vez la parte del cuello y las muñecas se puedan modelar separadamente, pero su movimiento relativo en el cuello y las mangas permanecerá fijo, salvo que alguna técnica extra sea desarrollada para crear este movimiento.

Una forma diferente usada para modelar y animar esqueletos es construir un sistema de contornos en 3-D que representen los elementos para los huesos, los

cuales son usados como una base para localizar esferas o elipsoides de diferentes radios. Por cada elemento del hueso, se almacena la posición de la esfera/elipsoide, juntos con su radio. En términos de polígonos, el modelo necesita poco almacenamiento.

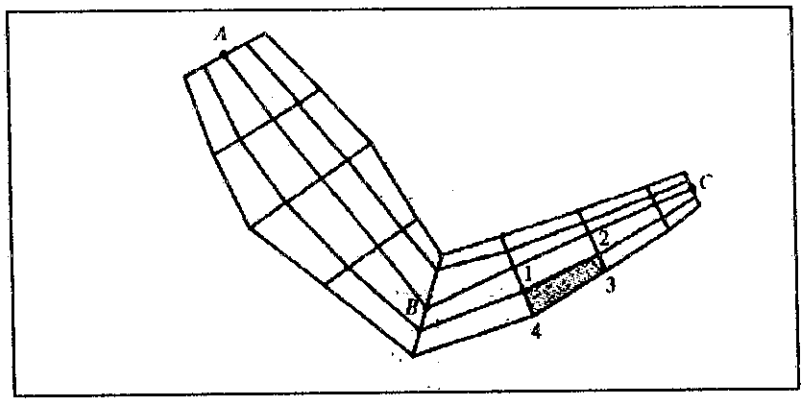



**Figura 7.9**: este punto de vista de la unión del codo muestra los dos elementos AB y BC unidos por el codo B. Pueden ser usadas secciones ortogonales para desarrollar la capa de piel que se ajusta dinámicamente a los movimientos de las uniones.

Cuando una animación compleja no es importante, el modelo puede ser construido a base de una red poligonal desarrollada con la asistencia de un sistema de modelo interactivo; algunos movimientos sutiles como una sonrisa pueden ser creados intercambiando las posiciones de los vértices. Otros movimientos, como la respiración, pueden ser logrados con una deformación o desplazamiento.

# 7.7 FENÓMENO NATURAL

Hubo una época en que el uso de los polígonos era la única forma para construir objetos en 3-D, y la complejidad de dichos modelos era mayor por la restricción de memoria y velocidad en las computadoras. Por ello, no se consideraba el modelar cosas como nubes, fuego, árboles, lluvia o espuma. No obstante, con el software moderno y las computadoras con multi-procesadores super rápidos con gran capacidad de memoria, dichas tareas se realizan con gran éxito.

Un fenómeno natural en el modelar algún objeto, como uno se lo imagina, está relacionado con los objetos y efectos que ocurren naturalmente en el mundo, y ha requerido de nuevas técnicas de modelación como sistemas de partículas, que junto con los procedimientos matemáticos para simular efectos de absorción atmosférica, arcoiris, espejismos, amaneceres y patrones de luz cáustica. Para describir cómo las olas del mar rompen en la playa en forma real, se necesita de una teoría respecto al tema y de los mecanismos de cómo las olas del mar interactúan con la inclinación de la playa.

Igualmente, el modelar y animar plantas requiere un estudio de los mecanismos de crecimiento de las mismas: cómo son inflenciadas por la luz y gravedad, la importancia de su forma y proporción, y cómo son necesarios diferentes niveles de detalle para representar un elemento simple, un grupo de elementos o un bosque.

## 7.7.1 SISTEMA DE PARTÍCULAS

El Particle Systems fue usado primeramente por Lucasfilm Ltd en 1982 para la película 'Star Trek II: the Wrath of Khan'. En la secuencia Génesis, se usaron partículas para modelar una pared en llamas que rodea un planeta, y que también han sido usados para modelar efectos de pasto, árboles, rociadores y fuegos

artificiales. Como sugiere el nombre, es una colección de partículas usadas para simular los efectos visuales creados por diversos fenómenos naturales. En el caso de un árbol distante, no es necesario modelar cada rama, y hoja: como el ojo es incapaz de distinguir hasta este nivel de detalle, todo lo que se tiene que crear es un efecto visual de colores que se interpondrán uno sobre otro y de reflejar la luz de acuerdo con las leyes estocásticas.

Una partícula individual puede ser asignada a diferentes atributos, tal que una posición en 3-D, color, tamaño, velocidad, aceleración, edad y tiempo de vida, puedan ser controlados y modificados por medio de procedimientos programados. En el caso de un árbol, las reglas están definidas para controlar su altura y su ancho de base. William Reeves usó las siguientes reglas para crear una forma de árbol para la secuencia de animación 'The Adventures of Andre and Wally B':

```
altura = mean_altura + ran()delta_altura
ancho  = altura(mean_ancho + rand()delta_ancho)
```

donde mean_height y mean_width son valores típicos de altura y anchura asociados con un tipo de árbol, y delta_height y delta_width representan la desviación máxima de los valores medios. La función rand() regresa un número aleatorio uniformemente distribuido entre -1.0 y 1.0.

El tronco del árbol y las ramas asociadas a él fueron modeladas usando formas de polígonos convencionales, la decoración de las hojas fue producido aplicando las reglas anteriores para generar alrededor de un millón de partículas aleatorias alrededor del tronco eje central. Aun el color de cada partícula fue seleccionado sobre la base de otras reglas.

Como un ejemplo que utiliza el particle system, es el de representar una fuente de agua en 3-D, que puede ser modelada considerando un chorro en 2-D de partículas de agua que pueden ser movidas por ángulo aleatorio alrededor de un eje vertical para crear un volumen de agua rociada. Para empezar, se necesita una fuente colocada en el suelo como se muestra en la Figura 7.10. Las partículas son lanzadas desde este punto hacia algún ángulo aleatorio para crear el efecto natural del chorro. Como se puede asumir que las partículas tienen diferentes masas, sus velocidades de salida variarán con una distribución similar. Y durante su trayectoria hacia arriba su velocidad inicial se reducirá debido a la fuerza de atracción de la gravedad, lo cual implica que en algún punto la fuerza gravitacional llevará a la partícula lo más alto posible y luego regresará hacia abajo con una velocidad mayor. Cuando la partícula choca contra la superficie de la fuente, crea una salpicadura donde una o más partículas viajan hacia arriba para crear posteriormente salpicaduras similares.

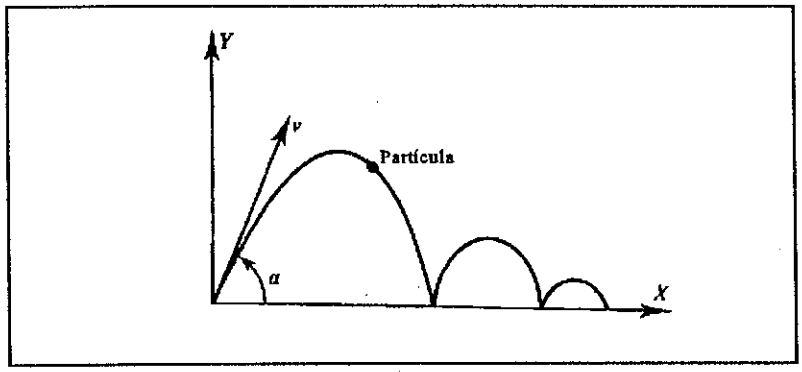



**Figura 7.10**: si una partícula que tiene masa y una velocidad inicial es proyectada lejos del suelo, describirá una trayectoria parabólica influenciada por la gravedad. Un procedimiento de animación puede simular este efecto y puede lograr que cuando la partícula regresa al suelo, más partículas pueden ser generadas que describirán trayectorias similares.

Para controlar las partículas, es necesaria una tabla de entradas en el software para registrar los distintos atributos, y se asume que esta tabla es capaz de registrar el estado de 1000 partículas. Cada partícula es una entrada en la tabla; para evitar que las partículas dejen el chorro a la vez, sus tiempos

de salida serán extendidos sobre un período de tiempo finito.

La animación continúa examinando la tabla para una partícula, cuyo tiempo de permanecer en el chorro terminó. Luego se le asigna una velocidad y ángulo aleatorios basados en las funciones siguientes:

$$v = \text{mean\_velocidad} + \text{rand}()\text{delta\_velocidad}$$
$$\alpha = \text{mean\_ángulo} + \text{rand}()\text{delta\_ángulo},$$

donde $v$ es la velocidad de la partícula y $\alpha$ es el ángulo que se forma con el suelo. El ángulo y velocidad de la partícula es el resultado de la energía ejercida por la partícula desde una presión imaginaria de agua a la fuente, pero cuando dicha partícula es creada, la gravedad la impulsa con una fuerza opuesta.

La velocidad puede ser dividida en dos componentes. La vertical y la horizontal. El componente vertical, que está sujeto a la influencia de la gravedad, es $vsen(\alpha)$, el componente horizontal el que permanece constante bajo la fricción del viento, y está representada por $vcos(\alpha)$. Consecuentemente, la altura de la partícula en el tiempo $t$ se calcula por:

$$y = vsen(\alpha)t - 1/2gt^2,$$

donde el término negativo representa la altura de caída de la partícula bajo la influencia de la gravedad. Similarmente, la distancia horizontal recorrida por la partícula es calculada por:

$$x = vcos(\alpha)t$$

Ignorando, por el momento, las salpicaduras generadas por las partículas que están cayendo, la animación procede a almacenar en la tabla de partículas el tiempo en que cada partícula empieza su formación y calcula la posición del punto en donde se encontrará en el tiempo. La tabla también mantiene la velocidad de salida $v$ y el ángulo $\alpha$ para cada punto.

Eventualmente una partícula caerá dentro de la fuente y da lugar a otra salpicadura que será el punto de naciemiento de otra partícula o grupo de partículas, y este punto debe ser detectado y representado por el procedimiento de animación.

Técnicas similares pueden ser usadas para imitar las explosiones de fuegos artificiales donde un grupo de partículas son lanzadas desde un cohete en explosión. En este caso, la posición final del cohete es registrado y las partículas lanzadas con una velocidad que es desiminuida por un valor fraccional; también se debe tomar en cuenta la resistencia del airte que hace que la explosión rápidamente desaparezca. El arrastre de la gravedad puede ser tomado en cuenta para simular la caída de la partícula al suelo.

## 7.7.2 MODELOS DE CRECIMIENTO

Para ejemplificar, se considera el problema del crecimiento de un árbol con un tronco sólido, y un sistema de ramas con hojas. Una técnica que se puede utilizar para modelar y simular el sistema de ramas es el uso de curvas de Bézier. Esto se puede ilustrar mejor con un ejemplo en dos dimensiones.

La Figura 7.11 muestra en tronco delimitado por los puntos $T_1$ y $T_2$, de donde crece un número determinado de ramas. Para asegurarse de que cualquier rama crezca del tronco con un cierto grado de inclinación continua, hay cuatro puntos de control que forman una curva cúbica de Bézier que ayudan en este proceso; estos puntos son $B_1$, $B_2$, $B_3$ y $B_4$. Nótese que la inclinación de la línea que conecta $B_1$ y $B_2$ es idéntica a la inclinación del tronco. Los puntos de control se seleccionarán usando un procedimiento que asegura esta inclinación continua aplicando un comportamiento estocástico para otros puntos, junto con la longitud de la rama.

Cualquier número razonable de ramas similares pueden ser permitidas para

crecer desde el tronco para crear el primer conjunto de ramas. En la misma figura, los cuatro puntos de control $C_1$, $C_2$, $C_3$ y $C_4$ sirven para controla el crecimiento de la rama, y la inclinación de la línea que conecta $C_1$ con $C_2$ es igual a la inclinación de la línea que conecta $B_3$ y $B_4$. Este proceso de bifurcación puede ser continuo para cualquier trayectoria para lograr el nivel de detalle requerido.

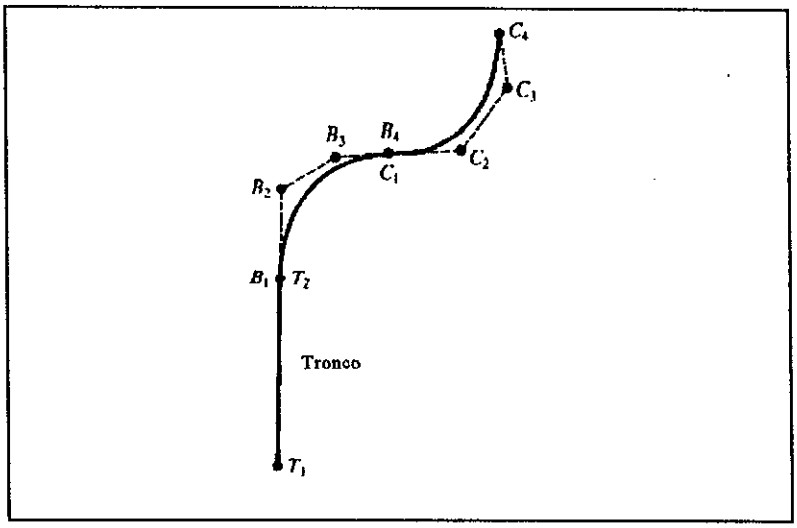



**Figura 7.11**: la primera rama que crece del tronco continúa creciendo en la misma dirección porque la inclinación de la línea que conecta $B_1$ y $B_2$ es igual a la inclinación de la línea que conecta $T_1$ y $T_2$. La curva cúbica de Bézier está definida por los cuatro puntos de control para el crecimiento y animación requeridas. Igualmente, la rama está definida por los puntos de control $C_1$, $C_2$, $C_3$ y $C_4$.

Como se mencionó anteriormente, el modelo no solamente se logra en forma automática, pero el árbol puede ser animado para mostrar su crecimiento en el tiempo, y también como reacciona a las fuerzas creadas por el viendo y gravedad. El modelo no está terminado solo con el trazado de líneas. El éxito de la técnica de modelación del árbol estará determinado por el que el árbol sea reconocido como un árbol natural; esto es logrado usando las reglas estocásticas para controlar la longitud de las ramas y ángulo de inclinación de las mismas.

Existe una técnica alternativa propuesta por Gordon Selley (Selley, 1990); dicha técnica de crecimiento de las ramas se muestra en la Figura 7.12, donde una rama antigua da lugar al crecimiento de una rama nueva por la selección de ángulos de divergencia y filotaxia; éstos son obtenidos usando procedimientos estocásticos que controlan los valores máximo y mínimo para el crecimiento de diferentes tipos de árbol.

Existe un algoritmo de definición de textura desarrollado por Selley (Selley, 1991); este algoritmo es usado para perforar hoyos en cualquier polígono. Otra técnica, es la descrita por D. Peachey (Peachey, 1985), que describe la textura de un cubo asociado a un objeto o escena. La textura del cubo toma forma de una tabla de números aleatorios, que sirve para describir detalladamente la superficie durante una representación.

La lámina 5 (ver apéndice) ilustra la efectividad de esta técnica: en primer plano, los árboles fueron modelados con esta técnica, mientras que en el fondo, los árboles están cubiertos por una ligera niebla.

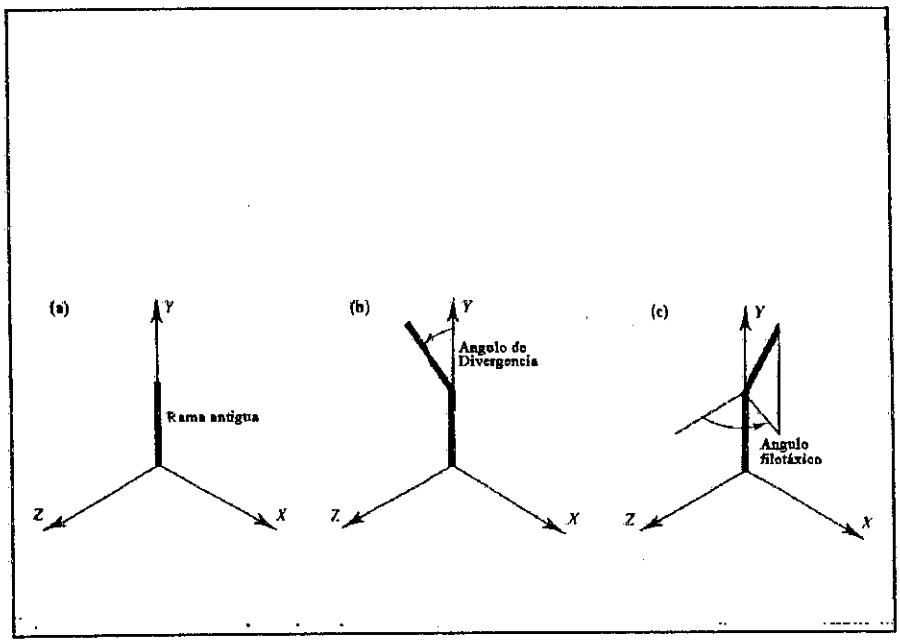



**Figura 7.12**: en (a) el sistema de ejes está alineado con la ramas antiguas y una rama nueva está creciendo por medio de un ángulo de divergencia (b) y un ángulo filotaxis rotacional (c). Este modelo es repetido hasta que se hayan creado suficientes ramas.

# CAPITULO 8

# TÉCNICAS ESPECIALES DE ANIMACIÓN

## INTRODUCCIÓN

En la animación tradicional, cada línea dibujada transmite un mensaje codificado de forma, peso y velocidad. Una historia es traducida de una secuencia de escenas estáticas en un mundo imaginario llevado a la vida por la experiencia del animador, que es el resultado de una habilidad innata para dibujar, respaldado por observaciones personales de actividades diarias y una cantidad de trucos adquiridos a través de la experiencia práctica. Esto le permite al animador conformar dibujos en formas y posiciones raras; las reglas de la perspectiva son flexibles, los objetos aparecen de ninguna parte, y las leyes de la naturaleza física pueden ser suspendidas sin aviso. Todo esto es logrado simplemente si se hacen marcas sobre hojas de papel.

La Animación por Computadora en 3-D no posee este grado de flexibilidad; la realidad virtual dice que tiene que ser construido a base de polígonos y partes que son animadas por matrices y quaternions, y finalmente realizados para representar formas geométricas precisas necesarias para cualquier ejecución de sus cálculos. Las herramientas para dibujar gráficas por computadora consiste de teclados, joysticks, mouse y digitalizadores; un conjunto totalmente nuevo de técnicas deben ser perfeccionadas por el animador para animar su mundo virtual. Además, algunos procedimientos y manipulaciones matemáticas deben ser un recurso para simular los efectos tradicionales de animación; lo anterior es el tema de este capítulo.

## 8.1 ANIMACIÓN DE UN CUADRO PRINCIPAL

Esta es una técnica conocida en la animación tradicional donde una secuencia de animación es desarrollada por inbetweening de dos dibujos que representan el cuadro inicial y el final. Estos frames deben tener un nivel razonable de imágenes coherentes; de otra forma el inbetweener no tendrá suficiente información para crear las nuevas imágenes. En la Animación por Computadora, lo único que no es almacenado numéricamente son los nombres de los objetos, lo que permite a un par o secuencia de valores ser interpolados. Esta técnica es usada en la Animación por Computadora, y todos los sistemas profesionales proporcionan facilidades donde el animador puede especificar aquellos parámetros que van a ser interpolados.

## 8.2 PROCESO DE ANIMACIÓN

Al hablar sobre estructuras enlazadas se sabe que, dados ciertos mecanismos físicos, estas estructuras podrían ser automáticamente animadas, si se alteran ciertos parámetros como la rotación angular de una rueda. Este tipo de animación tiene el nombre de 'animación procedural', el cual abarca cualquier sistema que pueda ser manipulado por reglas contenidas en un programa.

Otros ejemplos de animación procedural se encuentran en los sistemas de control de partículas, crecimiento de plantas, simulación dinámica de sistemas físicos.

Una técnica procedural simple puede ser usada para oscilación de péndulo, rebote de pelotas, aceleración de cohetes, vibración de objetos y cualquiera que obedezca a las leyes simples de mecánica; un ejemplo, son las manecillas de un

reloj que son posicionadas de acuerdo con un tiempo específico.

Para empezar, hay que decidir cómo se especificará el tiempo; para ello se usará el sistema de 24 horas, minutos y segundos, donde el ciclo de horas es de 0 a 23, y el ciclo de minutos y segundos es de 0 a 59. Se asumirá que 24 hrs 00 min 00 seg es equivalente a 00 hrs 00 min 00 seg. Esta tripleta del tiempo será traducida a tres ángulos para rotación de los segundos, minutos y horas.

La traducción de las unidades de tiempo [hrs,min,seg] a ángulos en unidades de grados [$ß_{hrs}, ß_{min}, ß_{seg}$] es la siguiente:

$$ß_{seg} = 6\ seg$$
$$ß_{min} = 6(min + seg/60)$$
$$ß_{hrs} = 30(hrs + min/60 + seg/3600)$$

Esto puede ser probado al substituir el siguiente ejemplo [2hrs,45min,30seg], el cual produce el ángulo [82.75°,273.0°,180.0°]. El segundero será rotado 180°, que corresponde a 30 segundos; el minutero es rotado 273°, que está formado por 270° que son 45 minutos, y los 3° extras equivalen a los 30 segundos barridos por el segundero, y la manecilla de la hora es rotada 82.75° que similarmente comprende los efectos de los segundos y minutos agregados a las horas. Estos ángulos pueden ahora ser usados para rotar los elementos para representar las manecillas del reloj, si se asume que su posición inicial está en la posición de las 12 en punto. De cualquier modo, se debe recordar que una rotación angular positiva está en el sentido contrario de las manecillas del reloj, lo que significa que las matrices de rotación de las manecillas deben ser inversas a los signos de los ángulos.

Ahora se necesita un mecanismo para manejar las manecillas para el actual número de frames, y tal vez el reloj tiene que empezar a una hora en particular. Hay que asumir que en el frame 1 la hora inicial es 4 hrs 0 min 0 seg, entonces para cada frame subsecuente, el parámetro de los segundos debe ser incrementado por alguna cantidad que depende de la velocidad de la animación. Si se toma la velocidad de 25 frames/seg, entonces cada frame es equivalente a 0.04 segundos. Además, en el frame *N*, deberán transcurrir 0.04*N* segundos, que deberá ser convertido en segundos, minutos y horas. Se puedę crear un procedimiento, dado el tiempo en el frame específico; juntos con el actual frame, puede automáticamente posicionar las manecillas para representar el tiempo real transcurrido. Si la llamada al procedimiento fuera *the_time*, entonces para el ejemplo la llamada sería:

```
the_time(1,4,0,0,N),
```

que indica al procedimiento que en el frame 1 la hora fue 4 hrs 0 min 0 seg, y las manecillas del reloj deben ser colocadas para el N-ésimo frame.

El ejemplo anterior establece el nivel de análisis necesario para diseñar procedimientos; frecuentemente requieren de un desarrollo considerable. En el caso del reloj, las manecillas deberían ir hacia atrás si el signo de los ángulos no es opuesto; una manecilla puede permanecer fija en ciertas horas, o simplemente no se desea mover. Estas son las realidades de la programación y desarrollo de software.

# 8.3 DEFORMACIÓN

La deformación free-form es una que tiene una mayor influencia sobre el estado del arte del diseño.

PICASO (Vince, 1976) incluye un comando WARP3D, el cual expone los objetos en 3D a diferentes acciones sobre las coordenadas *xyz*. Por ejemplo, las coordenadas *x* pueden ser procesadas por una función lineal, las coordenadas *y* pueden ser extendidas, y las coordenada *z* comprimidas. Este comando permite a

cualquier objeto ser deformado solamente, si se ajustan los parámetros en las funciones.

Scott Parry (Parry, 1986) mostró cómo un modelo CSG puede ser manipulado si se usan funciones de deformación, y una técnica que usa dichas funciones es la SIGGRAPH 1986 (Sederberg, 1986). Dentro de un período de tiempo corto ha sido utilizado en varias secuencias de animación, en particular 'Locomotion' (Pacific Images), que es mostrada en la lámina 10. (ver apéndice).

Los efectos de la deformación free form (FFD) no son totalmente nuevos; Albert Durer en su 'Treatise on Proportion' utilizó redes rectangulares para ilustrar cómo el rostro humano puede ser distorcionado por la red para generar siluetas relacionadas. D'Arcy Thompson (Thompson, 1961) en su libro On Growth and Form utilizó la teoría de las transformaciones para comparar formas naturales como hojas, ovejas y jirafas, crustáceos, peces. La Figura 8.1 muestra un ejemplo de este concepto. La idea principal del FFD es encerrar un objeto dentro de un enrejado de control de puntos.

Primero se mostrará cómo funciona esta técnica para un ejemplo en una dimesión, y luego para dos y tres dimensiones. Para ello, hay que considerar una secuencia de puntos que pueden ser deformados por las posiciones de los tres puntos de control.

La Figura 8.2 muestra un simple eje-$x$ con una línea dibujada entre los valores 2 y 4. Se podría imaginar a la línea como un conjunto infinito de puntos entre estos dos límites. Para lograr esto, los tres puntos de control $P_1$, $P_2$ y $P_3$ están posicionados en el medio y los extremos. Cualquier punto $x$ a lo largo de la línea puede ser ahora expresado como una fracción $t$ de la siguiente forma:

$$t = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

donde $x_{min} = 2$, y $x_{max} = 4$.

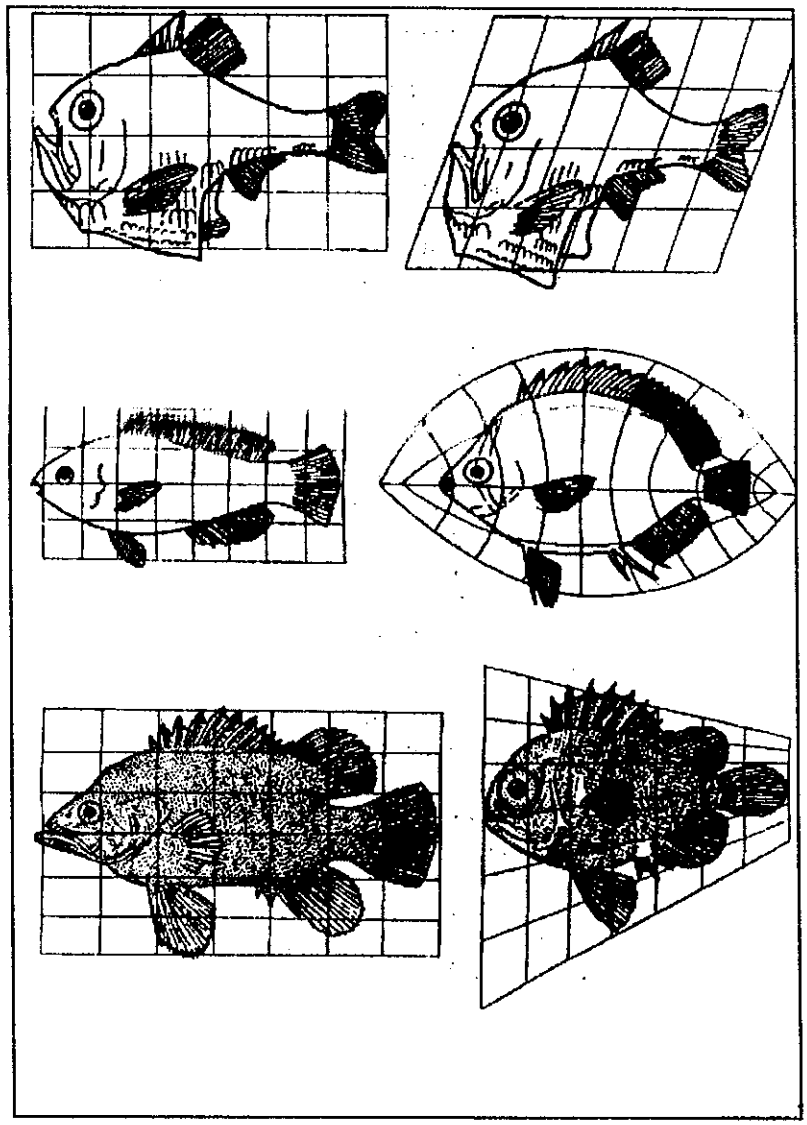


**Figura 8.1:** estas ilustraciones del libro On Growth and Form de D'Arcy Thompson muestran la idea de la teoría de las transformaciones.

La posición de cualquier punto deformado puede ser definido como:

$$x' = P_1(1 - t)2 + P_2(1 - t)2t \qquad + P_3t^2$$

Si se substituyen diferentes valores de $x$ para obtener valores equivalentes de $t$, se verá que $x' = x$, además, existe una distorsión cero. Si el punto de control central es adaptado a 3.5 en lugar de 3.0, todos los puntos entre los dos puntos extremos serán movidos. La Tabla 8.1 muestra los cambios que ocurren.

Se puede ver que estos valores de cada punto, excepto los puntos extremos, han sido aumentados por diferentes cantidades, y si el punto de control central ha sido establecido en 2.5, entonces los puntos tendrán que disminuir y ser movidos a la izquierda. Una vez que esto haya sucedido, si uno o ambos puntos extremos es movido, entonces hay que mantener $P_1 = 2$, $P_2 = 3$ y $P_3 = 5$. Esto produce los valores mostrados en la Tabla 8.2; de esto, se puede ver que la línea entera de puntos ha sido extendida a lo largo del punto de control $P_3$.

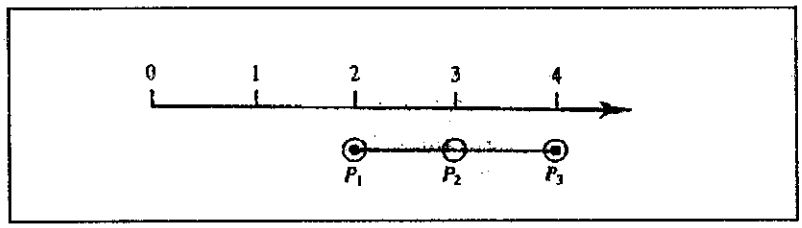



**Figura 8.2**: si existe una línea entre los valores 2 y 4, los tres puntos de control $P_1$, $P_2$ y $P_3$ pueden ser usados para deformar la distribución de los valores entre estos límites.

**Tabla 8.1**

| $x$ | $t$ | $x'$ |
|---|---|---|
| 2.0 | 0.0 | 2.00 |
| 2.2 | 0.1 | 2.29 |
| 2.4 | 0.2 | 2.56 |
| 2.6 | 0.3 | 2.81 |
| 2.8 | 0.4 | 3.04 |
| 3.0 | 0.5 | 3.25 |
| 3.2 | 0.6 | 3.44 |
| 3.4 | 0.7 | 3.61 |
| 3.6 | 0.8 | 3.76 |
| 3.8 | 0.9 | 3.89 |
| 4.0 | 1.0 | 4.00 |

**Tabla 8.2**

| $x$ | $t$ | $x'$ |
|---|---|---|
| 2.0 | 0.0 | 2.00 |
| 2.2 | 0.1 | 2.21 |
| 2.4 | 0.2 | 2.44 |
| 2.6 | 0.3 | 2.69 |
| 2.8 | 0.4 | 2.96 |
| 3.0 | 0.5 | 3.25 |
| 3.2 | 0.6 | 3.56 |
| 3.4 | 0.7 | 3.89 |
| 3.6 | 0.8 | 4.24 |
| 3.8 | 0.9 | 4.61 |
| 4.0 | 1.0 | 5.00 |

En dos dimensiones, se utiliza una matriz de puntos de control en la misma forma que para las superficies de Bézier. Para una interpolación cuadrática, se usa una matriz de 3 x 3, y para una cúbica una matriz de 4 x 4, el siguiente ejemplo utiliza una forma cuadrática.

Se necesitan los parámetros *u* y *v* para controlar el espacio 2-D donde cualquier punto $P'(x',y')$ está definido como:

$$P'(x',y') = [(1-u)^2 \; 2u(1-u) \; u^2] \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} (1-v)^2 \\ 2v(1-v) \\ v^2 \end{bmatrix}$$

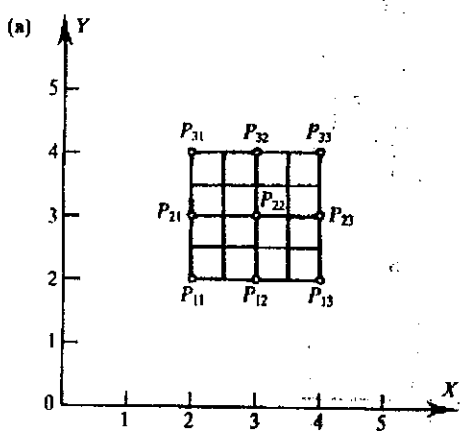
donde:

$$u = \frac{(x - x_{min})}{(x_{max} - x_{min})} \qquad v = \frac{(y - y_{min})}{(y_{max} - y_{min})}$$

La Figura 8.3a muestra una matriz de 3 x 3 de puntos de control superimpuesta en una red que representa la figura a ser deformada. Con los puntos de control como se muestra, la red es distorsionada, pero si el punto central $P_{22}$ es movido de (3.0,3.0) a (3.75,3.75) ocurre la deformación que se muestra en la Figura 8.3b. Si el punto de control $P_{33}$ es movido de (4.0,4.0) a (5.0,5.0) la distorsión se muestra en la Figura 8.3c.

Esto claramente demuestra cómo una figura en 2-D puede ser deformada si se cambian uno o más puntos de control, y para obtener un mejor control, podría ser usado un esquema de interpolación cúbica que necesita de nueve puntos de control. Si se intercambian diferentes conjuntos de puntos de control, se puede obtener una secuencia de animación contínua.

Para tres dimensiones, se necesitan tres parámetros, *u*, *v* y *w*, para controlar las direcciones de *x*, *y* e *z* y las redes en 3-D de puntos de control. Si se necesita una interpolación cuadrática en las tres direcciones, se necesitarán 27 puntos de control como se muestra en la Figura 8.4 -cualquier punto $(x',y',z')$ dentro de este volumen, puede ser deformado al sumar los términos polinomiales de Bernstein que multiplican los valores de los puntos de control.



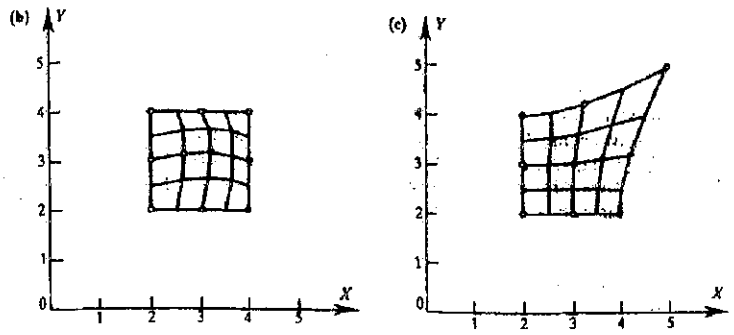



**Figura 8.3**: la mezcla de nueve puntos de control mostrados en (a) pueden ser usados para deformar la distribución espacial de cualquier punto encerrado. En (b), el punto de control central $P_{22}$ ha sido movido, mientras en (c) el punto $P_{33}$ también ha sido movido.

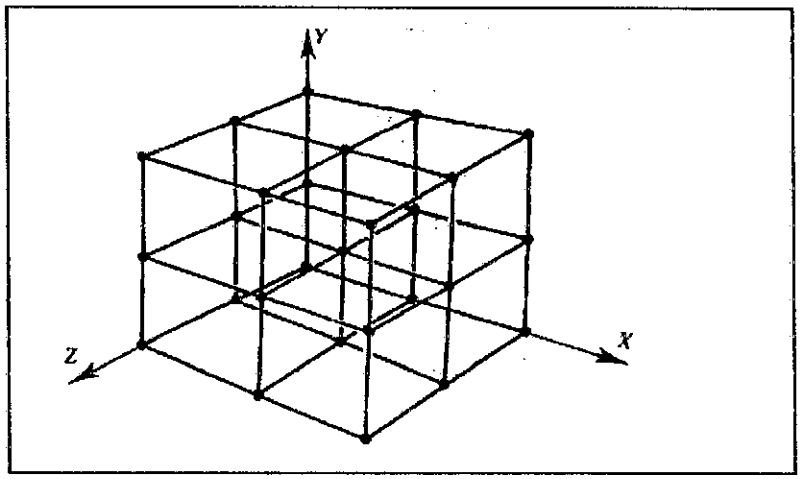



**Figura 8.4**: este diagrama ilustra como una red de 27 puntos de control es necesaria para deformar un volumen de espacio al usar FFDs cuadráticos.

# 8.4 ANIMACIÓN POR SUBSTITUCIÓN

La animación por substitución proporciona una forma simple y útil para desarrollar secuencias animadas al asociar vectores con los vértices de la superficie de un objeto. El vector define el camino a lo largo del cual el vértice se desplazará según algunos parámetros definidos por el usuario, y pueden ser creados manualmente, o lo más probable de dos posiciones intermedias de la superficie. El vector puede ser el vector resultado creado de dos más substituciones. Se verá un ejemplo en 2-D.

Se considera, una línea como se muestra en la Figura 8.5a que contiene los vértices *A, B, C, D* y *E*, donde cada vértice tiene un vector asociado para controlar su recorrido. Nótese que los vértices de los extremos, no parecen tener

un vector asociado; si existen sus vectores pero son de magnitud cero. La substitución asignada para cada vértice x_disp y y_disp pueden ser ahora controlados por un parámetro *t*, el cual varía entre 0 y 1, de la siguiente forma:

```
x_disp = tx_comp
y_disp = ty_comp,
```

donde x_comp y y_comp son los componentes *x* e *y* del vector. Cuando t = 0.5, cada vértice es desplazado la mitad a lo largo del vector para producir la forma mostrada en la Figura 8.5b, y cuando t = 1.0, cada vértice se mueve al final de sus respectivos vectores como se muestra en la Figura 8.5c. De cualquier forma, no se puede saber si el parámetro excede de 1, o es negativo.

Una animación por substitución involucra vectores, y se pueden usar las reglas de suma y resta de los vectores creando una substitución resultante de substituciones individuales. Si las magnitudes de estas substituciones son obtenidas a través de la acción de separar parámetros, se tendrán movimientos muy complejos. La Figura 8.6 muestra una secuencia de rostros animados al usar herramientas de animación por substitución.

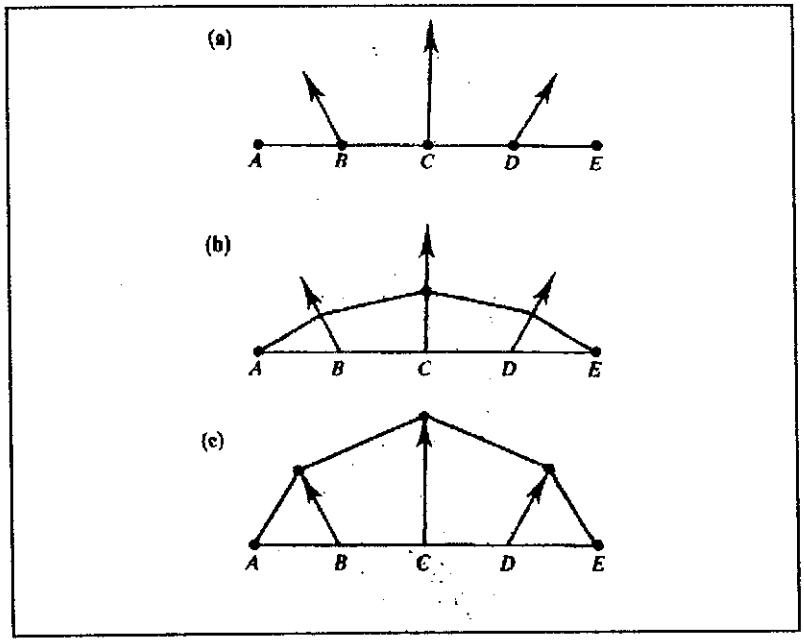



**Figura 8.5**: en la figura (a) son asignados vectores a cada vértice, puede ser animado si se usa un parámetro que mueve cada vértice a lo largo del vector una cantidad proporcional como se muestra en (b) y (c). Los vértices de los extremos permanecen fijos, ya que sus vectores son de magnitud cero.

# 8.5 COMPORTAMIENTO DE LA ANIMACIÓN

El Behavioural animation destina la simulación para patrones de comportamiento mostrados por ciertas especies de animales. Fue investigado por Susan Amkraut y Micheal Girard (Amkraut, 1985), en la secuencia de animación 'Eurythmy', forzaron campos para ser utilizados para influenciar el comportamiento de una bandada de aves volando sobre un patio. Durante su vuelo, las alas de las aves se mueven en una forma natural, mientras simultáneamente evitan el contacto con otras aves. Un comportamiento similar ha sido investigado por Craig Reynolds (Reynolds, 1987), cuyo trabajo en el Symbolics Graphics Division proporcionó el software para la secuencia de animación 'Stanley and Stella: Breaking the Ice', el cual utilizó bandadas de aves y bancos de peces animados por los algoritmos de comportamientos de bandadas.

El animar grandes grupos de objetos que no se dan cuenta uno del otro. En

el caso de sistemas de partículas que pueden contener cientos o miles de elementos discretos, cada partícula vive en su propio mundo y no se preocupa de la existencia de las demás partículas.

De cualquier forma, la naturaleza estocástica del nacimiento, edad y muerte de las partículas, junto con algunas leyes físicas de simulación reales, pueden resultar interesantes para alguna secuencia de simulación. Si las partículas son ahora reemplazadas por objetos reconocidos como un ave, que está aleteando sus alas, los movimientos de las aves podrían ser influenciados por su alrededor inmediato, y por cualquier otra regla que controla su comportamiento.

Dicho grupo o comportamiento de bandada es muy complicado en grandes bandadas de estorninos volando y que no existe un lider establecido; la bandada se mueve a base de señales rápidamente transmitidas a través del diámetro de la bandada comunicando que ha ocurrido un cambio de dirección. De alguna forma, el resto de la bandada permanece junta aún y cuando sufre rápidas y complejas maniobras, y es este comportamiento que es simulado en el modelo de comportamiento de banadada.

Algunos de los principios del algoritmo puede ser entendido, si se considera un escenario en 2-D ilustrado en la Figura 8.7. Aquí se ve un punto *P* rotando con un radio constante alrededor de un punto fijo, y un punto *Q* a su derecha que se trata de mantener con *P* cuando rota. Como no hay una conexión física entre *P* y *Q*, un conjunto de reglas se necesitan para permitir esta forma de comportamiento.

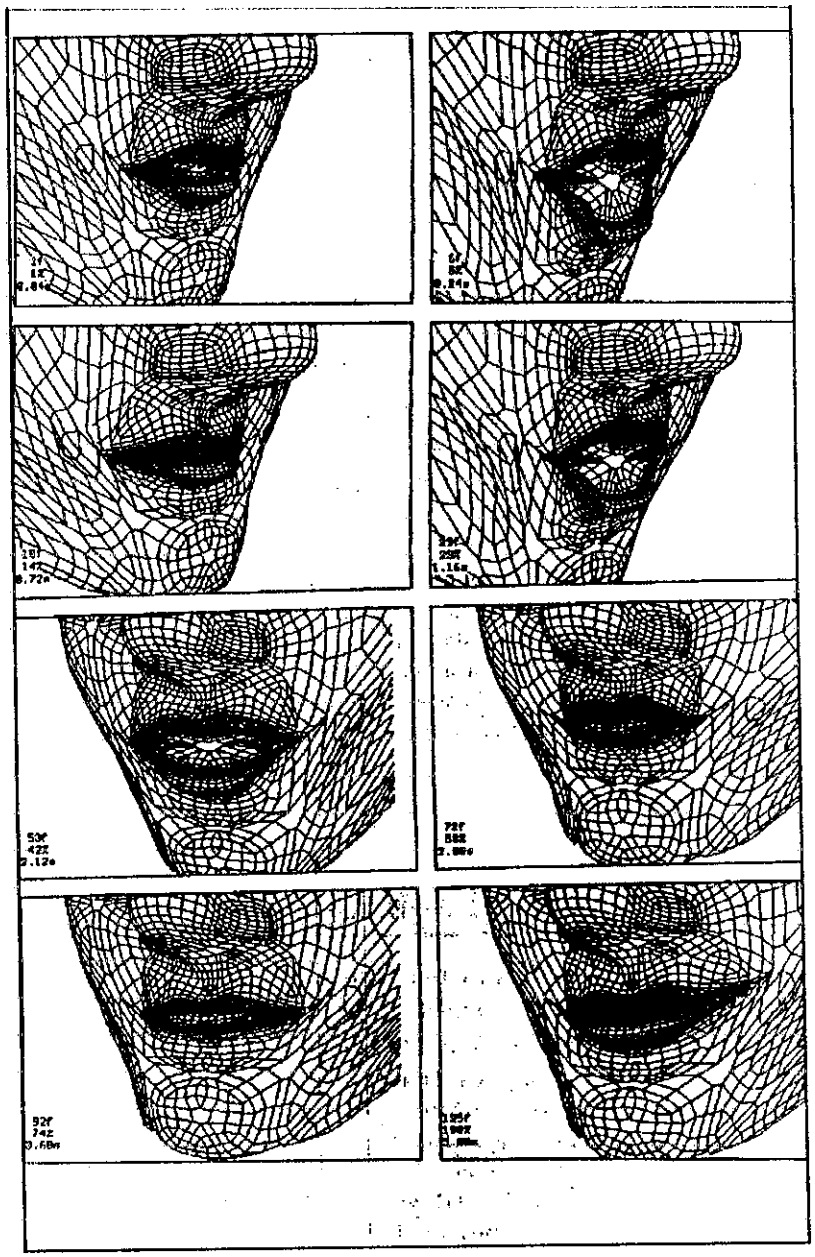


**Figura** 8.6: esta secuencia de ilustraciones son ejemplos de animación por substitución creadas por el sistema S-Animation Symbolics.

Como parte de dichas reglas se usarán unos parámetros tales como las posiciones previas y actuales de ambos puntos. Se necesitarán sus velocidades, pero lo más importante es incluir una regla que describe cuán cerca y lejos pueda estar *Q* de *P*, las reglas se pueden aplicar de la siguiente forma:

> *If Q continúa moviéndose con su velocidad actual, se moverá más cerca o más lejos de P?*

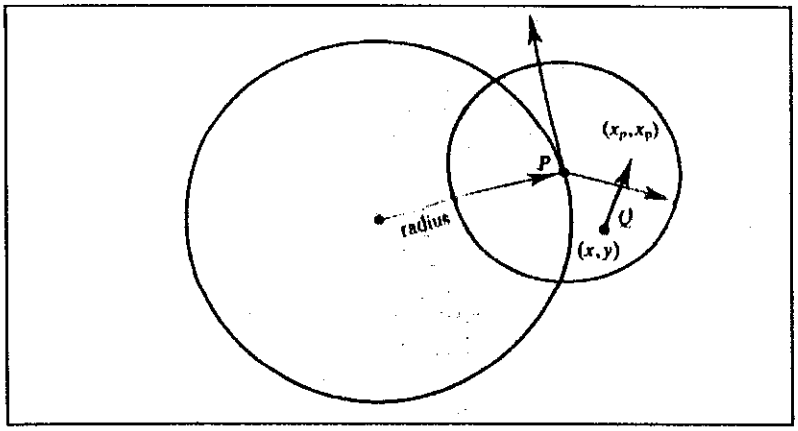



**Figura 8.7**

En cualquiera de los casos, hay que analizar si se cumple la regla del radio, se le permitirá a *P* moverse a su nueva posición rotada y se examinará la nueva posición espacial entre *P* y *Q*. Si *Q* se mueve más cerca o más lejos de P, la acción se resolverá modificando la velocidad de *Q*. Las cantidades por las cuales el comportamiento de *Q* son modificados pueden ser críticos: si es mucho, el sistema oscilará más; si es muy poco las fuerzas dinámicas colapsarán completamente, aunque exista una cantidad de valores que permiten tener un rango de comportamiento adecuado.

Ahora el sistema puede ser aumentado introduciendo más puntos cuyos atributos son matenidos en una tabla, y algunas reglas podrían ser incorporadas para controlar el comportamiento de grupo. Craig Reynolds utilizó la técnica que evita una colisión, el centro de bandada y similitud de velocidades. El evitar una colisión crea el deseo de evitar un impacto con un objeto o con otra ave; la similitud de velocidades complementa el evitar una colisión forzando en asimilar las velocidades de las aves vecinas, y el centro de bandada proporciona el deseo de ir hacia el centro de la bandada, lo que podría ser interpretado como una forma de preservación propia. Estas características deberán ser trasladadas en cambios de posición, velocidad, para lo cual se necesita un rango en el orden de importancia. Por ejemplo, un punto de la periferia de la bandada podría tener tres razones para moverse a la próxima posición: colisión inminente, alta velocidad relativa y la posibilidad de que un depredador ataque.

La magnitud de cualquier modificación en el comportamiento puede ser controlado por funciones que miden el potencial de un campo de fuerza. En el caso de la radiación electromagnética, se sabe que la intensidad de la luz disminuye con el cuadrado inverso de la distancia. Se pueden introducir relaciones similares al aplicar una escala no lineal para medir la importancia de varias condiciones. Reynolds utilizó un campo exponencial inverso centrado en el sistema local de coordenadas del ave para influenciar la sensitividad para colisionar. Tales campos de fuerza pueden se influenciados por la velocidad del objeto para que la dirección de vuelo sea más pronunciada.

# 8.6 SIMULACIÓN DINÁMICA

En la animación de dibujos animados, los mundos imaginarios creados por los animadores son generalmente inimaginables y la pregunta que se hacen es cómo las leyes de la física son rotas en un ambiente no material. Uno de los problemas que tienen los animadores es convencer al público que las cosas son reales, como un yunque que cae. Si el movimiento no se puede especificar correctamente, entonces el atributo que dominará -en caso de la masa- es normalmente cambiado a un tamaño exagerado, pero desafortunadamente, el hecho de que algo es grande no implica que sea pesado. Un paracaídas es más grande que un yunque y cae por la gravedad en una forma distinta. Entonces, un yunque grande, si se le da un movimiento errado, puede crear la impresión de que no tiene peso.

La especificación de movimiento, como se ha visto, no es muy difícil y hay muchas herramientas matemáticas disponibles para trasladar y rotar un objeto de una posición a otra. Una moneda que cae en una mesa puede primero dar vueltas y luego caer. Cuando toca la mesa, su futuro comportamiento depende de su velocidad

final, energía rotacional, su momento de inercia y otros parámetros. Eventualmente termina su recorrido girando en un punto, decrementando su frecuencia rotacional antes de perder todo su potencial y energía cinética a través de la fricción. ¿Es este grado de realismo importante en la animación? Tal vez no, en el caso de la moneda, pero cuando se trata del movimiento articulado de la simulación humana, sí es importante.

El esfuerzo para obtener un nivel alto de realismo está en la satisfacción de convertir algo sintético a la vida, y así destaca la dificultad de simular cada matiz de algún comportamiento simulado. En el caso de la locomoción humana, el movimiento de los brazos, el cuerpo oscila en el cambio de su centro de gravedad, la cabeza se inclina con la acción de las piernas, y los pies empujan firmemente contra el piso con una fuerza de dirección.

Una alternativa para estas técnicas es simular las dinámicas asociadas con un sistema de unión de elementos. Esto permite que los atributos, propiedades y leyes físicas como masa, inercia, rigidez y gravedad sean usados en un programa de computadora.

Los conceptos necesarios para investigar la simulación dinámica son el grado de libertad (degree-of-freedom, DOF por sus siglas en inglés), dinámica, limitaciones, dinámica, y energía cinética. El término grados de libertad está asociado con puntos abastractos, uniones entre objetos y sistemas de enlace. Cuando algo está descrito en términos de tres DOF, significa que se necesitan tres parámetros independientes para controlar su posición espacial.

Por ejemplo, un simple punto requiere tres coordenadas para fijar su posición en el espacio. Como el punto es un concepto abstracto, no tiene tamaño y orientación, consecuentemente los tres números son suficientes para describir su posición espacial, por lo que se dice que tiene tres DOF (degree-of-freedom, grados de libertad). Otro ejemplo, un anillo circular que puede deslizarse por un asta tiene dos DOF, como se necesitan dos parámetros para controlar su posición a lo largo del asta y su rotación angular alrededor de los ejes del asta. Finalmente, cualquier objeto volador tiene seis DOF, como se necesitan tres coordenadas para posicionarlo en el espacio y tres ángulos para fijar su orientación.

Como un robot, un maniqui puede ser construido a base de un sistema de enlaces que represetan los brazos, piernas, pies, manos, cuerpo y cabeza conectados todos por uniones. En realidad la rodilla, codo, cadera, muñeca o tobillo son uniones complejas que tienen más de un DOF, la mobilidad de las uniones está restringida por su propia forma, los músculos y tendones asociados. Estas limitaciones físicas resultan ser parte de un modelo de computadora para asegurarse de que el maniquí no pierda su figura.

En un ambiente de animación por computadora el animador especificará las limitaciones cinéticas para las uniones, para garantizar que no se excederán las velocidades lineal y angular. Algunos movimientos pueden ser simulados al introducir uniones rígidas y masas para unirlas cuando sean usadas en uniones flexibles para algún movimiento.

Cuando se habla de un sistema cinético, se refiere a su movimiento; en el caso de una estructura unida, el animador tiene la dificultad de especificar la velociad y aceleración de cada unión sobre un período de tiempo. Por ejemplo, el problema de animar la locomoción humana. Una forma simple, pero no efectiva de crear el movimiento, es definir el estado de las piernas en sus posiciones extremas, e interpolar los parámetros de posición para obtener las posiciones intermedias. El problema con este ejemplo es que no se puede garantizar que los pies tocarán el suelo, y si lo hacen, de que no se deslicen o que no traspasen el suelo.

El sistema PODA, desarrollado por Michael Girard y A.A. Maciejewski (Girard, 1985), proporciona un ambiente interactivo en el cual el animador puede especificar las trayectorias y lugares de apoyo de las uniones finales, en donde el sistema PODA calcula el movimiento de las uniones interiores usando cinética inversa. El problema principal para aplicar tales técnicas son las ecuaciones no lineales que resultan de las soluciones analíticas. PODA logra esto al asumir que el estado de las uniones de dos puntos diferentes a la vez están linealmente relacionados si el intervalo de tiempo es pequeño. Así, los cambios incrementales en la posición pueden ser relacionados por cambios incrementales en orientaciones

usando una matriz llamada Jacobina. Esta matriz cambia a cada intervalo de tiempo, y como PODA utiliza cinética inversa, el Jacobiano tiene que ser invertido a cada intervalo de tiempo.

La matriz no permite un inverso exacto, solo obtiene un pseudo-inverso, que matemáticamente hablando es aceptable, y proporciona una animación deseada. PODA proporciona al animador un control sobre patrones de pasos como cuando se corre o camina, trayectoria de una bicicleta, lugares de apoyo y trayectoria del cuerpo. PODA fue usado para producir la animación 'Eurythmy'. La lámina 10 muestra un frame para esta animación.

El sistema DYNAMO desarrollado por Paul Isaacs y Michael Cohen (Isaacs, 1987) utiliza las técnicas de dinámica inversa, que permite al animador asociar la masa de las uniones con la estructura enlazada, junto con las limitaciones cinéticas y los campos de fuerza gravitacional.

# CAPITULO 9

# SISTEMAS DE ANIMACIÓN

## INTRODUCCIÓN

En los últimos años, varios investigadores han obtenido soluciones importantes para problemas asociados con gráficas por computadora; esto ha sido logrado por la labor de Brensenham con su trabajo de despliegue de líneas rectas, el descubrimiento de fractales de Mandlebrot. Paralelo con estas investigaciones, ha habido una gran variedad de proyectos cuyo objeto ha sido diseñar sistemas para unificar al usuario en un ambiente unificado, en el cual las gráficas por computadora pueden ser utilizadas como una ayuda para diseñar y visualizar objetos en 3-D. Actualmente, es posible comprar un amplio rango de paquetes de software que pueden ser usados en publicidad, diseño de gráficos en televisión, arquitectura, diseño por computadora, y por último la animación.

El software permite que estas actividades sean diseñadas para trabajar con una variedad de computadoras, desde las microcomputadoras más sencillas hasta una supercomputadora; esto incluye diferencias de costos, velocidad, eficiencia, efectividad y un fácil uso. El desarrollo y mantenimiento del software es caro, y en el caso de la animación se necesita actualizarse con técnicas nuevas. Actualmente, la tendencia es seguir combinando la supercomputadoras y paquetes de software, que proporcionar facilidades para modelos de animación, modelos dinámicos y modelos de representación. Estos sistemas generalmente permiten al usuario de agregar programas extras para implementar efectos especiales, que no son parte de algún sistema de animación.

Existen técnicas geométricas y matemáticas usadas para modelar, para animar, para el control de la cámara, iluminación y efectos especiales, así como también matrices, polinomios, ecuaciones paramétricas, vectores, quaternions y Jacobianos. Dichas técnicas son importantes para un sistema de animación. Las técnicas gráficas deben ayudar a la interpretación de curvas, niveles de iluminación, color, velocidad, aceleración y modelos dinámicos. En este capítulo, se explicarán algunos sistemas de Animación por Computadora desarrollados, y de algunos sistemas que comercialmente son importantes.

## 9.1 ANTECEDENTES HISTÓRICOS

En la década de los '60s, los animadores por computadora no tenían acceso a las técnicas de hardware y software disponibles en la actualidad. La interpretación de colores no ha evolucionado, lo que significa que las imágenes son realmente líneas dibujadas; esto era fotografiado o capturado por una cinta de video. La computación interactiva, como se sabe, no existía; los programas estaban sujetos al uso de tarjetas perforadas. Los programas se escribían en lenguaje ensamblador, FORTRAN, u otros lenguajes, tales como Algol y Pascal, actualmente, el lenguaje C y C++ parecen ser los lenguajes más usados para desarrollar nuevos sistemas de gráficas por computadora.

Cuando se escribía en un lenguaje de alto nivel como FORTRAN, el programador era responsable para editar cada aspecto de la animación que debía incluir el tamaño de la imagen, cálculo de las proyecciones perspectivas, almacenamiento de coordenadas, eliminación de líneas ocultas, y aplicar las operaciones de matrices necesarias, trasladar y rotar objetos. Como estas actividades eran parte de un programa de animación, es fácil desarrollar una librería de sub-programas, que podían ser usadas cada vez que se necesitaran. Esto era, y todavía lo es, una forma de desarrollar grandes programas que requieren accesos frecuentes a los mismos comandos.

Las librerías de comandos para programas de animacíon pueden ser escritos

en un período de tiempo corto. Aunque esto tiene una desventaja, las computadoras se vuelven más lentas, y esto se refleja cuando se desean actualizar varios parámetros que tienen que ver en el control de la secuencia de animación. Obviamente, es un trabajo tedioso e incita a desarrollar lenguajes para que el animador pueda usarlos para controlar directamente la computadora sin interrupciones.

La idea básica de los lenguajes de animación es permitir al animador que desarrolle diferentes secuencias de animación usando palabras como 'ROTATE' (rotar), 'TRANSLATE' (trasladar) y DRAW (dibujar), juntas sin nombres, parámetros o valores de coordenadas que describen cosas dentro de una secuencia.

Uno de los centros más importantes donde se desarrolló la Animación por Computadora en la década de los '60s fueron los Laboratorios Bell Telephone; aquí, Ken Knowlton desarrolló su lenguaje FORTRAN IV BEFLIX en una IBM 7094. Fue usado para crear películas, y en 1970 desarrolló EXPLOR como una ayuda para los artistas para crear simples imágenes en 2-D compuestas de patrones repetidos con distintos niveles de aleatoriedad. Otro trabajo desarrollado por Ronald Baecker en el MIT, que es el sistema GENESYS, identificó la importancia de expresar comandos de animación gráficamente. En los sistemas modernos donde el comportamiento dinámico es simulado, este método de descripción y comunicación es fundamental para el diseño por parte del usuario.

Otros sistemas que se han desarrollado son:

CAMP, por J. Citron y J. Whitney, 1968
CAFE, por J. Nolan y L. Yarbrough, 1968
ANIMATOR, por P.A. Talbot, 1971
ARTA, por L. Mezie y A. Zivian, 1971
MSGEN, por N. Burtnyk y M. Wein, 1971
SCANIMATE, por F.J. Honey, 1971
CAESAR, por F.J. Honey, 1971
MOP, por E. Catmull, 1972
CAAS, por E. Catmull, 1972
ANTICS, por A. Kitching, 1973
ANIMA, por C. Csuri, 1975
GRASS, por T. De Fanti, 1976
PICASO, por J. Vince, 1976
ANIMA II, por R. Hackathorn, 1977
ANTTS, por C. Csuri, 1979
TWEEN, por E. Catmull, 1979
SOFTCEL, por G. Stern, 1979
CGAL, por P. Comninos, 1980
SAS, por D. Zeltzer, 1982
ZGRASS, por T. De Fanti, 1983
BBOP, por G. Stern, 1983

La lista anterior es un testimonio para la importancia dada al desarrollo de un lenguaje de Animación por Computadora, y aún después de muchos años de investigación, no existe una interface interactiva universal usada por los animadores.

Como se mencionó anteriormente, algunas empresas de animación por computadora han desarrollado su propio software, para ser usados en computadores gráficos de alta definición.

# 9.2 SISTEMAS DE ANIMACIÓN COMERCIAL

En esta sección, se describirán cinco sistemas de animación más conocidos, usados por empresas de animación profesionales: estos sistemas son Alias, Explore, SoftImage, Symbolics y Wavefront.

# 9.2.1 ALIAS

El sistema Alias Powe Animator fue desarrollado en Alias Research, Inc. en Toronto, Canadá, y trabaja sobre una gran variedad de computadoras de Silicon Graphics como, IBM RISC Systems/6000.

## *Modelación*

El Alias Research no es conocida por sus sistemas de animación por computadora, pero por los sistemas CAD usados en cada área de la industria del diseño, industria automotríz e industria espacial. En dichas industrias, las herramientas para modelar se basan en desarrollos de descripción de superficies, lo que requiere la implementación de NURBS. NURBS es un método de descripción geométrica de la superficie, partes B-spline, y su forma rotacional (donde las ecuaciones paramétricas son expresadas como un radio de dos polinomios) permite modelar en forma precisa superficies como: esferas, cilindros. Además, los parámetros relacionados a los puntos de control pueden tener una distribución no uniforme.

El producto PowerAnimator permite al animador construir modelos a base de polígonos y del NURBS. También el PowerAnimator desarrolla extrusiones y superficies de revolución.

## *Simulación*

El ambiente de animación es creado a través de un sistema de manejo de menú, que proporciona un control directo sobre los objetos, fuentes de luz y cámaras virtuales. Cualquier parámetro que controla esto puede ser desplegado gráficamente relacionando un valor numérico con un contador de frames.

NURBS es usado para diseñar trayectorias de objetos y cámaras, y con las características de los puntos de control de inserción y modificación, los efectos de aceleración y desaceleración se implementan fácilmente. Aunque, existen ocasiones cuando los parámetros tienen que ser modificados de acuerdo a reglas o condiciones de una simulación dinámica; en estas circunstancias, el animador hace un procedimiento usando un Lenguaje de Descripción de Escena (SDL, por sus siglas en inglés); dentro de este ambiente, las características que controlan las propiedades de superficie, efectos de luz, mapas de textura y desplazamiento pueden ser modificadas.

## *Representación*

Las características que posee el PowerAnimator anticipa las necesidades del animador para secuencias previas sombreadas en tiempo real, aunque en un nivel bajo de fidelidad de la película. El QuickShade proporciona esta facilidad y permite al animador revisar la claridad entre objetos, superficies blandas, superficies que se interceptan, mientras se mueven en tiempo real.

Las fuentes de luz incluyen puntos, focos, áreas y dirección de luces con los atributos de color, brillantés y descomposición, todo esto puede ser modificado usano el SDL, o por parámetros de curvas. Los atributos de los objetos como color, resplandor y textura son declarados interactivamente usando superficies sombreadas y, como estos parámetros están disponibles para el animador a través de una secuencia, pueden ser modificados por medio de herramientas SDL.

Aparte de la alta calidad de PowerAnimator, el QuickRender, tiene dos características, RayCasting y RayTracing; RayCasting representa escenas con luces speculars y sombras usando mapas de reflexión, y RayTracing es una implementación del algoritmo clásico de trazado de rayos para revelar múltiples reflejos, refracciones y sombras. La facilidad del Paint Box permite al animador crear una

imagen retocada con una variedad de efectos de pintura incluyendo operaciones de aerógrafo y, de quitar y colocar.

Finalmente lo más sobresaliente es el conjunto de procedimientos predefinidos para crear texturas sólidas reales como: el cielo, nubes, montañas, atmosfera y texturas en 2-D. Con estas herramientas, los objetos pueden ser hechos en madera usando caoba o arce, de piedra usando jade o mármol, y sus parámetros pueden ser modificados mientras el objeto es animado.

## 9.2.2 EXPLORE

El sistema Explore de Thomson Digital Image tiene sus orígenes en la compañia de simulación Thomson-CSF y el Institut National l'Audiovisuel, y ahora es ampliamente usado a través del mundo para crear animaciones en 3-D para televisión, publicidad, simulación, arquitectura y diseño industrial.

El sistema actual, Explore V2.3, corre en gran rango de computadoras Silicon Graphics, desde el Personal Iris hasta el Power Series, y proporciona al usuario características de modelación de polígonos y superficies, animación en 3-D, representación y fotorrealismo. Algunas opciones disponibles incluyen modelación NURBS, sistemas de pintura, trazado de líneas, fonts de 2-D y 3-D.

Este sistema proporciona al usuario características como pop-up menus, ventanas definidas, vistas en tiempo real y despliegue de áreas segmentadas.

### *Modelación*

FACE es un modelo poligonal y es controlado por medio del mouse, teclado, digitizing pad o dial box. Es usado para construir extrusions y superficies; trasladar, rotar con transformaciones lineales, y transformaciones no lineales para inclinación, esferas y de enrollar. El modelo NURBS es una opción que implementa todas las características interactivas proporcionadas por los B-splines, que incluye edición de curvas, generación de superficies, elevación, extrusion, los cuales son complementadas por operaciones Booleanas sobre sólidos.

### *Animación*

ANIM es un módulo jerárquico patra controlar objetos, fuentes de luz, y la cámara virtual. Los frames principales pueden ser desarrollados interactivamente y usados para crear secuencias, que son manipuladas usando gráficas en 2-D relacionando parámetros con tiempo. Otra característica importante incluye rastreo de objetos con la cámara, una visualización animada del campo de la cámara de su punto de vista y punto de enfoque, y un lenguaje de alto nivel para efectos procedurales.

Las estructuras articuladas son muy importantes para desarrollar movimientos reales, y la herramienta ARTIC proporciona al animador un gran ambiente en el cual es usada la cinética inversa para conducir estructuras esqueléticas. ARTIC proporciona herramientas especiales para construir estructuras unidas e incorporar limitaciones cinéticas en la forma de límites de movimiento y resistencia.

### *Representación*

El módulo RENDER realiza escenas usando un algoritmo scan-line con los sombreados tradicionales de Gouraud y Phong, y también incorporar otras características útiles como texturas sólidas, sombras, mapeo de reflexiones, para manejo de acabado mate. TEX3D es un editor interactivo que permite al usuario producir objetos de textura real de cualquier forma usando texturas sólidas. Están disponibles dieciséis módulos de textura sólida en donde los parámetros controlan el color, forma del filtro, perturbación normal de color que puede ser controlada por el usuario. Además, están disponibles treinta y cinco librerías

de texturas como madera, piedra, mármol y agua, los cuales pueden ser usados para crear otra variedad de texturas.

El módulo opcional RAY incorpora todas las características del RENDER y usa trazado de líneas para desarrollar sombras exactas, reflejos, transparencia y refracción, mientras PARTICLE es otra opción para investigar los beneficios de sistemas de partículas para modelar los efectos de lluvia, nieve, gas, fuego, cataratas y explosiones. Estos sistemas pueden asignar los atributos de posición, color, velocidad, peso y animación usando dinámica, restricción de flujo dentro de un volumen.

# 9.2.3 SOFTIMAGE

SoftImage Inc. fue fundada en 1986 por Daniel Langlois, quien fue co-director de la película animada por computadora 'Tony de Peltrie'. Está localizada en Montreal, Canadá, pero tiene subsidiarias de ventas y oficinas de soporte en todo el mundo. El sistema SoftImage 3-D permite un amplio rango de aplicaciones incluyendo emisión y animación de películas, diseño de producción y diseño industrial, arquitectura y visualización científica. Es un sistema totalmente integrado en que todas las funciones están disponibles en un programa. El usuario puede modelar, animar, reproducir y representar en una actividad continua.

## *Modelación*

El módulo Model contiene todas las herramientas necesarias para generar curvas polinomiales, superficies y objetos. Esto tiene una representación límite en la forma de polígonos, partes de superficie, y pueden ser manejados por operaciones Booleanas y una red de deformación. Una librería paramétrica proporciona al usuario con un extenso rango de primicivas desde un círculo hasta un dodecaedro.

Las curvas 3-D pueden ser trazadas como objetos y pueden ser definidas con interpolación lineal, interpolación B-spline o con interpolación Cardinal. Las herramientas están disponibles para insertar, borrar y mover puntos, ajustar tensión, fusión de curvas polinomiales y ajustar niveles de continuidad en cada punto de las curvas Bézier.

Además hay otras características para extrusions, superficies de revolución, partes bi-cúbicas, oblicuas, elevaciones, cortes, redondez, y en cualquier etapa el usuario puede crear o borrar vértices, bordes y polígonos sobre modelos existentes, interactivamente.

## *Animación*

El módulo Motion proporciona un control total del movimiento del key-frame para todos los componentes de la escena incluyendo cualquier objeto o posición de grupos, rotación, escala y forma; los atributos tales como reflexión, transparencia y refracción, y para todas las otras entidades en la escena incluyendo atributos de luz, atributos de cámara, efectos atmosféricos.

Cuando las posiciones del key-frame son salvados, los cursos del movimiento se hacen disponibles para que el animador los pueda editar e incorporar en otra escéna. Los cursos del movimiento también pueden ser definidos como curvas poligonales y usados para controlar objetos, luces y la cámara. Las herramientas están disponibles para ajustar la duración y establecer tiempos para un modelo, cámara o fuente de luz a lo largo del curso del movimiento.

El módulo Actor permite una animación interactiva basada en cadenas articuladas cinética y dinámicamente controladas. Esto se puede combinar para formar esqueletos dentro de una envoltura que puede ser deformada interactivamente. Esto hace posible la creación de animaciones complejas combinando cadenas cinéticas y cadenas dinámicas, cuyo comportamiento depende de sus propiedades físicas (densidad, fricción, inercia) y sobre fuerzas físicas

como gravedad o viento, los cuales pueden ser animados por métodos convencionales.

### *Representación*

El módulo Matter proporciona herramientas para obtener material y propiedades de superficie para crear madera, vidrio y efectos de agua. Se pueden especificar colores usando RGB, HLS o HSV, y modelos de sombreado que incluyen Gouraud, Phong.

Los mapas de textura son accesados automáticamente con interpolación bilineal y que pueden ser asignados a un amplio rango de componentes de superficie como ambiente, difusión, specular, desplazamiento, transparencia y reflexión. Los mapas pueden ser aplicados usando proyecciones planares, cilíndricos, esféricos. El mapa de transparencias puede ser usado para modular lo opaco de una superficie, y permitir la creación de efectos de superficies complejas.

## 9.2.4 SYMBOLICS

El sistema de animación Symbolics consiste de un ambiente integrado por hardware y software, además, recientemente ha salido al mercado una nueva versión que corre en computadoras Apple Macintosh. Difiere de otros sistemas en que LISP es usado para implementar software orientado a objetos, lo que permite al usuario detener el sistema en cualquier momento, ingresar un procedimiento extra y continuar con el proceso.

El sistema es manejado por un computador central con monitor monocromo en donde los comandos son ingresados a través de una interface de manejo de menú usando un mouse. Una pantalla común se muestra en la Figura 9.1, la cual muestra ventanas que contienen la imagen animada, menús, comandos, y representación gráfica de los parámetros de animación. Un segundo monitor despliega las imágenes representadas totalmente a color, y también es usada para trabajos de pintado electrónico.

El software consiste de cinco productos básicos: S-Geometry, S-Dynamics, S-Render, S-Paint y S-Utilities.

### *Modelación*

El sistema S-Geometry sirve para esculpir un modelo, donde el usuario empieza por seleccionar una librería, o definiendo algún volumen primitivo. Esto es desarrollado interactivamente por el usuario seleccionando vértices, bordes o lados, que son manejados para moldear la forma deseada. El proceso en tiempo real de las computadoras centrales permite al usuario ver el proceso de construcción desde cualquier punto de vista, aumentar el tamaño en áreas que interesan más, añadir detalles, y luego volver al punto de vista original. Cuando un objeto necesita alguna característica en particular, puede ser resuelto con un procedimiento; esto puede ser ingresado directamente en la ventana de animación o ingresándolo separadamente en un buffer de edición. Como algunas escenas son visualmente complejas, existe la facilidad de remover líneas para clarificar alguna escena.

El software orientado a objetos proporciona un ambiente ideal para permitir modelos jerárquicos, lo que permite al animador obtener grupos de objetos interactivamente o partes de objetos, y realizar operaciones específicas con ellas.

### *Animación*

El S-Dynamics proporciona toda una gama de herramientas de animación para control de cámara, pruebas en tiempo real, desplazamiento en la animación, guía;

todas estas herramientas se encuentran activas a través de una interface común en la cual los atributos dinámicos están desplegados gráficamente. También se pueden declarar nuevos operadores dinámicos. La misma interface permite al animador seleccionar secuencias específicas para reproducirlas a diferentes velocidades, retroceder, adelantar.

### *Representación*

El producto S-Render se encarga de todas las interpretaciones internas. El S-Render utiliza mapeo de textura, mapeo de reflactancia, mapeo de opacity y mapeo de desplazamiento, los cuales pueden ser combinados independientemente con mapas de opacity para controlar efectos de transparencia. Las imágenes pueden ser representadas directamente en la pantalla del monitor, para almacenar un frame para cintas de video o película, o vía digital CCIR 601 o Ethernet a discos de almacenamiento.

Muy a menudo un animador necesita acceso a sistemas de pintura para retocar frames individuales; esto es una característica útil en el sistema Symbolics. El S-Paint realiza su trabajo y ofrece un sistema digital sofisticado de 32-bit con lápiz de presión sensitiva para crear texturas, reflejos, mapas de opacity, para retocar cualquier imagen interna y para diseñar arte en animación en 2-D. Además, el S-Paint puede ser usado para pintar una trayectoria en 3-D o codificar una figura directamente.

## 9.2.5 WAVEFRONT

En 1984 Bob Abel, y Larry Barels formaron Wavefront Technologies, Inc. Hasta 1987, el software Wavefront fue usado manualmente para aplicaciones en televisión, pero cuando sus formatos de archivo y especificaciones de interface fueron conocidos por el público, por los usuarios en la industria, a los usuarios militares, y a los usuarios en la educación y aplicaciones espaciales; todos ellos se convirtieron en compradores importantes. Actualmente el sistema ha evolucionado en The Visualizer Series, que consiste del Data Visualizer, The Personal Visualizer y The Advanced Visualizer; esto es para satisfacer la demanda de mercado en la Animación por Computadora en 3-D. El software Wavefront corre en IBM RISC System 6000, Hewlett-Packard, Digital Equipment Corporation y SUN.

Además la industria de la televisión ha sido el mayor usuario de las imágenes generadas por computadora; también más aplicaciones científicas requieren de herramientas más poderosas para transformar datos numéricos a películas animadas en 3-D. Una observación importante acerca de Wavefront es que las aplicaciones de simulación y media comparten muchas características para animar y representar objetos.

El Data Visualizer proporciona un ambiente interactivo para visualizar datos técnicos, que concluirían en los objetos en fluídos dinámicos computacionales, oceanografía, ingeniería estructural y geología. Para explorar estos datos, los cuales pueden ser sobre redes regulares, irregulares o sin estructura, el usuario tiene acceso para el corte de planos para despliegue de vistas seccionales.

El Personal Visualizer está dirigido a diseñadores de mecánica, arquitectos, ingenieros automotrices y diseño de productos. Puede importar descripciones geométricas de una variedad de sistemas CAD, y permitir al usuario editar los archivos y quizá crear vistas de objetos complejos. Esto puede ser iluminado por diferentes fuentes de luz

El Advanced Visualizer proporciona al usuario facilidades para animar, representar y modelar, así como también aceptar archivos del Data Visualizer y del Personal Visualizer. Corre sobre plataformas UNIX, y su arquitectura abierta le hace muy flexible para interactuar con otros sistemas.

## *Modelación*

La versión actual permite polígonos, superficies Bézier y Cardinales, construir geometrías sólidas, NURBS. El usuario puede esculpir objetos interactivamente con comandos para deformación de figuras tales como: torcer, inclinar, enrollar, flamear. Los objetos pueden ser importados de sistemas CAD por medio de un archivo con formato ASCII, IGES o a través de traductores directos para muchos sistemas como: Integraph, DXF, GDS, CADDS/4X, IDEAS, ANVIL, MOVIE.BYU y VDA-FS.

## *Animación*

El módulo de animación es llamado PreView y proporcina al animador algunas características generales tales como interpolacíon key-frame, movimiento de curvas, metamorfosis de objetos, reducción y extensión de volúmenes. También permite la animación de estructuras jerárquicas y proporciona un lenguaje para definir procedimientos de secuencias de animación. Un rango elegido de frames puede ser reproducido hacia adelante o hacia atrás en tiempo real o en tiempos continuos. Una de las características más importantes del PreView es que al menos están disponibles 100 canales para manejar cualquier atributo de una secuencia de animación. Estos datos pueden tener una fuente interna, o una fuente externa en un formato ASCII o binario. Esto permite a una animación ser creada usando datos numéricos derivados de un proceso de simulación dinámico para crear movimientos naturales. El PreView puede manejar directamente software de análisis dinámico tal como ADAMS.

## *Representación*

El módulo Image permite el sombreado de Gouraud y Phong, trazado de líneas, mapeo de reflexión, efectos de neblina, sombras, transparencia, refracción. A los objetos se les puede asignar atributos como: resplandor, transparencia, índice refractivo, color, características de textura y altura de la superficie usando mapas; mientras las luces pueden ser controladas por intensidad, color, atenuación, alcance, junto con efectos atmosféricos de niebla y luz ambiental.

Finalmente, el sistema ofrece un rango de funciones compuestas para crear escenas complejas con base en imágenes independientes; esto hace posible efectos como transparencia variable, ajuste, cambios de tamaño del cuadro.

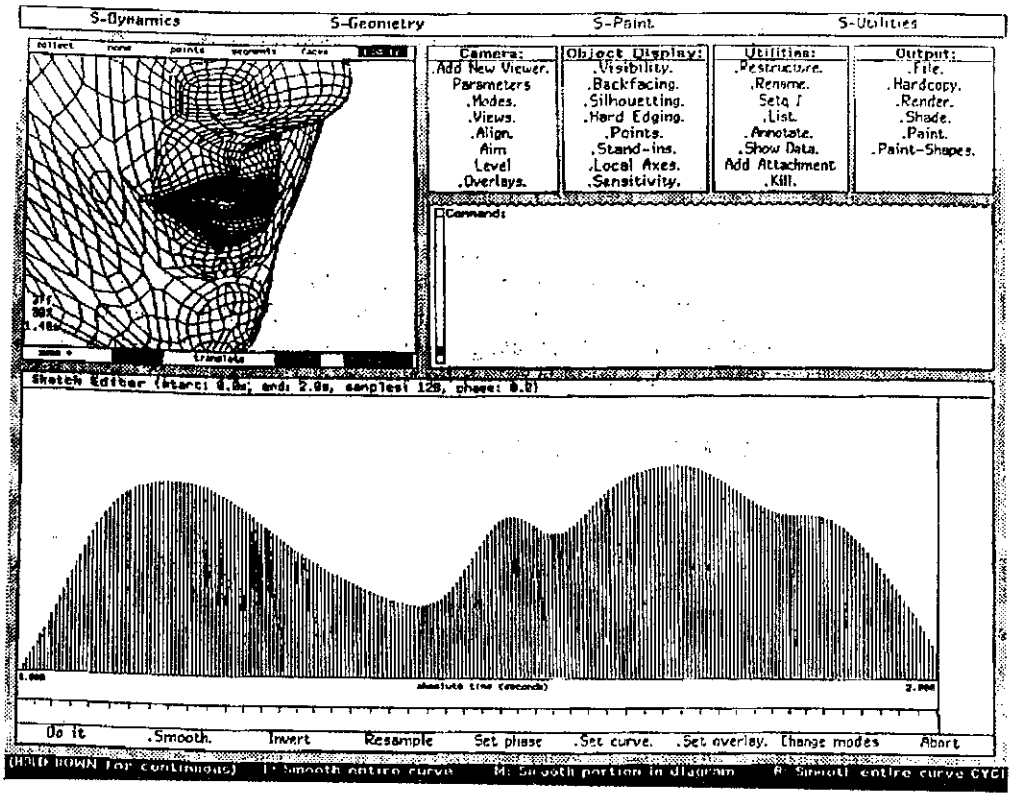



**Figura 9.1**: este diagrama muestra el área de ventanas y el menú interactivo del sistema Symbolics' S-Animation, también muestra el área de comandos.

# CAPITULO 10

# HARDWARE PARA LA ANIMACIÓN POR COMPUTADORA

Es solamente durante los últimos años que la Animación por Computadora ha proporcionado un costo efectivo alternativo para crear imágenes animadas; esto ha sido a causa del desarrollo de la tecnología de bajo-costo y computadoras rápidas, y avances complementarios en software. Las interfaces de usario como windows, íconos y menúes, el uso del teclado se ha reducido por la introducción del mouse (ratón).

## 10.1 COMPUTADORAS

Un adecuado sistema de Animación por Computadora necesita, sobre todo, una computadora con memoria adecuada, procesamiento de alta velocidad, gran capacidad de espacio en disco y varios periféricos de gráficos de entrada/salida. Algunos años atrás, los tiempos de rendimiento de 15 a 40 min/cuadro era común y esto significaba que una secuencia de animación de 30 segundos requería entre 200 y 400 horas de interpretación para la secuencia final. Hoy en día, con estaciones de trabajo con velocidades altas de multi-procesador, es común hacer representaciones en el orden de algunos minutos. El incremento de la ejecución del hardware se ha visto virtualmente compensada por las imágenes sofisticadas y complicadas requeridas por los clientes; además no se debe desestimar la ejecución de procesamiento necesario para la Animación por Computadora.

Aunque algunas computadoras son adecuadas para aplicaciones de gráficas, no se pueden considerar como alternativas válidas para incorporar a las estaciones de trabajo frame store integrales, los z-buffers y las operaciones de matriz. Obviamente mucho depende de la naturaleza de la aplicación, y como los proyectos de Animación por Comutadora explotan los beneficios de la simulación dinámica, radiosity, ray tracing, mapeo de textura y anti-aliasing, incrementará la demanda de hardware más rápido.

## 10.2 ALMACENAMIENTO EN DISCO

Como el tiempo de representación puede ser considerable, existe una tendencia para almacenar las imágenes en el disco, y así puedan ser utilizadas dentro de uno o dos segundos después. Además, existe la necesidad de almacenar imágenes 'en el caso de que sean necesarias después' -una vez descubierto que el espacio en disco, ya sea su capacidad en cientos, miles o megabytes, rápidamente desaparece si estas políticas son seguidas. Esto es cuando no se necesita una computadora para probar que una animación de 10 segundos, a una velocidad de 25 cuadro/seg., para una resolución de pantalla de 1 millón de pixels de totalidad de colores necesita 750 Mbytes de almacenamiento en disco.

## 10.3 CAPTURA DE IMAGENES

Aunque algunos medios de animación almacenan las imágenes en disco, a veces deben ser almacenadas en otro medio como una película o cinta de video. El video es el medio ideal para la televisión, comercial, corporación y aplicaciones educativas, la película es usada para manejar secuencias de efectos especiales en el cine.

El principal propósito de las películas es estar disponible cuando automáticamente se toma una foto digital y se despliega sobre una resolución lisa CRT, donde cada componente de los colores primarios de una imagen proyectada

individualmente y fotografiada a través de un filtro de colores primarios antes de que la película continúe al siguiente cuadro. Estos registros pueden dejarse correr automáticamente bajo el control del computador central, y poder trabajar en películas de 16 mm, 35 mm., y aún registrar diferentes formatos de películas.

Los avances en la tecnología del video tienen capacidad de registrar simples cuadros sobre una cinta de video. El formato VHS ha sido usado, aunque existe una tendencia de eliminar algunos cuadros, lo cual puede ser aceptable durante el desarrollo de una secuencia de animación pero no podrá ser tolerada bajo otras circunstancias. La filmación del video profesional utiliza cintas formateadas con códigos de tiempo que pueden hacer un recorrido a una posición deseada en la cinta y registrar la imagen deseada. Una desventaja es que cuando se desarrolla la secuencia de registro, la operación del recorrido resulta tardada y las cabezas de grabación resultan desgastadas.

Un desarrollo más reciente es el uso de discos ópticos reemplazables que se les puede escribir una vez y regrabados varias veces más. Son ideales para desarrollar una secuencia de animación total, y tienen una capacidad de aproximadamente 30 minutos de animación -y resultan en un formato conveniente para archivar proyectos.

## 10.4 POST-PRODUCCIÓN

Es muy difícil encontrar una secuencia de Animación por Computadoras producida sin estar sujeta a alguna post-producción; esto solamente consiste de una simple editación, y podría involucrar cuadros individuales que van a ser modificados en un sistema de pintura y digitalmente arreglada con otras imagenes. En esta técnica, es muy difícil distinguir entre elementos naturales y sintéticos en muchas de las televisiones gráficas de la actualidad.

Tal vez el sistema de editación de Harry Quantel ha tenido más influencia sobre el arte de este tipo más que cualquier otro sistema. La naturaleza digital de sus funciones de procesamiento de imágenes significa que muchas de las imágenes pueden ser realizadas sin deterioro en la calidad de la imagen final. Esto ha transformado la forma en que las secuencias de Animación por Computadora son hechas, haciendo posible incorporar el video, fotografía y sombreado, con elementos de Animación por Computadora.

El estilo de la Animación por Computadora en la televisión o publicidad han sido influenciados por los procesos de post-producción. Los diseñadores de gráficas que usan Harry pueden diseñar historias que incorporen vida, y modelar tomas que parezcan reales, por ejemplo. Las teteras animadas por computadora, botellas de leche danzantes, cajas de cereales, cajas de detergente pueden ser representados claramente, que cuando son creadas con acción, es imposible distinguir entre elementos reales y sintéticos.

Además, las cámaras controladas por computadora pueden ser programadas para recorrer algún modelo a escala. Al final de la toma, los datos coordinados de la curva en el espacio 3-D puedan ser ingresados al sistema de Animación por Computadora. Cuando las imágenes animadas por computadora son creadas con aquellas obtenidas de las tomas, una perspectiva común garantiza que la imagen compuesta es impecable.

# CAPITULO 11

# APLICACIONES DE LA ANIMACIÓN POR COMPUTADORA

## INTRODUCCIÓN

En comparación con la animación tradicional, la Animación por Computadora es relativamente nueva, y en el corto período de su desarrollo ha demostrado un potencial ilimitado en un amplio número de aplicaciones. Además, no existe otra forma de crear dibujos animados más divertidos; es una forma revolucionaria para simular y visualizar un mundo animado en 3-D. La habilidad de construir mundos imaginarios dentro de la memoria de la computadora parece fantasía o increíble, y además, crear e interactuar con estos ambientes virtuales está basado en conceptos que son realmente algo simples.

Una de las razones del éxito de la Animación por Computadora en diferentes áreas es que se puede usar el mismo hardware y software sólo con hacerles algunas pequeñas modificaciones. Logotipos de empresas, motores de autos, moléculas, muebles y edificios pueden ser representados con las herramientas de modelos de polígonos, superficies recortadas y geometría sólida constructiva, lo cual puede ser animado y representado por diferentes técnicas que se han indicado anteriormente. Para demostrar esta increíble flexibilidad de aplicaciones, se verá ahora cómo la Animación por Computadora es aplicada en diferentes disciplinas.

## 11.1 TELEVISIÓN

Uno de los usos del diseño gráfico por televisión es crear gráficas animadas para programas, noticieros, publicidad. Tradicionalmente, la acción en vivo, modelos a escala, generador de caracteres, cell animation y piezas de tarjetas que podrían ser usados para producir estas secuencias, aunque actualmente los sistemas de coloreado por computadora, sistemas compuestos digitales, control de movimiento y Animación por Computadora en 3-D son herramientas de diseño conocidos.

La tecnología digital ha cambiado la producción en televisión: la naturaleza de las imágenes de video les permite ser almacenadas en la computadora para poderlas manipular después. Estas se pueden ver en perspectiva, iluminadas, incorporadas con otros objetos sintéticos en 3-D. Esta integración de gráficas generadas por computadora con efectos de video digital proporciona a los diseñadores un medio creativo que parece no tener límites.

Los noticiarios tienen un uso particular en la Animación por Computadora al utilizar pseudo animaciones en 3-D creados sobre sistemas electrónicos de coloreado. Una noticia de última hora puede rápidamente utilizar una base de datos de mapas que puede ser arreglada con otros símbolos e imágenes genéricos para crear animaciones sofisticadas, para que transmitan en unos cuantos minutos después de haber recibido la noticia original. Aunque esto hace más emocionante la televisión informativa, los diseñadores gráficos trabajan rápidamente para estar más cerca del modo de operación en tiempo real. Láminas 6 y 7 son parte de una secuencia de animación usada por la BBC en el noticiero matutino.(ver apéndice).

La Animación por Computadora es un medio muy flexible para crear secuencias de títulos para programas, y no existen restricciones para el diseñador gráfico sin importar el medio que utiliza. La secuencia podría hacerse completamente con una computadora. También la acción en vivo puede ser ingresada en una secuencia de animación en forma de mapas de textura que pueden ser mapeadas sobre las superficies en movimiento.

Como se ha mencionado anteriormente, el sistema de control de la cámara en movimiento puede proporcionar el trayecto de un vuelo en 3-D que se puede ingresar en un sistema de animación, entonces, con el uso de una editación digital adaptable, las dos secuencias de imágenes pueden ser compuestas para proporcionar una unión perfecta.

Las secuencias animadas por computadora juegan una parte muy importante en la programación de gráficas para ilustrar un proceso complejo en 3-D, una relación matemática, los componentes de un montaje, o la visualización de datos estadísticos. Más que la habilidad de la animación por computadora, dichas secuencias podrían ser animadas manualmente o creadas usando modelos.

# 11.2 INDUSTRIA

El diseño con la ayuda de la computación siempre ha sido identificado como una aplicación para las gráficas por computadora, y con la llegada de la animación costo-efectividad, los diseñadores son capaces de animar sus modelos virtuales y descubrir características que pueden ser conocidas hasta después del proceso de manufacturación. Por ejemplo, un componente como la ventana de la puerta de un auto puede ser animada para ver si interfiere físicamente con otros elementos. Un asiento de auto puede ser colocado dentro del interior del mismo para descubrir si un rango de conductores son capaces de alcanzar los pedales de control y el timón. Los vehículos pueden ser modelados y simulados para descubrir sus características de suspensión; los cálculos del centro de gravedad pueden ser estudiados para investigar la estabilidad bajo diferentes condiciones de carga.

En realidad, dado cualquier modelo en 3-D de un montaje, los componentes individuales pueden ser animados para descubrir su relación física con otros componentes.

Aunque esto no significa que cualquier sistema CAD es capaz de realizar este trabajo. Los sistemas convencionales CAD proporcionan las herramientas de diseño necesarias para desarrollar y visualizar objetos virtuales, pero si ya están animados es más fácil especificar los ejes de rotación, superficies de contacto, mecanismos de deslizamiento y enlace. Las aplicaciones industriales para la Animación por Computadora son variadas. Por el momento, la animación stop-frame es aún usada para preparar discos de video; un ejemplo podría ser el mostrar a los trabajadores de una plataforma de petróleo las rutas de salidas de emergencia. Esto involucra un equipo de cámara y fotografías exactas de cada posición a lo largo de la ruta. Desafortunadamente, el proceso total toma dos o tres días, sobre los cuales los niveles de luz cambian, sin mencionar las condiciones climáticas. Una alternativa es tomar una base de datos CAD para la plataforma y remover los detalles innecesarios. Las rutas de salida pueden ahora ser seleccionadas y una secuencia de animación preparada para 'hacer volar' la cámara virtual a lo largo de diferentes caminos de salida. De hecho, la naturaleza digital de las imágenes simplifica todo el proceso de preparar un video final en disco.

# 11.3 SIMULACIÓN DE VUELO

La Animación por Computadora juega un papel importante en la simulación de vuelo donde generadores de imágenes en tiempo real son usados para proporcionar mapas de texturas realistas de aeropuertos y los terrenos circundantes. La cabina del piloto, que es una réplica de un modelo de tamaño natural, efectivamente resulta de la cámara virtual de la computadora. Los controles de vuelo del piloto alimentan señales digitales directas a un programa de simulación con las características de vuelo del avión. Esto sirve para predecir dónde se encontrará el avión en un tiempo de milisegundos, y especifica la posición en el espacio con desvíos, ángulos de rotación e inclinación. Con estos datos, el generador de imágenes representa una escena mapeada en aproximadamente 0.02 segundos. En realidad, son creadas tres imágenes para proporcionar un campo de 180° de visión.

Como el piloto archiva el simulador, una vista en 3-D de tiempo real de un

ambiente virtual es generado automáticamente. Las imágenes contienen muchas de las características mencionadas en capítulos anteriores. Por ejemplo, otros aviones pueden despegar o aterrizar, lo que es logrado moviendo dinámicamente el avión a lo largo del espacio de curvas pre-calculado. También pueden ser animados los buses y taxis del aeropuerto, personas moviéndose detrás de las ventanas del edificio de la terminal, las luces de los autos a lo largo de la autopista.

Las condiciones del tiempo son simuladas usando una variedad de técnicas. Por ejemplo, la niebla, que es muy importante en el entrenamiento del piloto. Similarmente, cuando se vuela entre las nubes, la dispersión de las luces de las alas también pueden ser simuladas. La nieve es simulada seleccionando un diferente conjunto de mapas de textura. La caída de la nieve puede ser simulada por sistemas de partículas que son desplegadas usando las características caligráficas de los proyectores de despliegue; esto hace que el piloto pareciera que está volando en nieve verdadera.

El nivel de las nubes puede ser colocado a cualquier altura, y cuando el avión se mueve a través de ellas, el detalle de la tierra es eliminada hasta que un nuevo mapa de textura revele un lugar fuera de la capa de nubes. En condiciones de tormentas, las luces intermitentes son reflejadas usando puntos de luces con un incremento momentáneo y simultáneo en imágenes brillantes.

Las luces de aterrizaje iluminan el camino usando las características de los focos como fuentes de luz, pero como un aeropuerto tiene tantas fuentes de luz, es imposible calcular sus efectos de iluminación en tiempo real. Consecuentemente, son simulados con polígonos luminosos que parecen ser resplandencientes, y mapas de textura luminosa formados por trazos transparentes.

La base de datos del aeropuerto generalmente requiere de seis hombre-mes para completarse, como el aeropuerto puede estar localizado cerca de la ciudad cuyos edificios altos, monumentos, puentes, ríos tienen que ser modelados. En vista de que el rango sobre dichos modelos son vistos a diferentes distancias por el piloto, éstos deben ser modelados tres o cuatro veces para diferentes niveles de detalle.

## 11.4 ARQUITECTURA

Los arquitectos han usado gráficas por computadora en 2-D como una ayuda para la distribución de los planos del suelo y la organización de los servicios de gas, electricidad, agua, aire, sistemas de drenaje. Actualmente, los sistemas 3-D son importantes en la visualización de interiores y exteriores.

Aunque la arquitectura parece ser una aplicación obvia para la Animación por Computadora, se debe apreciar que la base de datos de un gran edificio puede almacenar cientos de elementos, cada uno con una geometría detallada. Además, el sombreado de Gouraud y Phong no es un modelo de sombreado real para visualizar interiores iluminados por paneles de luz difusa y luz del día, y los exteriores donde las condiciones del día, la hora, reflejos y sombras son importantes para el cliente.

De cualquier modo, el trazo del rayo de luz proporciona un método excelente para identificar sombras, reflejos y luz specular en paredes de vidrio de edificios, y se ha probado que la radiocidad es un método efectivo para revelar los complejos niveles de luz creados por múltiples reflejos difusos. Desafortunadamente; ambas técnicas son costosas en tiempo de procesamiento.

La radiocidad depende de un gran número de factores que relacionan la interacción de la luz difusa entre partes pequeñas de dos superficies; una vez que esto ha sido calculado, se crean imágenes reales. De hecho, algunos sistemas son capaces de generarlas en tiempo real. Si el costo de este desarrollo puede ser mantenido, tendrán un gran impacto sobre la arquitectura. La lámina 8 ilustra el nivel de realismo que es posible al usar las herramientas y modelos adecuados.

A pesar de algunos problemas, la Animación por Computadora es usada para tener una mejor visualización sobre la arquitectura y construcción de proyectos, donde es difícil imaginar como las construcciones nuevas son afectadas por el medio ambiente. Secuencias aéreas pueden proporcionar una visión más detallada de estos proyectos, y aclarar áreas de confusión creadas por los planos de

elevación sobre un diseño.

Muchos proyectos a gran escala utilizan actualmente estas técnicas y con la tendencia hacia las computadoras de bajo costo y alta velocidad; la Animación por Computadora será más utilizada en el proceso de diseño en la arquitectura.

# 11.5 PUBLICIDAD

La publicidad es una aplicación popular de la Animación por Computadora porque los proyectos proporcinan a menudo a los animadores la oportunidad de explorar y desarrollar nuevas técnicas. Se pueden modelar cocinas, saxofones, pianos, cajas de detergente, baños, teteras, cucharas, dulces, biscochos. Estos elementos son animados con FFDs, inbetweened, movidos a lo largo de curvas B-spline, compuestas con acción en video hasta que el cliente está convencido que el mensaje deseado será comunicado al público. Si bien la secuencia final puede durar unos segundos, para llegar a esta etapa, se necesitan algunas semanas de trabajo.

El modelar es una actividad que consume tiempo, especialmente cuando los objetos tales como autos, motores, ranas y paisajes tienen que ser creados. Los modelos tienen que ser construidos con el conocimiento de cómo ellos serán animados, y los cambios de último momento pueden tener dramáticas repercusiones sobre su construcción.

La animación de líneas de prueba, balanceo de colores, selección de texturas y ajuste de niveles de iluminación requieren un matiz fino, a menudo se necesita que la secuencia sea representada más de una vez, hasta que llega el plazo; en este punto, el video o película está hecho.

Al igual que las gráficas por televisión, la publicidad necesita una integración de la Animación por Computadora que a menudo es incorporada con imágenes obtenidas de la acción en vivo. Un balance de escalas y perspectivas entre los dos sistemas es muy importante; que, como los modelos por computadora son dimensionales, pueden ser interpretados en cualquier nivel de escala al ajustar ciertos parámetros.

El nivel de realismo logrado en estas secuencias es muy alto, y algunas veces el público no se da cuenta que se han usado computadoras. Tal vez de lo que sí se dan cuenta es que encuentran difícil entender cómo se producen los efectos. La lámina 9 demuestra el nivel extraordinario de creatividad en el diseño.(ver apéndice).

# 11.6 EFECTOS ESPECIALES EN PELÍCULAS

Películas como 'Tron', 'El Abismo', 'Vuelo del Intruso', 'Terminator 2', 'Lawnmower Man' y 'Total Recall', todas han demostrado que existe un lugar para la Animación por Computadora en la realización de efectos especiales. Aunque, los modelos a escala y sistemas de control de movimientos han probado ser un acercamiento al costo efectivo en la simulación de escenas de extraterrestres, la Animación por Computadora ha sido exitosa para modelar fuego, humo, explosiones, naves espaciales, humanoides y cabezas humanas hechas con agua.

Igualar la escala y perspectiva entre imágenes sintéticas y reales es vital si el efecto es convincente, y los equipos de efectos especiales tratan de asegurarse que la composición óptica o digital no delate los diferentes orígenes de las imágenes.

Aun cuando una película solamente contiene cinco minutos de Animación por Computadora, dichos minutos incluyen efectos que no podrían ser creados usando un proceso de bajo costo. Cinco minutos a 25 cuadros/seg. requieren 7500 imágenes, y si esto toma, en promedio, 15 minutos de representación, un total de 1875 horas son necesarias para su creación. Pero esto es solamente para una representación de animación; es mucho más cuando el director no está satisfecho con alguna escena. Consecuentemente, es necesario un gran equipo de personas y computadoras para trabajar en dichos proyectos.

# 11.7 DIBUJOS ANIMADOS EN 3-D

Los dibujos animados en 3-D han sido un vehículo conveniente para demostrar cómo puede ser una Animación por Computadora de flexible. 'Luxo Junior' (1986) de John Lasseter es una muestra de la evolución de la Animación por Computadora, y títulos subsecuentes 'Red's Dreams' (1987), 'Tin Toy' (1988) y 'Knickknack' (1989) han sido una influencia en el desarrollo de la industria de la Animación por Computadora. De cualquier modo, el potencial de la Animación por Computadora de los dibujos animados en 3-D fue apreciado antes en el desarrollo de este tema -proyectos como 'The Works' (1980) del NYIT y 'Tony de Peltrie' (1985) fueron puntos claros de cómo la Animación por Computadora se ha desarrollado. Con los continuos avances en hardware y software, se pueden explorar áreas más complejas.

En años recientes, otros proyectos como 'Locomotion' (1990) de PDI, 'Eric, the Dynamic Worm' (1989) de Gavin Miller, 'Don Quixote' (1990) de Videosystems y 'Eurythmy' (1989) de Susan Amkraut y Michel Girard, ha llevado a la Animación por Computadora a nuevos niveles de realismo, particularmente en el área de movimiento. Desafortunadamente, estas secuencias son cortas y son el resultado de investigaciones considerables y dedicación para resolver problemas complejos. De cualquier modo, conforme se vayan resolviendo los problemas, se estará más cerca al día en que los dibujos animados en 3-D sean producidos usando solamente Animación por Computadora.

Recientemente, se han logrado grandes avances en muchas áreas, desde el modelo y movimiento de elefantes, hasta el modelo de galaxias y agujeros negros. Actualmente, parece que virtualmente cualquier cosa es posible.

Aunque es inevitable que la Animación por Computadora establecerá un lugar para una variedad de estilos de gráficas que han empezado a aparecer, la industria tradicional de dibujos animados no dejará de trabajar para ir perfeccionando un estilo.

# 11.8 VISUALIZACIÓN CIENTÍFICA

Unos veinticinco años atrás la plotter de 12 pulgadas fue el periférico más deseado en la Animación por Computadora, normalmente manejado en línea por un computador central con 24 Kbytes de memoria. Era usado normalmente para desplegar gráficas relacionando cantidades asociadas con conjuntos de datos experimentales; esto representó la base de la visualización científica.

En los siguientes años, surgió la computadora personal, minicomputadoras, sistemas de cómputo masivos y la super computadora. Pero la gente todavía dibujaba las gráficas, ya que los avances en hardware y software no permitían el rápido despliegue de superficies a color usando proyecciones y perspectivas isométricas. Estas técnicas son usadas para interpretar un amplio conjunto de datos acumulados de un esfuerzo científico en geología, astronomía, oceanografía, física atómica y simulación. Si bien algunos de estos conjuntos de datos pueden ser representados por isosuperficies estáticas (superficies que representan un valor escalar), muchos no son solamente tri-dimensionales en el sentido espacial, pero incluyen atributos de presión, temperatura, velocidad, dirección, tensión, compresión, densidad, y composición material. Así, el problema fundamental en la visualización científica actual es la interpretación gráfica de datos multi-dimensionales.

Extraer varias dimensiones en las dos dimensiones físicas de una pantalla de despliegue, no es tan difícil; de hecho es usado cada día en las cadenas de televisión en el pronóstico del tiempo donde un mapa de 2-D muestra la distribución espacial del tiempo. El color se puede usar para indicar temperaturas locales; las flechas pueden ser usadas para mostrar la dirección y fuerza del viento, y nubes diagramáticamente transparentes pueden reflejar el nivel de nubosidad.

En similares técnicas de visualización científica, son usadas para trasladar datos multi-dimensionales en modelos visuales donde los atributos

físicos son traducidos en color, posición, sombreado, transparencia y efectos de brillo, juntos con la ayuda geométrica de isosuperficies, vectores, sistemas de partículas, mezcla de estructuras, y objetos poligonales.

Los sistemas de Animación por Computadora son usados actualmente por la comunidad científica para proporcionar visualizaciones animadas de conjuntos de datos dependientes del tiempo. Por ejemplo, en el análisis finito de elementos (FEA, por sus siglas en inglés), donde los objetos son representados dentro de una computadora como una mezcla espacial unida de algunos cientos de nodos, la representación interna puede ser procesada matemáticamente para calcular la tensión, esfuerzo y desviación, que es resultado de una fuerza o torque imaginario. Cuando dichos datos son visualizados, la estructura simulada puede ser vista muy flexible, y tal vez vibrante, y la técnica de pseudo-color usada para revelar los puntos bajo la tensión. La Animación por Computadora permite al modelo ser examinado desde cualquier punto de vista o trayectoria, y reprentar con la superficie detalles y sombras, incorporados con otros modelos, y aun compuestos con imágnes del mundo real.

Otro aspecto de la simulación es que el fluido dinámico computacional (CDF, por sus siglas en inglés), que es usado para calcular efectos de flujo del viento o flujo de fluido sobre la superficie de un objeto. Este es una importancia particular para las industrias automovilística, aeronáutica y espacial. Los túneles de viento son el medio convencional para descubrir el flujo del aire y factores de arrastre para un objeto, aunque con el poder de las computadoras, dichas simulaciones pueden ser hechas usando un modelo por computadora.

Una de las últimas aplicaciones de la Animación por Computadora en la visualización científica es en la interpretación de procesos de simulación dinámica. La simulación dinámica es usada para aumentar el realismo de estructuras articuladas; en la industria, dichas técnicas son importantes en el diseño de actividades. La estabilidad en robots, autos, camiones, es vital para su uso seguro bajo varias condiciones dinámicas de operación. Una vez más estos sistemas de enlace, amortiguación, masa, elasticidad, centros de gravedad y movimientos de inercia pueden ser representados matemáticamente en una computadora. Estos sistemas pueden entonces estar sujetos a fuerzas y torques y, con las herramientas de dinámica inversa, su movimiento puede ser exactamente adivinado. Los valores numéricos que representan estos movimientos pueden fácilmente ser importados a un sistema de animación.

## 11.9 EL FUTURO DE LA ANIMACIÓN

Qué pasará con la Animación de Computadoras en el futuro? Se ha adelantado en una era en que es imposible distinguir entre realidad y realidad virtual. Esto ya es posible en la actualidad. Actualmente se tienen sistemas de representación en tiempo real, capaces de generar imágenes reales en poco tiempo. La tecnología de despliegue ha progresado, ya que los sistemas de despliegue están disponibles y usados para substituir imágenes generadas por computadora. Los sistemas de realidad virtual nos permiten interactuar con nuestro medio ambiente de animación en muchas formas. Los modelos pueden ser observados con la ayuda de guantes interactivos, y un maniquí puede ser animado con datos tomados de juegos interactivos.

En la actualidad, se desarrollan redes neurales, sistemas basados en el conocimiento y sistemas de aprendizaje que tienen una influencia en la Animación por Computadora. Porqué debería una colección de polígonos estar dados por algún nivel de inteligencia, así que en lugar de ser dirigidos por el animador para cada cuadro, aprenden por medio de ensayar una secuencia hasta que el animador esté satisfecho? Podríamos ser capaces para dirigir nuestros modelos futuros por medio de la voz? El lenguaje es una forma natural de comunicación con otro ser humano, a su vez parece una forma de comunicarse con nuestros complementos sintéticos; esto es posible en la actualidad.

Mientras esta tecnología va hacia adelante, las nuevas herramientas para modelar que serán desarrolladas, involucrarán digitalización automática de una escena real. ¿Por qué construir superficies usando mapas cuando la geometría 3-D

puede ser extraída de imágenes de satélite? Tal vez los sistemas futuros no requerirán que cada superficie sea descrita; probablemente podrán ser creadas en tiempo real por un diseñador que no solamente pueda ver el objeto, sino también sentirlo.

Las anotaciones anteriores no son predicciones; muchas de ellas ya existen de alguna forma. El futuro real de la Animación por Computadora podrá ser más fantástico.

Existe una tentación para aceptar que lo que existe en la actualidad permanecerá para las siguientes décadas y lo que queda será descubierto. Pero ésta no es la actitud que se debe tomar. No podemos permitir que las limitaciones de tecnología y sistemas limiten nuestro pensamiento. Actualmente, el trazado de rayos de luz es usado regularmente, y algunos sistemas pueden producir imágenes de baja resolución en tiempo real.

No hay duda que nos estamos adelantando a los sistemas de bajo costo y tiempo real, los cuales se encargarán en el hardware de la mayoría de las representaciones realizadas actualmente por el software. Esto evitará el cuello de botella asociado con la manipulación y despliegue de pixels. De hecho, además en las áreas en las que actualmente el animador espera que el computador represente la imágenes, muy pronto serán determinadas por la experiencia del animador, lo que significa más esfuerzo en el diseño de herramientas para modelar objetos, y técnicas nuevas para combinar definiciones del usuario sobre animación creativa con aquellas proporcionadas por la animación dinámica.

Actualmente han empezado las investigaciones en todas estas áreas, por lo que el futuro parece esperanzador, ya que se dejarán de construir objetos a base de polígonos y funciones matemáticas. Por el momento, el futuro no ha llegado, y la razón porque se usa actualmente polígonos, y que se tarda algunos minutos por imagen, y se usan funciones matemáticas, es porque no tenemos elección, y además, puede ser que nos guste todo esto para que se nos pueda hacer más soportable.

# APENDICE

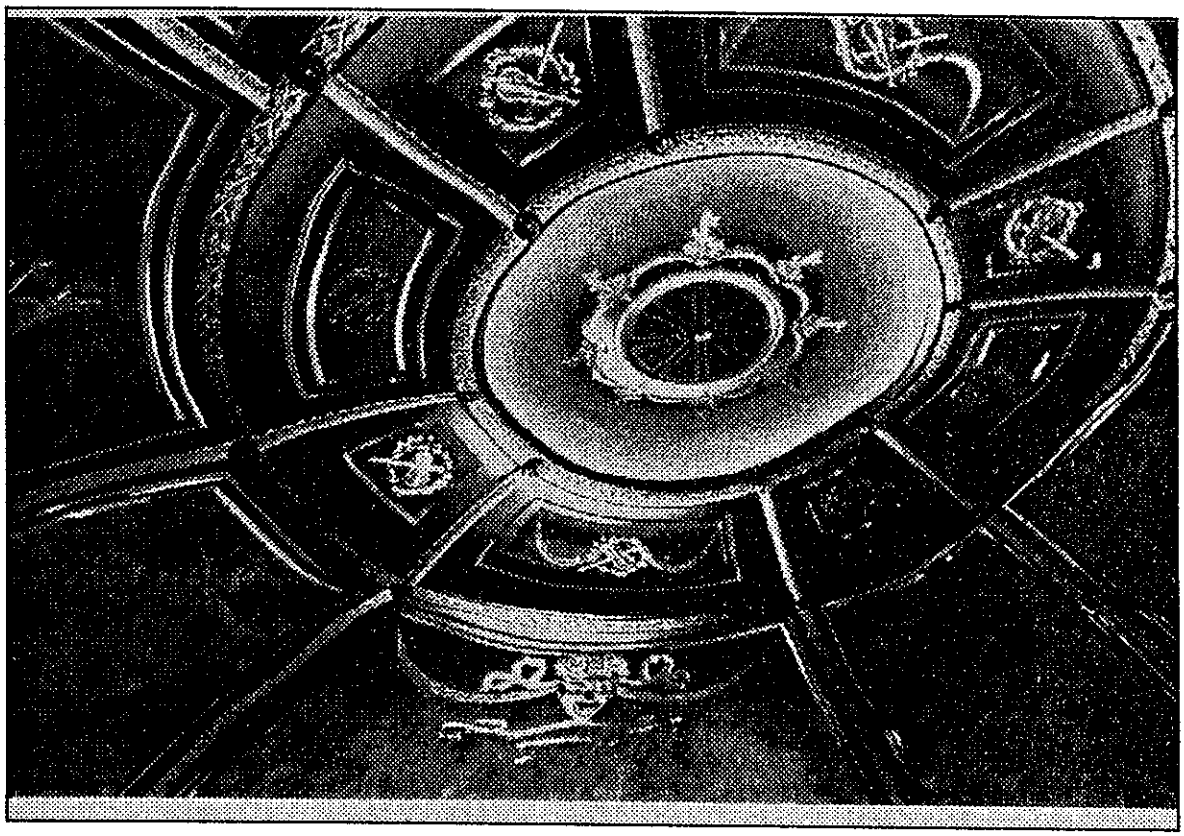


**Lámina 1**: esta imagen es de la secuencia de animación "Enter the Elgin". Este proyecto involucra modelos muy complejos una amplia gama de textura, abultamiento, desplazamiento, opacidad y mapeo specular. La luz requiere de una especial atención para crear un ambiente misterioso.

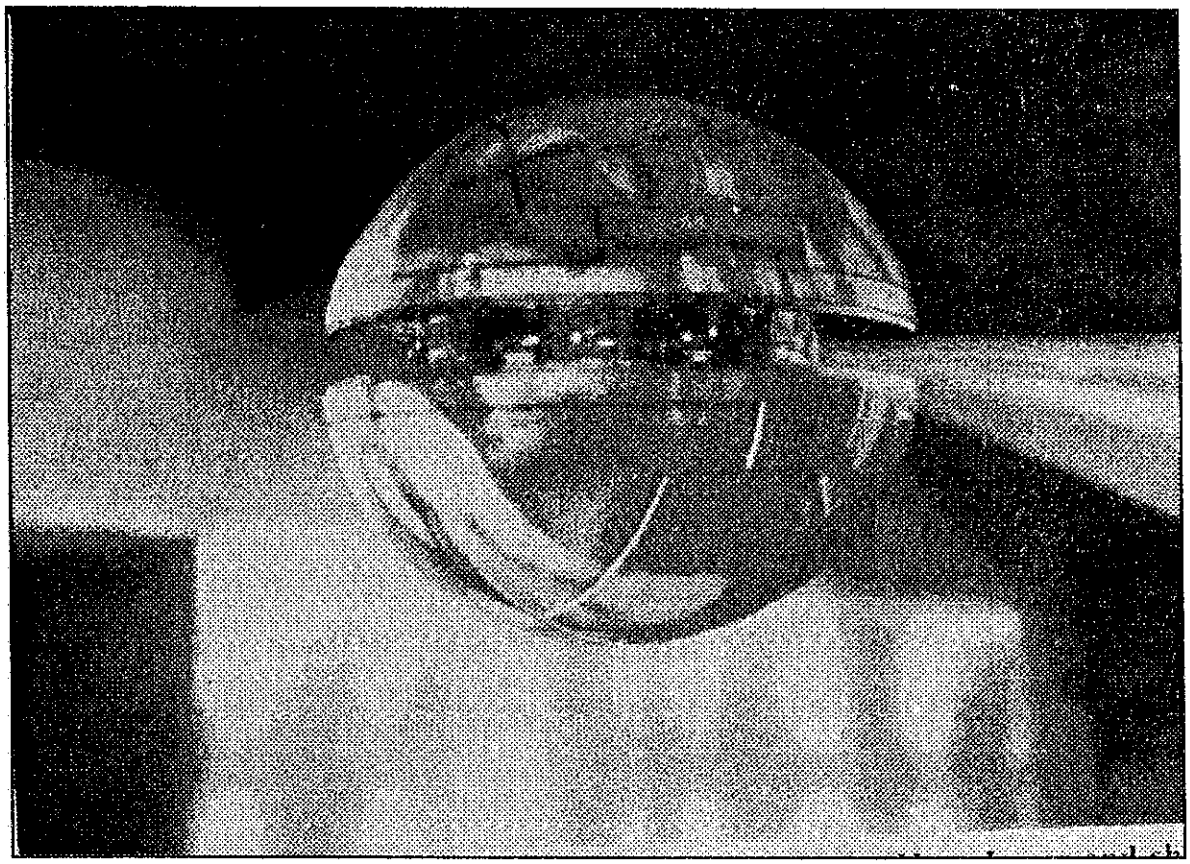


**Lámina 2**: esta imagen es tomada de un anuncio de publicidad, y muestra un modelo de computadora compuesta de movimiento real utilizando el sistema de edición digital Quantel's Harry. El modelo fue creado usando Alias y utilizando un mapeo del medio ambiente para formar reflejos reales sobre la superficie.

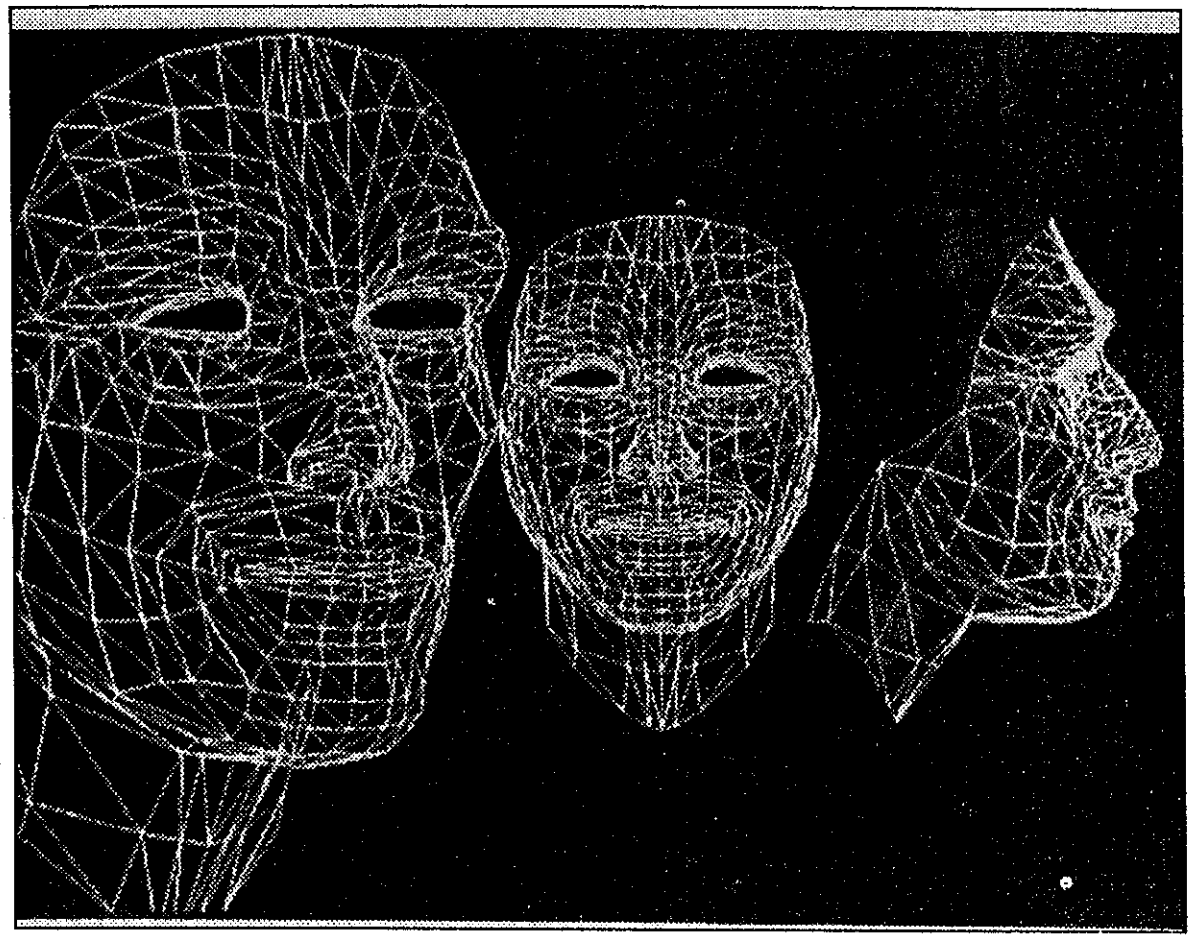


**Lámina 3**: esta es una imagen obtenida de las investigaciones de Keith Water sobre expresión facial. Las líneas dibujadas ilustran la mezcla de polígonos necesarios para soportar los detalles de la cara, para así tener un mayor acercamiento para crear expresiones humanas reales.

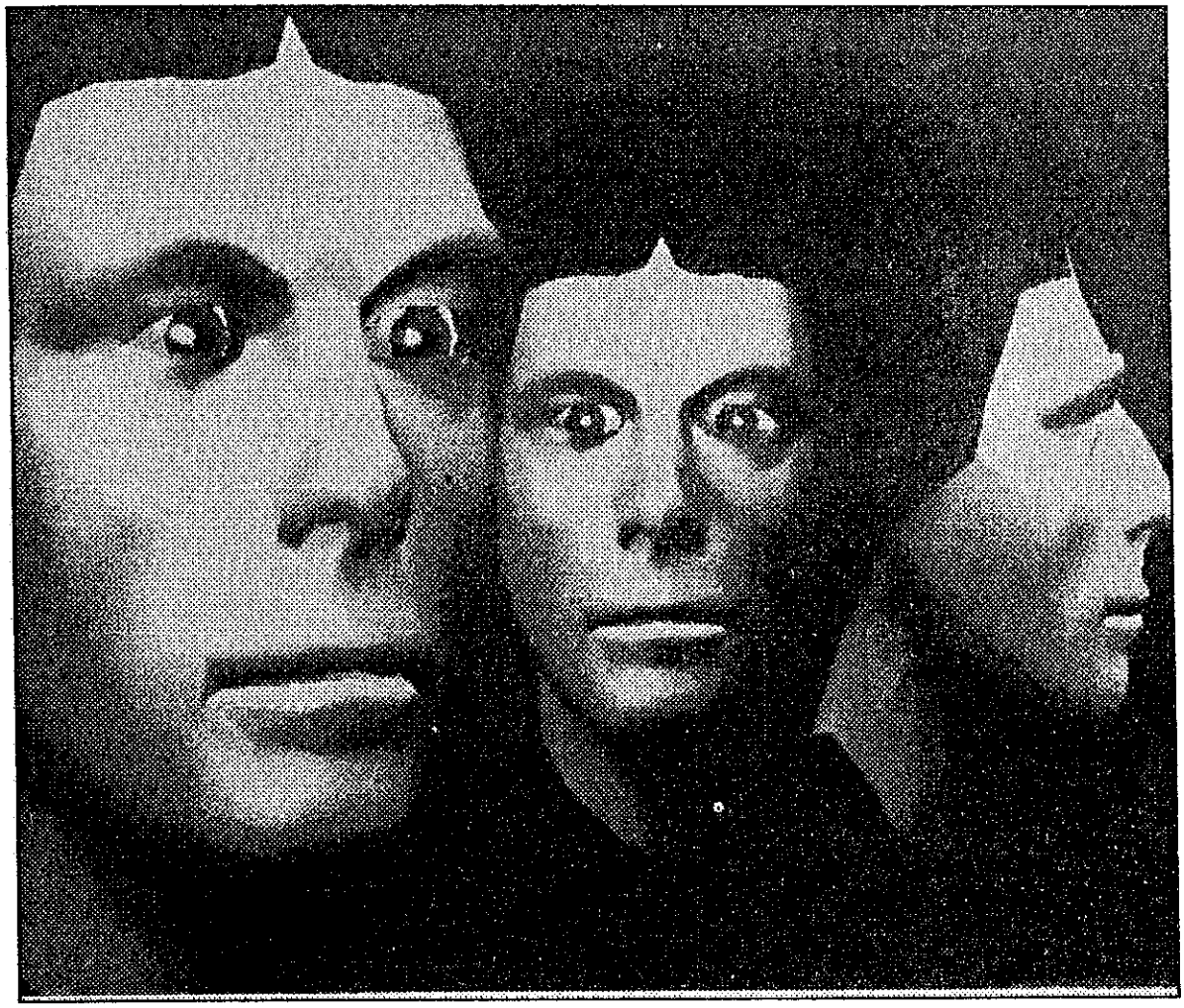


**Lámina 4**: esta gráfica muestra la parte final del modelo mostrado en la lámina anterior.

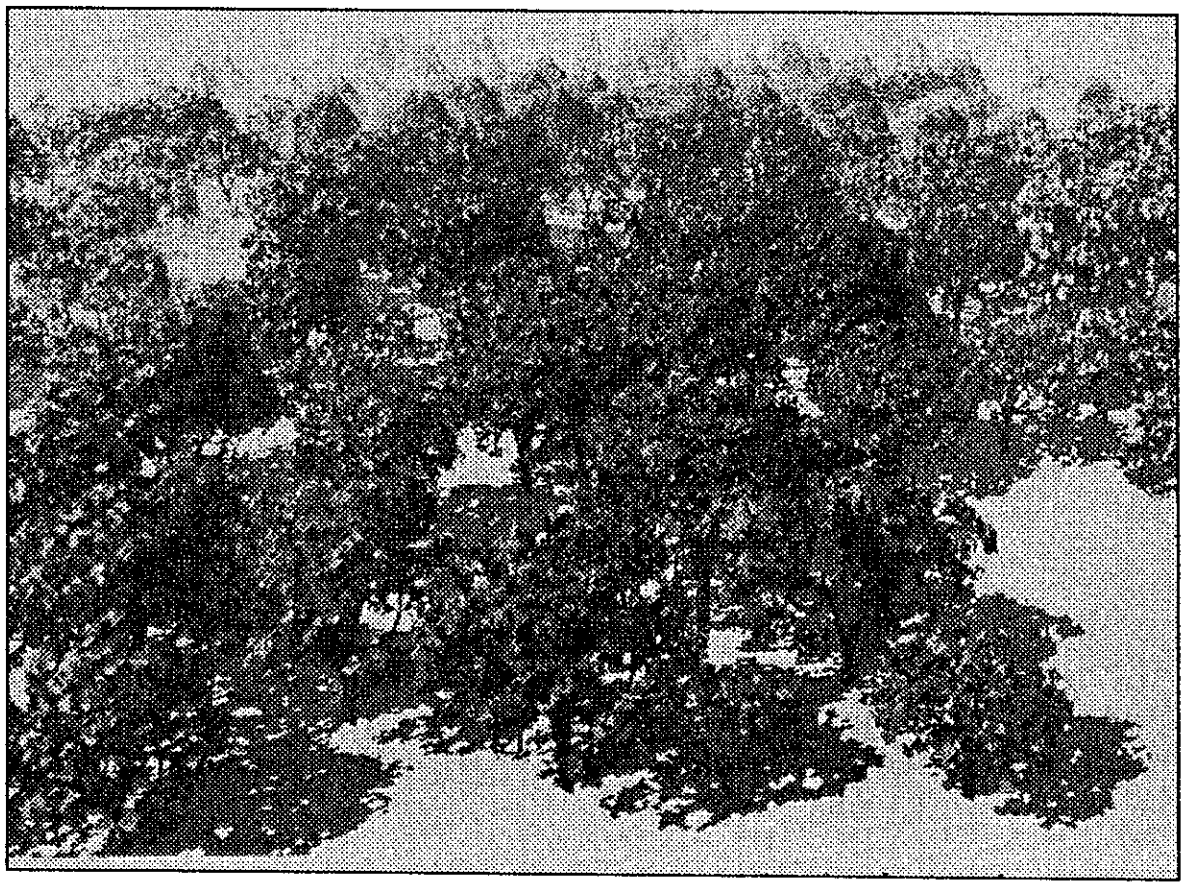


**Lámina 5**: un sólo árbol utiliza un modelo de crecimiento para desarrollar el sistema de ramas, con textura de hojas creadas con piezas con agujeros. Esta escena del bosque muestra como este modelo es efectivo y que simula los niveles de detalle asociados con dicho escenario. Nótese como este modelo establece un sentido real por la profundidad.

**Lámina 6**: esta escena fue tomada de la secuencia de apertura de un programa de noticias de al BBC, y que ilustra el rango de efectos especiales que son utilizados en dicha secuencia. El software fue implementado en una máquina AT&T Pixel Machine.


**Lámina 7**: esta escena al igual que la anterior, fue tomada de la secuencia de apertura del programa de noticias Breakfast News de la BBC.

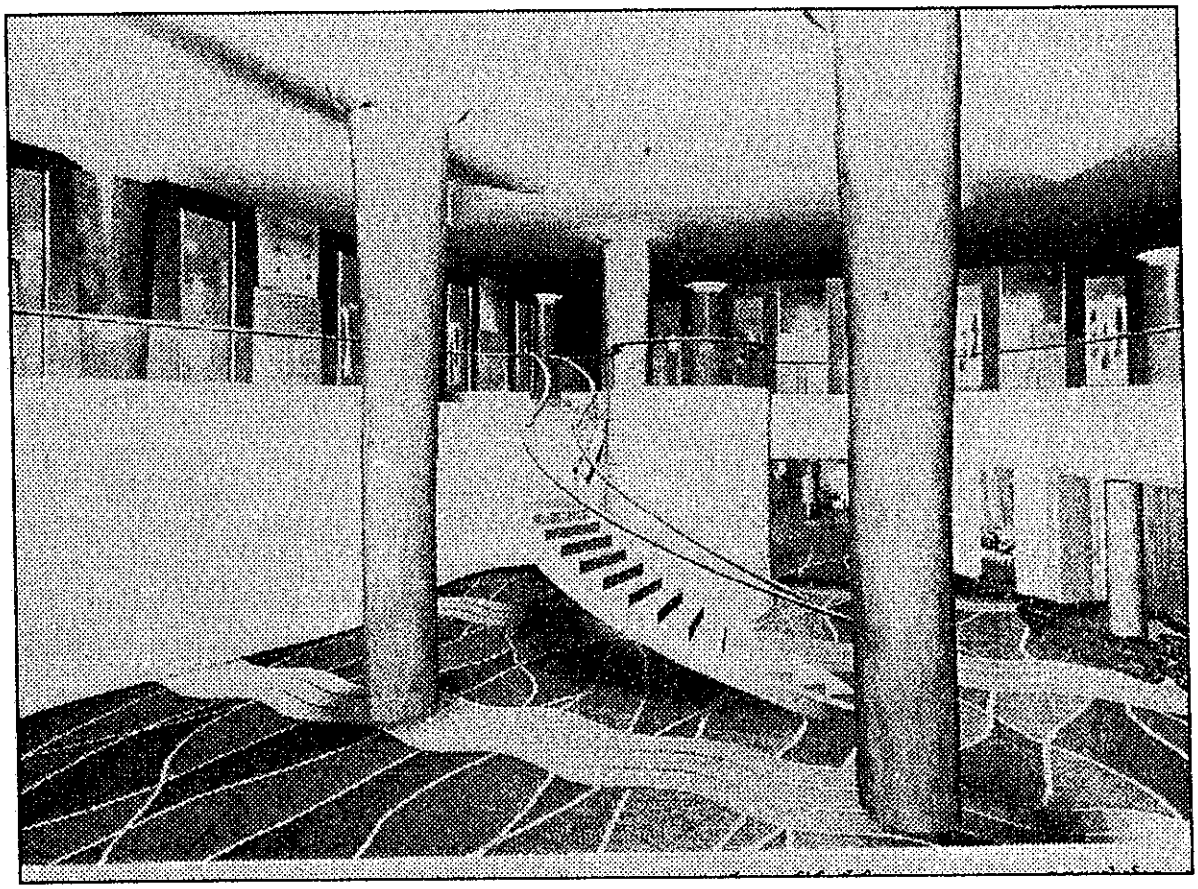


**Lámina 8**: esta imagen presenta el lobby de un hotel en Japón, que fue hecha por un diseñdor de Toronto (Canadá). Fue modelada usando Alias y muestra como las habilidades pueden ser usadas para crear escenas interiores reales.

**Lámina 9**: esta escena es un comercial de Twister Fries y es un ejemplo excelente del esfuerzo de creatividad en los anuncios de publicidad. La imagen fue compuesta usando el sistema de edición Quantel, que muestra la similitud de las papas fritas y el movimiento generado por computadora del agua.

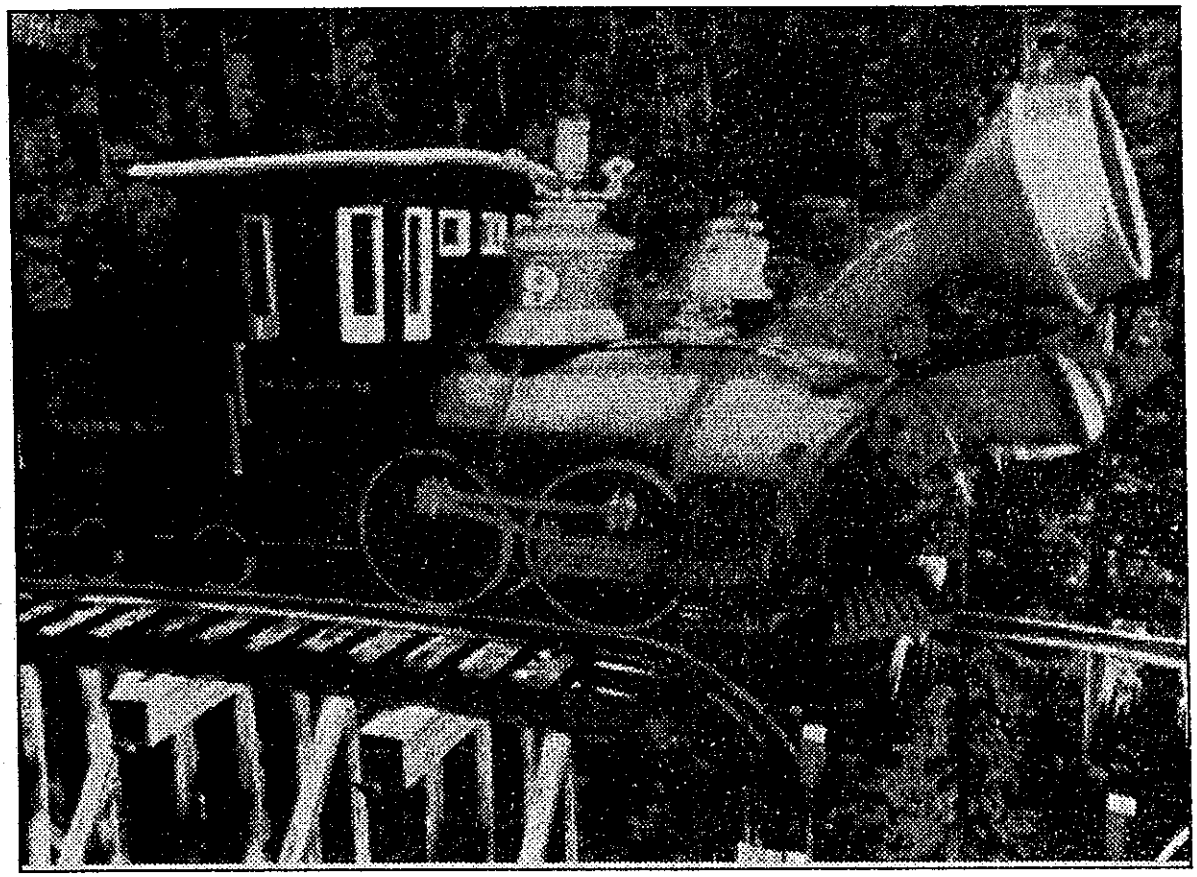


**Lámina 10**: esta imagen fue tomada de la animación 'Locomotion', utiliza la deformación. El FFDs proporciona las herramientas de animación para representar el modelo con un grado de flexibilidad.

# CONCLUSIONES

1. Existe una gran variedad de sistemas de animación disponibles en la actualidad basados en sistemas para PC's que ofrecen un ambiente integrado para modelar, animar y representar objetos de la realidad. Los animadores profesionales indican que no existe un 'sistema perfecto', ya que cada uno tiene sus ventajas y desventajas.

2. Par animar los objetos, una interface debe permitir al animador desarrollar un guión para describir el comportamiento de los objetos, fuentes de luz y posición de la cámara. Esta interface introducirá el uso de canales para controlar los atributos individuales del cuadro, y mostrar esta información gráficamente.

3. El equipo de computación está cambiando rápidamente y, a pesar del gran desarrollo en los últimos diez años, el tiempo de representación está medido en minutos. La capacidad del procesador es comparado con el incremento de la fidelidad de la imagen.

4. El almacenamiento en disco es importante para muchos sistemas de animación por computadora por la cantidad de imágenes que son almacenadas en el disco. Por lo que se pueden usar otras técnicas de procesamiento de imágenes, compuestas de otras imágenes, las cuales son retocadas usando programas antes de ser usadas.

5. La animación por computadora sigue evolucionando como una tecnología que proporciona un amplio rango de oportunidades relacionadas con costo y efectividad para animar imágenes.

6. La televisión utiliza la animación por computadora para titular programas, presentación de noticicias, gráficas explicativas, mapas de clima. Las imágenes sintéticas elaboradas por computadora no necesitan del diseño de materiales tradicionales, lo cual reduce el tiempo para su producción.

7. Muchos procesos industriales ya utilizan computadoras y sistemas CAD, que son usadas para visualizar y simular experimentos, mientras se usan datos en forma digital.

8. Los efectos especiales en películas han empujado a la animación por computadora hasta sus límites, pero dichos proyectos a menudo crean características que se encuentran en cierta forma más rápido en los sistemas de animación comercial. Las secuencias de efectos especiales normalmente involucran el uso en un formato de 70mm de alta resolución.

9. La visualización científica se ha convertido en la aplicación perfecta para la animación por computadora, ya que es difícil imaginarse cómo los problemas científicos pueden ser resueltos usando cualquier otro método. Las herramientas de modelación, animación y representación son efectivas para realizar imágenes de objetos familiares, y son igualmente efectivas en el traslado de conjuntos de datos abastractos multi-dimensionales derivados de experimentos y simulaciones por computadora, en secuencias animadas efectivas.

10. El comportamiento de la animación es una técnica poderosa para animar un conjunto de objetos que tienen que exhibir un conjunto de comportamientos, tal es el caso del comportamiento de los grupos de aves y peces.

11. La simulación dinámica es particularmente útil para animar estructuras conectadas en alguna forma jerárquica.

12. Para el desarrollo de las técnicas de animación por computadora, es necesario el uso de técnicas matemáticas para los diferentes cálculos de los atributos de los objetos a animar, ya sea para determinar la intensidad de color, posición, brllantez, reflejo de luz, la intensidad de las sombras.

# RECOMENDACIONES

1. La aplicación de la animación en la industria, mostraría con exactitud como son los componentes de las máquinas y como están ensamblados para saber su funcionalidad, además por medio un modelo simulado por computadora se tendría un diseño del producto final, lo que aumentaría los beneficios con respecto a la calidad de los productos y en el aumento de la producción

2. El implementar un proyecto de simulación de vuelo, en el que se pueda entrenar a los futuros pilotos de la escuela nacional de aviación, que beneficiaría a los mismos para su capacitación.

3. El uso de la animación por computadora en la televisión guatemalteca, ha aumentado en los últimos años, ya que se ha visto que el uso de estas innovaciones la audiencia aumenta, pero el uso de la animación está limitado ha ciertos programas de noticias y de presentaciones juveniles, esto también se puede hacer en programas educativos infantiles, para que los niños aprendan de una forma más interactiva.

4. Se puede implementar un curso para la carrera de Ingeniería Industrial e Ingeniería Mecánica Industrial, donde se enseñe el uso de que se le puede dar a la animación en estas diferentes áreas, por ejemplo, en el área de Mecánica Industrial puede ser útil para el diseño de componentes de una máquina, lo que resultaría en beneficio para saber con más precisión como está conformado cada uno de sus componentes, así, de esta forma se va a ir avanzando con la tecnología.

5. La implementación de un curso en la carrera de Ingeniería en Ciencias y Sistemas, en donde se mostrarían las diferentes aplicaciones de la animación, y donde pueden salir expertos que trabajen en ésta área en el país.

6. La aplicación de la animación por computadora en el área de arquitectura, mostraría exactamente como es el diseño de una construcción, desde el punto de vista del interior, exterior, elevaciones, iluminación, etc.

# GLOSARIO

**Anti-aliasing:** se refiere a estrategias que conciernen con la remoción o reducción de artecfactos al tamaño más pequeño, que sería un pixel.

**Atomo:** elemento primario de la composición química de los cuerpos.

**Back-face:** de un polígono planar es invisible cuando su superficie normal está frente a la cámara.

**Bit:** es un elemento indivual de un punto codificado en binario.

**Bump-mapping:** perturba una superficie normal durante su representación para crear una superficie abultada.

**Byte:** es un grupo de ocho bits, los cuales son capaces de almacenar 256 patrones únicos.

**CCS:** es el sistema de coordenadas de la cámara.

**CDF:** fluidos dinámicos computacionales, que es utilizado para calcular los efectos del flujo del viento o el flujo de fluidos sobre la superficie de un objeto.

**Cell animation:** es una animación tradicional que utiliza páginas delgadas de acetatos llamadas celdas, que son pintadas a mano para crear múltiples escenas.

**Coordenadas:** líneas que determinan la posición de un punto en el espacio o en una superficie.

**Extrusión:** proceso continuo de fabricación con prensas de las materias plásticas y de los metales.

**FEA:** análisis finito de elementos, donde los objetos son representados dentro una computadora como una mezcla espacial unida, de algunos cientos de nodos o puntos.

**FFD:** es una deformación libre de formas; una técnica de animación por comptuadora para distorsionar estructuras de 2-D y 3-D.

**Frame Buffer:** ver Frame Store.

**Free-form:** Ver FFD.

**Frame Store:** es un dispositivo de memoria usado para mantener una línea de pixels de la imagen donde cada pixel es mapeado por dentro de un determinado número de bits.

**Frustum:** es una pirámide truncada que representa el volumen de visión de la cámara.

**Knot-vector:** es una secuencia de valores paramétricos no decrecientes usados en la generación de curvas B-spline.

**Look Up Table:** es un dispositivo de memoria base usado con frame stores, que contiene intensidades de color substituidos por los valores mantenidos en el frame store.

**Matriz:** cuadro compuesto por número reales y complejos ordenados en líneas y columnas.

**Metrónomo:** instrumento para medir el tiempo musical e indicar el compás.

**Mip-mapping:** es un conjunto jerárquico de mapas de texturas filtradas desarrollado para reducir el tiempo de envío durante las operaciones de mapeo de texturas anti-aliasing.

**Modelos Moiré:** son efectos de movimientos de ondas causados por la interferencia de modelos regulares, tales como una textura repetitiva desplegada sobre un dispositivo raster.

**Molécula:** partícula formada de átomos que representa la cantidad más pequeña de un cuerpo que pueda existir en estado libre.

**Motion Blur:** está basado en la observación del movimiento de objetos que son grabados en una película.

**Obturador:** aparato que cierra el objetivo y puede abrirse durante un tiempo determinado para dar paso a la luz.

**Octree:** es una estrategia de particionamiento del espacio 3-D cuando un volumen es recursivamente dividido en una organización jerárquica de octantes.

**PAL:** es una acrónimo para la fase de la línea de alteración.

**Pitch:** es un movimiento rotacional alrededor de un eje-x horizontal ortogonal a la dirección-z

**Pixel:** es el elemento más pequeño de una imagen localizado en un punto en la pantalla.

**Ray-casting:** es una estrategia de detección de superficies escondidas generalmente asociadas con la representación de objetos CSG.

**Renderman:** es una interface poderosa especificada entre los modelos de 3-D y técnicas de representación fotorealistas.

**Roll:** rodar, ondular.

**Radiocidad:** es un modelo de iluminación global usado para intensidades de luz resultantes de múltiples reflejos difusos.

**Raster:** se refiere a una línea de pixels dibujados con CRT.

**Software:** se refiere a los programas de computadora utilizados para la creación e interpretación de las imágenes.

**Space-filling curves:** son curvas que teóricamente son capaces de convertir completamente un área con longitud infinita para un área finita.

**Spot-light:** es una fuente de luz capaz de iluminar una escena dentro un cono

angular restringido.

**Texel**: es la dirección más pequeña de un elemento de un mapa de textura.

**Vector**: segmento rectilíneo de longitud definida trazado trazado desde un punto dato y que sirve para representar ciertas magnitudes geométricas o magnitudes físicas.

**Virtual**: posible, que no tiene efecto actual. Que tiene existencia aparente pero no real: imagen, objeto virtual.

**Voxel**: es un espacio de volumen y representa al equivalente pixel en 3-D.

**WCS**: es el sistema mundial de coordenadas; es un espacio referenciado para construir modelos 3-D, fuentes de luz y cualquier trayectoria utilizada para propósitos de la animación.

**Workstations**: es la configuración de un procesador, pantalla, teclado y almacenamiento local en disco, todo conectado en una red de área local.

**Yaw**: es un movimiento rotacional alrededor del eje-y y es normalmente.

# BIBLIOGRAFIA

**Baecker, Ronald M**. Gráficas por Computador : Estados Unidos, Addison Wesley, 1,990

**Batty, Michael**. Microcomputer Graphics : Inglaterra, Chapman and Hall Ltd., 1,987

**Ezzell, Ben**. Programación de gráficos en Turbo C++ : Estados Unidos, Addison Wesley, 1,990

**Foley, James D**. Fundamentals of Interactive Computer Graphics : Estados Unidos, Addison Wesley, 1,983

**Hearn, Donadl / Baker, M. Pauline**. Gráficas por Computador : México, Prentice-Hall Hispanoamericana, S.A.1,986

**McClelland, Deke**. Drawing on the PC : Estados Unidos, Business One Irwing, Desktop Publishing Library, 1,991

**Morales Vides, José Fernando**. La Computadora, una generadora de mensajes visuales: Guatemala, Universidad del Valle, 1,987

**Pavlidis, Theodosios**. Algorithms for Graphics and Image Processing : Estados Unidos, Computer Science Press, 1,981

**Scott, Joan E**. Introduction to Interactive Computer Graphics : Estados Unidos, New York Wiley, 1,982

**Vince, John**. 3-D Computer Animation : Estados Unidos, Addison-Wesley, 1,992

**Voland, Gerard**. Modern Enigeering Graphics & Design : Estados Unidos, 1,987

**Watt, Alan**. Fundamentals of Three-dimensional Computer Graphics : Estados Unidos : 1,989

**Watt, Alan / Watt, Mark.** Advanced Animation and Rendering Techniques (Theory and Practice) : Estados Unidos, Addison-Wesley, 1,992

Aportes a la tecnología de ambientes gráficos de alta resolución. Business Week: The Gee-Whiz Company, 18 de Julio de 1,994

Computer Graphics: Proceedings.Estados Unidos, Anual Conference Series : New York, 1,993

Computer Images. Estados Unidos, TIME-LIFE Books , 1,985