Capítulo 3. Visualización

En este capítulo se presentan diversas técnicas y algoritmos para el desplegado de objetos poliédricos tridimensionales en pantalla, así como conceptos teóricos importantes relacionados con la visualización de escenarios.

Contar con un modelo eficiente para la representación de objetos tridimensionales no es suficiente para poder visualizar un escenario virtual, es necesario además emplear técnicas para desplegar estos objetos en pantalla, para lo cuál dos aspectos son muy importantes:

+*Determinación de superficies visibles*.- antes de dibujar figuras en pantalla es necesario saber qué superficies o partes de ellas son visibles al observador y cuáles no, para evitar que superficies ocultas sean visibles y que superficies visibles sean completa o parcialmente ocultadas.

+*Modelos de iluminación*.- intentan simular propiedades de los objetos, como color, cantidad de luz que reflejan, textura, etc. para hacerlos parecer reales.

Los cálculos necesarios para realizar estos aspectos son complejos y consumen una gran cantidad de tiempo, y si tomamos en cuenta que la mayoría de las escenas contienen miles de polígonos, realizar todos estos cálculos impediría tener un tiempo de respuesta apropiado para el navegador. Debido a esto innumerables técnicas han sido desarrolladas.

3.1 Determinación de superficies visibles

Muchos algoritmos han sido desarrollados para determinar las superficies visibles de una escena. Estos pueden ser categorizados en dos ramas:

+*Algoritmos que trabajan al nivel de objetos*: son realizados a la precisión con que cada objeto es definido y determinan la visibilidad de cada objeto. Una ventaja de estos algoritmos es que son independientes de las propiedades de la pantalla como su resolución, sin embargo, debe realizarse el paso extra de convertir los objetos a la resolución necesaria. Ejemplos de algoritmos de esta clase son el ordenamiento por profundidad y los árboles BSP.

+*Algoritmos que trabajan al nivel de imágenes*: éstos son realizados a la resolución de la pantalla y determinan qué objeto es visible en cada píxel. Como ejemplo de este tipo de algoritmos está el Z-buffer.

En la determinación de superficies visibles hay dos aspectos muy importantes:

+Saber qué superficie de un cuerpo es visible dada la posición del observador y cuál no (técnica llamada eliminación de caras ocultas o *back-face culling*). Si todos los cuerpos de una escena son poliédricos, entonces aproximadamente la mitad de las caras se ven y la otra mitad no; de esta manera, si una cara no es visible al observador es descartada y no

entra al proceso de *rendering*, lo que mejora el tiempo de respuesta. Cabe notar que esta técnica es sencilla de implementar.

+Saber qué superficie o parte de una superficie se debe pintar en pantalla.

3.1.1 Coherencia

En los mundos virtuales, los polígonos que los forman generalmente contienen relaciones entre varias partes de la escena o entre sus elementos cercanos. Estas interdependencias son conocidas como coherencia [PAVL82]. Coherencia puede ser especificada de varias maneras y puede ser expresada en términos tanto a nivel de objetos como de imágenes.

En términos más precisos coherencia se refiere al grado en que partes de una escena o su proyección presentan similitudes locales [FOLE90]. Así, algoritmos para determinar las superficies visibles pueden tomar ventaja de la coherencia en escenas, en el sentido de que es posible reutilizar cálculos realizadas con una parte del mundo o imagen en zonas cercanas, incluso sin cambios o con cambios incrementales que son más eficientes hacer que recalcular todo de la nada. Las principales coherencias definidas son las siguientes[PAVL82]:

+*Coherencia de objetos*: la visibilidad de un objeto puede ser determinada examinando el sólido que lo cubre, el cuál debe estar en una forma simple como una esfera o un poliedro con caras paralelas a los planos de coordenadas. Con esto, si dos objetos

están por completo separados, la comparación debe realizarse sólo entre los objetos y no entre las caras que los componen.

+*Coherencia de caras*: las superficies generalmente varían muy poco a través de la cara de un cuerpo, lo que permite que cálculos realizados sobre partes de una cara sean modificados incrementalmente para aplicarlos a regiones adyacentes. Además, debido al tamaño pequeño de los polígonos de un cuerpo comparados con el tamaño de la imagen final, a menudo es cierto que si una parte de la cara es visible, lo es la cara completa.

+*Coherencia de aristas*: la visibilidad de una arista cambia sólo cuando atraviesa otra arista visible o incide en una cara visible.

+*Coherencia de arista implicada*: si una superficie plana penetra a otra, entonces la línea de intersección puede ser determinado por dos puntos de intersección.

+*Coherencia de líneas de barrido*: segmentos que son visibles en una línea son propensos a ser visibles en la siguiente línea.

+*Coherencia de áreas*: un conjunto de píxeles adyacentes son a menudo cubiertos por la misma cara visible.

+*Coherencia de profundidad*: partes adyacentes de un cuerpo generalmente están cercanos en profundidad, mientras que diferentes cuerpos en la escena tienden a estar separadas en profundidad. Así, una vez que la profundidad de un punto de una superficies es calculada, la profundidad del resto de los puntos de la superficies puede ser calculada con una simple ecuación diferencial.

+*Coherencia de cuadros*: en animaciones, la imagen no cambia mucho de cuadro a cuadro. Cálculos realizados en un proceso puede ser utilizados en el siguiente.

+*Coherencia de geometría*: el número de posibles configuraciones de visibilidad y ocultamiento en cualquier vértice es limitada. Si todos los sólidos son convexos y si el

ángulo máximo entre aristas convergentes es menor a 180°, entonces las aristas extremas son visibles y la visibilidad de las otras aristas depende de sus posiciones relativas al plano definido por las dos primeras. Por lo tanto, en figura 3.1.A) las aristas A-B Y A-C son visibles y las demás dependen si se encuentran delante o atrás del plano definido por los puntos A, B y C. Por otro lado, si el ángulo excede los 180°, entonces todas las aristas tienen la misma visibilidad [PAVL82] (Fig. 3.1.B).

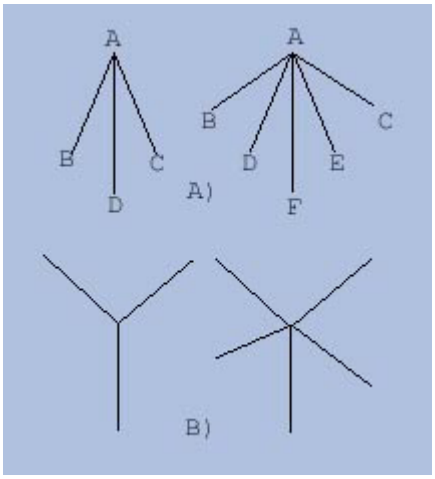

Fig. 3.1.- A)Si las aristas pertenecen a un objeto convexo, entonces las aristas extrema son visibles. B)Para un objeto convexo, todas las aristas alrededor de un vértice tienen la misma visibilidad.

3.1.2 Modelos de alambre

Utilizando el modelado de superficies vistos en la sección 2.1, una forma muy simple de presentar el mundo en pantalla consiste en simplemente pintar todas las aristas de todos los polígonos.

Los modelos de alambre, también conocidos como w*ireframes* tienen muchas ventajas contra las demás representaciones: no es necesario guardar información de color o

textura de la superficie del cuerpo, basta con almacenar los vértices y aristas de los polígonos, además, para realizar transformaciones sobre los objetos y mostrarlos no es necesario aplicar cálculos y técnicas para determinar las superficies visibles, basta con mover los vértices de todos los polígonos que forman la escena y pintar las aristas. Todo esto hace que presentar objetos como modelos de objetos sea una técnica muy rápida en cuanto a tiempo de respuesta y fácil de implementar. La figura 2.1 presenta cuerpos primitivos como modelos de alambres.

Sin embargo, tiene un gran inconveniente: la imagen generada es poco realista e incluso para objetos muy complejos, difícil de distinguir a menos que sea rotada o trasladada.

Debido a que secuencias de escenas usando modelos de alambres es suficiente para probar el dinamismo de una animación y la interacción en tiempo real es posible, es muy utilizada para realizar prototipos de animación para comerciales, efectos especiales en películas y sistemas de videojuegos[WATT92].

3.1.3 Algoritmo Z-Buffer

Este algoritmo fue desarrollado por Catmull y es uno de los más sencillos de implementar. Es necesario tener un buffer para el color de cada píxel y un z-buffer con el mismo número de entradas para guardar un valor de *z*. El algoritmo es sencillo: el buffer de color es inicializado al color de fondo y el z-buffer al valor del plano más alejado de la escena. Luego los polígonos son analizados por líneas de barrido en cualquier orden y para

cada uno de ellos, si el valor *z* del polígono en la posición (*x*, *y*) es más cercano que el valor actual guardado en el z-buffer, entonces se reemplaza el color del píxel en el buffer de color y el valor *z* en el z-buffer. Una vez que todos los polígonos son analizados, el buffer de color puede ser copiado a la pantalla y el proceso se repite de nuevo.

El pseudo-código del algoritmo es el siguiente [FOLE90], las funciones *WritePixel* y *ReadPixel* trabajan sobre el buffer de color y las funciones *ReadZ* y *WriteZ* sobre el z-buffer:

```
procedure zBuffer;
var
   pz:integer;
begin
  for y:=0 to YMAX do
     for x:=0 to XMAX do
        begin
           WritePixel (x,y,BACKGROUND_VALUE);
           WriteZ(x,y,0);
        End
  for each polygon do
     for each pixel in the polygon's projection do
        begin
            pz:=polygon's z-value at pixel coords (x,y);
            if pz >= ReadZ (x,y) then
               begin {New point is not farther}
                  WriteZ(x,y,pz)
                  WritePixel(x,y,polygon's color at pixel coords (x,y))
               end
        end
end;
```

Este algoritmo es muy flexible ya que las superficies de los cuerpos no tienen que ser necesariamente superficies planas, cualquier superficies que pueda calcularse en cada punto de la proyección un valor para el color y un valor para *z* puede utilizarse con este mecanismo. Además, para imágenes estáticas permite realizar imágenes con cualquier número de objetos debido a que el algoritmo dibuja un polígono a la vez sin necesidad de

tener información de los anteriores cuerpos en memoria. De la misma manera permite tener objetos de maneras muy diferentes, por ejemplo, dibujar polígonos, superficies curvas, fractales, etc. donde incluso cada uno de estos elementos es realizado por programas diferentes.

Si el cálculo del color de cada píxel toma mucho tiempo, puede mejorarse el rendimiento ordenando los polígonos de acuerdo a su valor de *z* más lejano y aplicando el algoritmo empezando con los polígonos más cercanos.

Sin embargo presenta algunas desventajas para implementar técnicas más sofisticadas como transparencias, reflexiones y sombras que pueden ser implementadas a costa de perder la simplicidad del algoritmo [GLAS89].

Debido a que la implementación de este algoritmo necesita una gran cantidad de memoria para guardar el buffer de color y el z-buffer para resoluciones muy grandes, algunas máquinas implementan este mecanismo desde el hardware.

3.1.4 Algoritmos de lista de prioridad

Los algoritmos de lista de prioridad trabajan tanto a nivel de objetos como a nivel de imágenes. La manera como funcionan es ordenando los polígonos en una estructura de manera que primero se pinten los más lejanos y después los más cercanos.

Este algoritmo es analógico a cómo pinta un pintor sus pinturas: comenzando con las más alejadas y dibujando poco a poco las más cercanas obscureciendo así regiones ya pintadas. Si ningún polígono se sobrepone a otro en el valor *z*, entonces basta con dibujar los polígonos en orden de lejanía, este algoritmo es conocido como *el algoritmo del pintor*. Sin embargo, generalmente los polígonos se interpolan de manera que ningún ordenamiento es adecuado. Esta sobreposición de polígonos puede ocurrir entre dos (Fig. 3.2, izquierda) o bien cíclicamente entre cualquier número de polígonos (Fig. 3.2, derecha). Cuando cualquiera de estos casos ocurre es necesario dividir los polígonos.

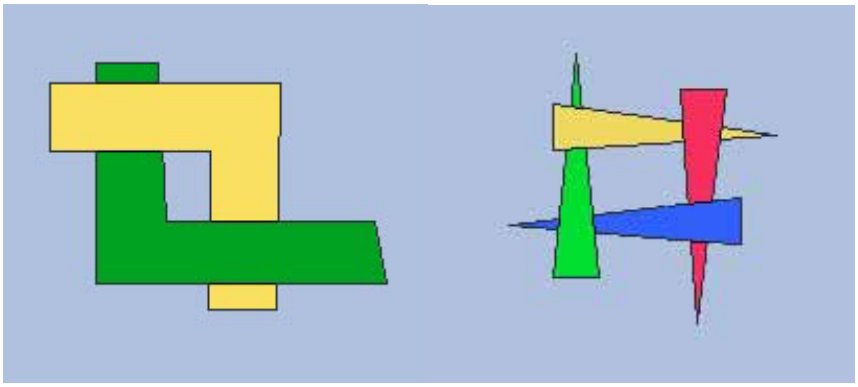

Fig. 3.2.- Interpolación entre dos polígonos.(izquierda). Interpolación cíclica(derecha).

Los principales algoritmos bajo este esquema son el ordenamiento por profundidad y los árboles BSP descritos a continuación.

3.1.4.1 Ordenamiento por profundidad

Este algoritmo fue desarrollado por Newell y Sancha y pinta los polígonos en el buffer de atrás para adelante (hacia el observador)[FOLE90]. Es una combinación de los

esquemas orientados a objetos y a imágenes. Para realizar este procedimiento es necesario implementar tres operaciones:

1.- Ordenar los polígonos de acuerdo al valor z más lejano de cada uno.

2.-Resolver cualquier situación en la que los valores *z* de cualquier par de polígonos se sobrepongan.

3.- Dibujar cada polígono de acuerdo a su valor *z*.

Los pasos 1 y 3 son sencillos de realizar, sin embargo el paso 2 es costoso en tiempo debido a que para saber si un polígono es sobrepuesto por otro en *z* hay que realizar varias pruebas contra todos los demás polígonos.

Dado un polígono *S*, para cada polígono que esté sobrepuesto a *S* en profundidad, hay que realizar las siguientes verificaciones. Si ninguna de estas pruebas es verdadera no es necesario dividir el polígono. Las pruebas están listadas según su complejidad [BAKE86]:

1.- Los rectángulos que encierran a los polígonos en el plano *xy* no se enciman.

2.- La superficie *S* se encuentra por completo atrás o enfrente de la superficie que se encima, respecto al plano de visión.

3.- La proyección de las dos superficies en el plano de visión no se encima.

Para este algoritmo hay que tomar en consideración que es posible que un conjunto de superficies se encimen en *z* cíclicamente (Fig. 3.2 derecha).

3.1.4.2 Árboles BSP

Su principal ventaja contra los demás algoritmos es que permite un desplegado lineal de los polígonos desde cualquier punto de vista, ya que no necesita ordenarlos cada vez el observador cambia de posición. Por esta razón es muy útil en aplicaciones donde los objetos son estáticos pero el observador no.

Su desventaja es un costo muy alto en tiempo y memoria al momento de crear el árbol (ver sección 5.2), sin embargo, una vez construida la estructura, el dibujo de los objetos es lineal con respecto al número total de polígonos, mediante un recorrido recursivo del árbol BSP. Para una descripción más detallada del algoritmo ver sección 5.3.

3.1.5 Algoritmos de línea de barrido

Estos algoritmos trabajan a nivel de imágenes de una manera muy parecida al relleno de polígonos, sólo que en lugar de llenar uno, se trabaja con múltiples polígonos a la vez. Para cada línea de barrido que es procesada, todos los polígonos que intersectan esa línea son examinados para determinar cuál es visible [BAKE86]. En cada posición de la línea evaluada, cálculos de profundidad son realizados para cada superficie para determinar cuál es la más cercana al plano de proyección.

Para este algoritmo es necesario utilizar una estructura parecida a la descrita en la sección 2.2.2 almacenando además las normales de cada polígono. Para facilitar la búsqueda de superficies que intersectan la línea que se está evaluando, es necesario crear

una lista de aristas que intersectan la línea de barrido actual ordenadas según su valor en *x* (incremental). Así mismo, se necesita una bandera para cada polígono inicializada a falso.

El mecanismo es el siguiente: primero se inicializa el buffer al color del fondo. Luego, para cada línea analizada se van tomando las aristas que intersectan esa línea. Si la arista es un límite izquierdo del polígono, es decir, se va a entrar al polígono, la bandera del polígono se enciende y al llegar a su limite derecho se apaga. Mientras esto se va haciendo, es necesario mantener una lista de los polígonos que tienen la bandera encendida, es decir, que se está dentro de ellos. Para cada punto se debe realizar una comparación de profundidad para cada uno de estos polígonos y del más cercano se obtiene el valor de color para ese píxel. Si en algún momento sólo hay un polígono activo, no es necesario realizar comparaciones de profundidad.

La ejemplificación de este algoritmo puede observarse en la figura 3.3. Para la línea de barrido 1, se genera la lista de aristas que la intersectan ordenadas por su valor *x* {*AB*, *BC*, *HE*, *GF*}. Al llegar a la arista *AB*, la bandera de la superficie *S1* es activada y toda la sección hasta llegar a la arista *BC*, donde la bandera de *S1* es desactivada, se dibuja con el color de la superficie *S1*. En este caso no es necesario realizar comparaciones de profundidad pues sólo hay una superficie activa. Sin embargo, en la línea 2, al llegar a la arista *HE*, dos superficies están activas (*S1* y *S2*) y para determinar el color a pintar es necesario determinar cuál de estas superficies es la más cercana en ese punto.

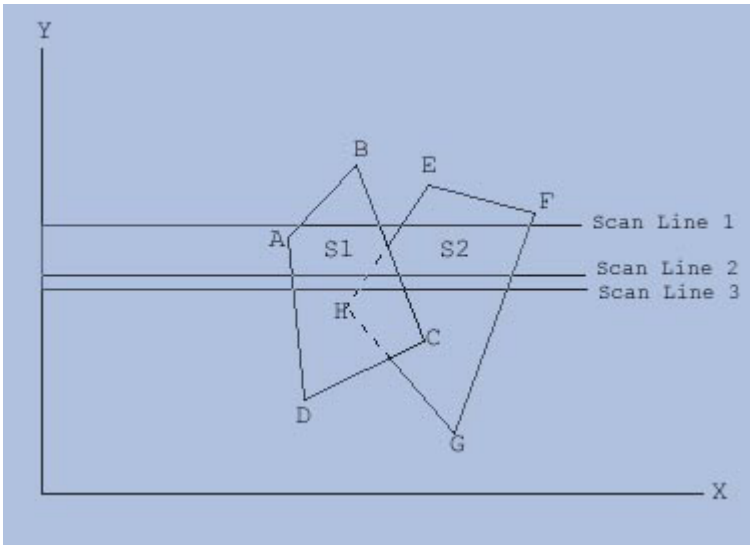

Fig. 3.3.- Líneas de barrido atravesando la proyección de dos superficies, *S1* y *S2* en el plano de proyección. Las líneas punteadas indican los límites de la superficie oculta.

Una de las ventajas de este tipo de algoritmos es que no es necesario realizar una división de polígonos, sin embargo el algoritmo es complejo y es necesario el uso de mucha memoria.

3.1.6 Algoritmos de subdivisión de áreas

El método de subdivisión de áreas utiliza la coherencia de áreas en una escena localizando áreas visibles que representan partes de una superficie simple [BAKE86]. La idea general de estos algoritmos es la estrategia de dividir y conquistar. Un área de la imagen proyectada es examinada; si en ésta es sencillo decidir qué polígonos son visibles, entonces son pintados. En caso contrario el área es subdividida en pequeñas áreas en las que la decisión lógica es aplicada recursivamente [FOLE90]. De esta manera, conforme las

áreas son más pequeñas, menor es el número de polígonos dentro de ellas y decidir qué polígonos son visibles es más fácil.

Para implementar este método, es necesario realizar varias pruebas que puedan identificar de una manera rápida si el área contiene una sola superficie o es muy compleja. Ahora, si hay una sola superficie en el área es necesario comparar la superficie contra los límites del área. En este caso hay cuatro posibles configuraciones entre una superficie y su área [BAKE86]: (Fig. 3.4).

+*Superficie circundante*: superficie que cubre completamente el área.

+*Superficie sobrepuesta*: es aquella que una región está dentro del área y otra fuera.

+*Superficie interna*: está completamente dentro del área.

+*Superficie externa*: está completamente fuera del área.

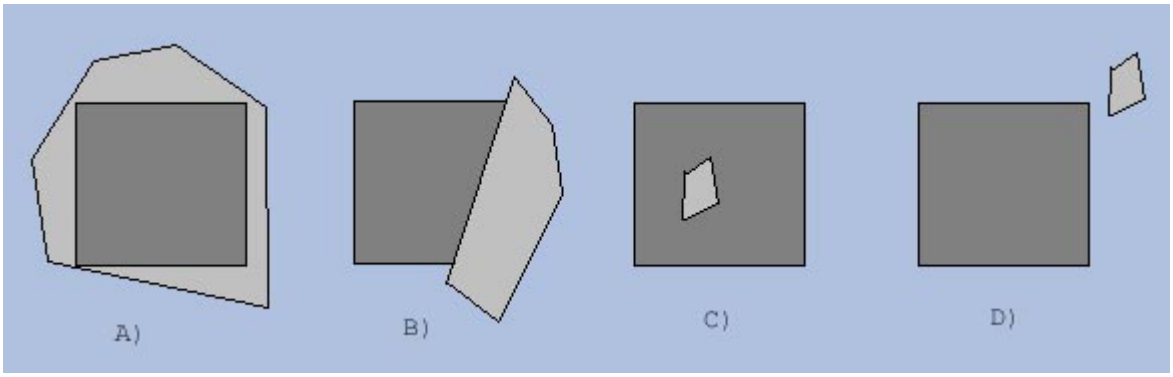

Fig. 3.4.- Posibles configuraciones entre la relación de una superficie y un área:
A)Superficie circundante, B)Superficie sobrepuesta, C)Superficie interna, D)Superficie externa.

Basados en estos casos, determinar si un área debe ser dividida puede ser decidido si alguno de los siguientes casos es falso:

+Todas las superficies están fuera del área.

+Sólo una superficie interna, externa o sobrepuesta está en el área.

+Una superficie circundante obscurece todas las superficies dentro de los límites del área.

3.1.7 Ray Traicing

La técnica *ray tracing*, también conocida como *ray casting*, determina la visibilidad de superficies trazando rayos imaginarios desde el ojo del observador hacia los objetos de la escena [FOLE90]. Esta técnica comenzó a utilizarse alrededor de 1980 y fue extremadamente estudiada en la siguiente década. Contrariamente a la creencia de que esta técnica fue inventada por desarrolladores de gráficas por computadora, fue René Descartes quién en 1637 en su publicación "René Descartes' treatise" utilizó esta técnica en geometría óptica para explicar la forma del arco iris[WATT92].

Originalmente, la técnica de *ray tracing* consiste en una división lógica del plano de proyección en una cuadrícula de la misma resolución que la de la pantalla. Luego, para cada celda en la cuadrícula del plano de proyección, un rayo es disparado desde el centro de proyección (ojo del observador) a través de la celda hacia los objetos de la escena. El número de rayos generados debe ser igual al número de píxeles en la pantalla. El píxel en esa celda es puesto al color que le corresponde al objeto que intersecte primero al rayo (Fig. 3.5).

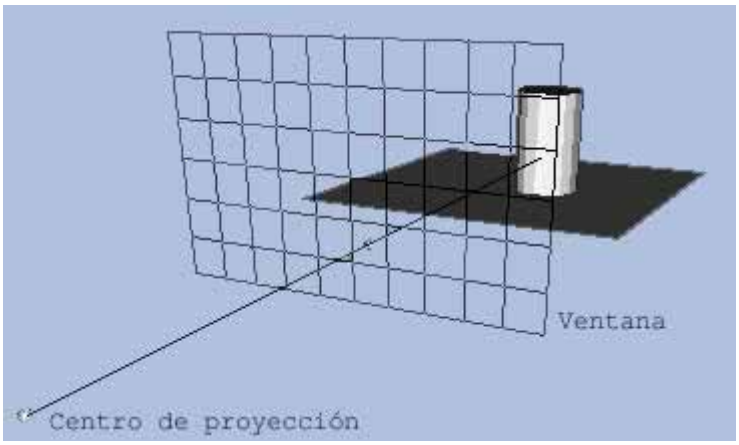

Fig. 3.5.- Un rayo es disparado desde el centro de proyección a través de cada píxel de la ventana para determinar la intersección más cercana.

Ray traicing permite realizar imágenes estáticas muy realistas ya que es posible implementar reflexiones y refracciones de luz, transparencias y sombras en objetos. Para producir reflexiones de la luz entre diferentes cuerpos el siguiente procedimiento recursivo puede ser aplicado: una vez que el rayo intersecta una superficie, éste es reflejado según su ángulo de incidencia. Este procedimiento continua hasta que el rayo incide en un generador de luz o pase a través de la escena. En cada punto de la superficie que incida el rayo, cálculos del color son realizados de acuerdo a las propiedades de la superficie y de la luz incidente, ya sea proveniente de un generador de luz o bien reflejada por otra superficie. La figura 3.6 muestra una imagen generada utilizando *ray traicing*.

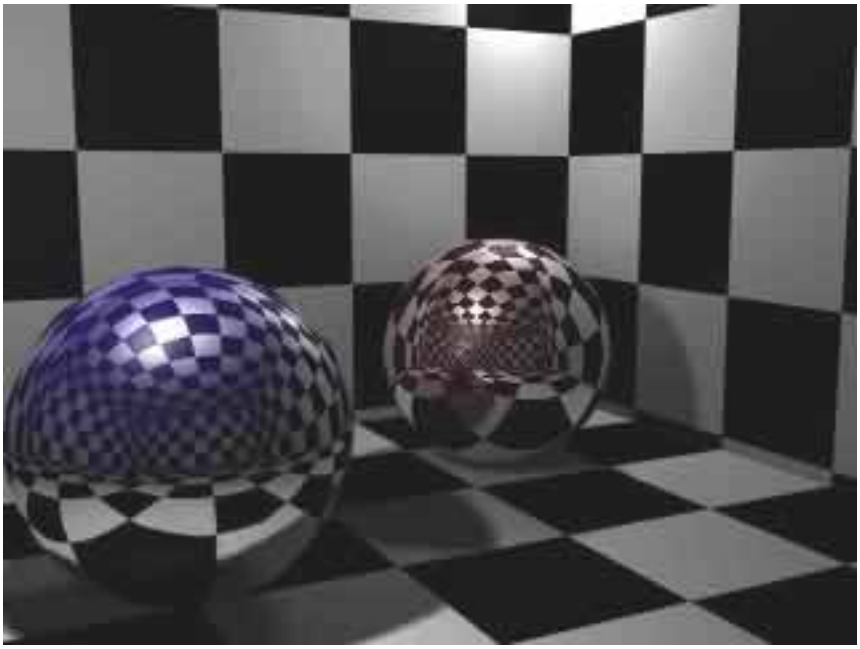

Fig. 3.6.- Dos esferas con una superficie reflejante en un cuarto cuadriculado. Imagen creada utilizando *ray tracing* con el programa Studio 3D.

3.2 Modelos de iluminación

Utilizando los modelos a nivel de objetos, una vez que se conoce qué superficies deben ser dibujadas en pantalla, el siguiente paso consiste en realizar los cálculos de color para dibujar cada una de las superficies. En algoritmos que trabajan a nivel de imagen, donde es dibujado un píxel a la vez, estos cálculos son efectuados para cada píxel.

Las técnicas para dibujar las superficies, también conocidas como iluminación o sombreado de superficies, juegan un papel importante en la generación de imágenes realistas. Utilizando estas técnicas, objetos curvos modelados mediante mallas de polígonos pueden ser dibujados muy semejantes a cómo son en la realidad, simulando su curvatura

muy eficientemente. De igual manera que en el modelado de sólidos, técnicas que producen mejores sombreados también requieren de más tiempo de cálculo.

Las superficies son sombreadas de acuerdo a su posición, orientación y características tanto de la misma superficie como de las luces que la iluminan. Algunos objetos tienen superficies brillantes mientras que otros superficies que reflejan muy poca luz. Además, algunos objetos están constituidos de materiales muy opacos, mientras que otros son más o menos transparentes. Un modelo de iluminación debe considerar todas estas propiedades para producir una iluminación real sobre las superficies de los objetos.

Una de las maneras en que los gráficos por computadora intentan simular el hecho de que objetos de diferentes tipos de materiales responden de manera diferente cuando son iluminados, es simulando propiedades de las superficies e interactuando estos modelos con el ambiente geométrico [WATT92].

En esta sección se muestran las técnicas de iluminación más conocidas e implementadas en el navegador: iluminación de Lambert, iluminación de Gouraud e iluminación de Phong, así como los efectos de luz también implementados, como luz ambiental, atenuación de la luz, reflexión difusa y reflexión especular de los cuerpos.

3.2.1 Reflexión difusa

Cuando los escenarios son iluminados por fuentes de luz dirigidas, es decir, que todos los rayos de luz provienen de un punto, el brillo de los objetos varía de una zona a otra, dependiendo de la dirección y la distancia de la fuente de luz.

Las superficies presentan reflexión difusa que provoca que éstas aparezcan con el mismo brillo desde cualquier punto de vista pues reflejan la luz con la misma intensidad en todas las direcciones[FOLE90]. Sin embargo, la cantidad de luz reflejada en cada superficie varía en función del ángulo de incidencia con la normal del polígono[WATT92].

De una manera más formal, para una superficie *S*, la cantidad de luz reflejada por ella depende sólo del ángulo θ entre el vector *L* que va de la superficie hacia la luz y la normal de la superficie *N* (Fig. 3.7), y es proporcional al coseno de este ángulo (ley de Lambert).

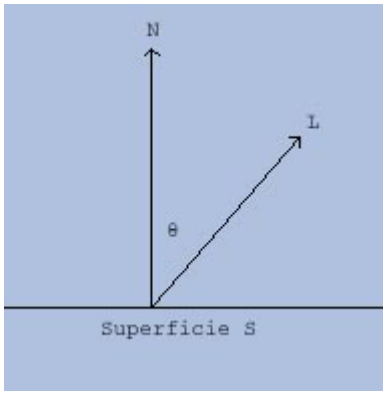

Fig. 3.7.- Ángulo de incidencia θ entre la dirección de la luz *L* y la superficie de la normal *N*.

La ecuación para calcular la intensidad de un color *I* es

$$I=I_p*Kd*\cos\theta \quad (3.1)$$

donde I_p es la intensidad de la luz, *Kd* es el coeficiente de reflexión de la superficie que varía entre 1 y 0 según las propiedades del objeto y el ángulo de incidencia θ está en el rango 0°<0<90°. Objetos que reflejan mucha luz tienen un coeficiente de reflexión cercano a 1, mientras que aquellos que absorben la mayoría de la luz incidente tienen valores cercanos a 0.

Si en lugar de un solo valor para la intensidad de color se utiliza una representación de color mediante los valores RGB (rojo, verde y azul), se puede calcular cada componente de color con una ecuación por separado. Por ejemplo, para el componente rojo I_r:

$$I_r=I_{pr}*Kd_r*\cos\theta \quad (3.2)$$

Bajo este esquema, los coeficientes de reflexión difusa Kd_r, Kd_g y Kd_b determinan el color del cuerpo.

3.2.1.1 Luz ambiental

En un escenario real, la luz emitida por la fuente de luz es reflejada por algunas superficies con diferente intensidad y color, produciendo lo que se conoce como luz ambiental. Cuando la luz ambiental es reflejada por una superficie, produce una iluminación uniforme sobre la superficie desde cualquier punto de vista, siempre que ésta sea visible.

Por simplicidad, se puede considerar la intensidad de la luz ambiental como una constante I_a. Si las superficies son expuestas sólo a luz ambiental, se puede expresar la intensidad de la reflexión difusa en cualquier punto de la superficie como

$$I = I_a * Kd \tag{3.3}$$

Sin embargo son pocos los casos en los que las superficies son iluminadas sólo con luz ambiental. Para obtener sombreados más realistas, es necesario tomar en cuenta el color del objeto, añadiendo la ecuación 3.1 a la ecuación 3.3:

$$I = I_a * Kd + I_p*Kd*\cos\theta \tag{3.4}$$

3.2.1.2 Atenuación de luz

Los modelos de luz presentados tienen la deficiencia de que si dos o más superficies paralelas se enciman en su proyección, no será posible distinguirlas unas de otras debido a que se iluminarán con la misma intensidad, ya que las ecuaciones anteriores no toman en cuenta la distancia de las superficies a las fuentes de luz. Para lograr este efecto es necesario introducir un factor de atenuación f_{att} a la ecuación 3.4:

$$I = I_a * Kd + f_{att} * I_p*Kd*\cos\theta \tag{3.5}$$

Debido que la cantidad de luz que llega a una superficie es inversamente proporcional a la distancia d_L de la superficie a la fuente de luz, el factor de atenuación puede elegirse como:

$$f_{att} = 1 / (d_L + d_o) \tag{3.6}$$

siendo d_o una constante para prevenir la división por 0.

3.2.2 Reflexión especular

A ciertos ángulos, las superficies brillantes reflejan toda la luz incidente, independientemente de sus valores de reflexión. Este fenómeno produce una región de luz reflejada del mismo color que la luz incidente. Para un reflector ideal, como un espejo, el ángulo de incidencia es igual al ángulo de reflexión[BAKE86].

Sea *R* el vector unitario que representa la dirección de la reflexión especular, *V* el vector que apunta en la dirección del observador y *L* el vector que apunta a la fuente de luz (Fig. 3.8) la reflexión especular puede verse en un reflector perfecto sólo si *V* y *R* coinciden ($\Phi = 0$). Objetos reales presentar reflexión especular en un rango alrededor del vector *R*.

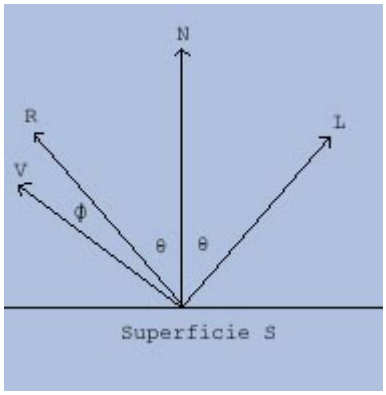

Fig. 3.8.- Para un reflector perfecto, el ángulo de incidencia θ es el mismo que le ángulo de reflexión.

El método desarrollado por Phong Bui Tuong para modelar la reflexión especular se basa en definir la reflexión especular proporcional a $\cos^n \Phi$. El valor asignado a *n*

determina el tipo de superficie que se está viendo, para objetos muy brillantes *n* toma valores muy grandes (200 o más).

La reflexión especular es utilizada generalmente junto con la reflexión difusa mediante la siguiente ecuación:

$$I = I_a * Kd + f_{att} * I_p (Kd*\cos\theta + Ks * \cos^n \Phi) \quad (3.7)$$

3.2.3 Modelos de iluminación de polígonos

Para sombrear una superficie se puede calcular para cada punto visible la normal de la superficie y aplicar el modelo de iluminación deseado. Sin embargo, este método es demasiado costoso en cálculos, por lo que se han desarrollado modelos más eficientes para sombrear polígonos: iluminación de Lambert, iluminación de Gouraud e iluminación de Phong.

3.2.3.1 Iluminación de Lambert

Bajo ciertas condiciones un objeto con superficies planas puede ser sombreado de una manera efectiva utilizando intensidades constantes de color para cada superficie[BAKE86]. En el caso cuando una superficie es expuesta sólo a luz ambiental y no presenta propiedades como texturas o sombras, el modelo de iluminación de Lambert genera una representación bastante precisa de la superficie, sin embargo, en objeto curvos aproximados con mallas de polígonos, éstos se distinguen claramente, disminuyendo el efecto de parecer superficies curvas.

Con la iluminación de Lambert, también conocida como iluminación de caras, cada polígono del objeto es dibujado con una intensidad constante. Para cada polígono, la intensidad de color en un punto del polígono es calculada y todo el polígono es dibujado con ese valor. La iluminación de Lambert se basa en la reflexión difusa donde la intensidad de la superficie depende del ángulo de incidencia de la luz con la normal de la superficie y es proporcional al coseno del ángulo (ver sección 3.2.1). Este método es muy efectivo cuando la orientación de los planos de las caras adyacentes de los objetos cambian muy poco. Para objetos donde esto no se cumple, es posible mejorar la imagen aumentando el número de polígonos que representan al objeto. La figura 3.9 muestra un conjunto de esferas pintadas empleando el modelo de iluminación de Lambert.

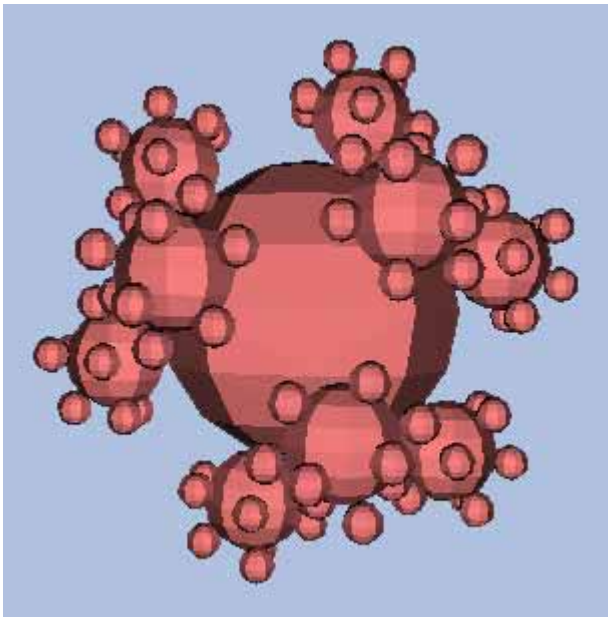

Fig. 3.9.-Imagen de esferas realizadas con un nivel de partición igual a nueve e iluminación de Lambert. Nótese cómo los polígonos que forman las esferas se distinguen claramente.

La técnica de iluminación de Lambert es la más rápida y desde el punto de vista de la programación la más sencilla; es utilizada comúnmente cuando un diseñador o animador quiere verificar la forma o movimiento de un modelo pero no necesita mucho realismo, sino resultados rápidos[GLAS89]. Su rapidez radica en que sólo la brillantez de un punto de cada polígono es calculado para pintar el polígono completo, comparado con otras técnicas en las que el valor de cada punto del polígono es calculado. La diferencia de tonalidad entre polígonos produce un efecto que visualiza las aristas aunque no son dibujadas explícitamente.

3.2.3.2 Iluminación de Gouraud

La iluminación de Gouraud o interpolación de sombras es una técnica para hacer parecer un modelo de polígonos que representan una superficie curva muy parecida a la real, mejorando la desventaja que presenta Lambert donde las superficies se ven formadas por polígonos. La idea central de la iluminación de Gouraud es calcular la intensidad de color para cada vértice del polígono y luego interpolar trilinealmente estos valores a lo largo del polígono.

Bajo este esquema, las normales de cada vértice del polígono deben ser almacenadas en su estructura; si la descripción del objeto (archivo externo) no contiene la información de las normales de cada vértice, cada una de éstas puede ser calculada como el promedio de las normales de los polígonos que contienen a ese vértice.

Esta técnica de iluminación genera objetos poliédricos con color no constante en sus caras, dando la ilusión de tener superficies curvas, pero es considerablemente más lenta debido a que es necesario calcular el color en cada vértice del polígono (que por lo menos tiene tres) y pintar el polígono píxel por píxel, interpolando estos valores.

Tomando como referencia la Fig. 3.10, las fórmulas de interpolación de intensidades son:

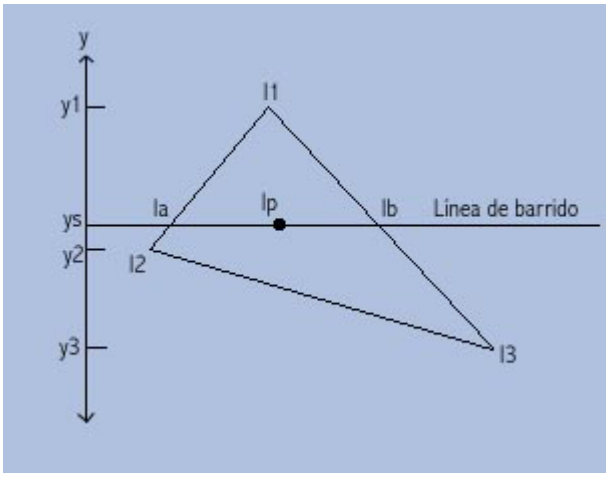

Fig. 3.10.- Interpolación de intensidades a lo largo de las aristas de un polígono y de las líneas de barrido.

$$I_a = I_1 - (I_1 - I_2)\frac{(y_1 - y_s)}{(y_1 - y_2)}$$

$$I_b = I_1 - (I_1 - I_3)\frac{(y_1 - y_s)}{(y_1 - y_3)}$$

$$I_p = I_b - (I_b - I_a)\frac{(x_b - x_p)}{(x_b - x_a)}$$

A pesar de ser una técnica muy eficiente para simular objetos reales curvos, presenta varios deficiencias. Las luces muy enfocadas en una región (como de una linterna), pueden no ser mostradas si ninguna sección de luz cae sobre los vértices del

polígono. Además, aunque elimina la discontinuidad de intensidades entre los polígonos, la interpolación lineal de las aristas puede producir rayas más o menos brillantes sobre la superficies, llamadas *Mach bands*. Otro problema es que aunque el número de polígonos que representan la superficie curva aumente, la silueta de los objetos continua siendo poligonal. La siguiente figura muestra en escenario compuesto por esferas pintadas utilizando la técnica de Gouraud.

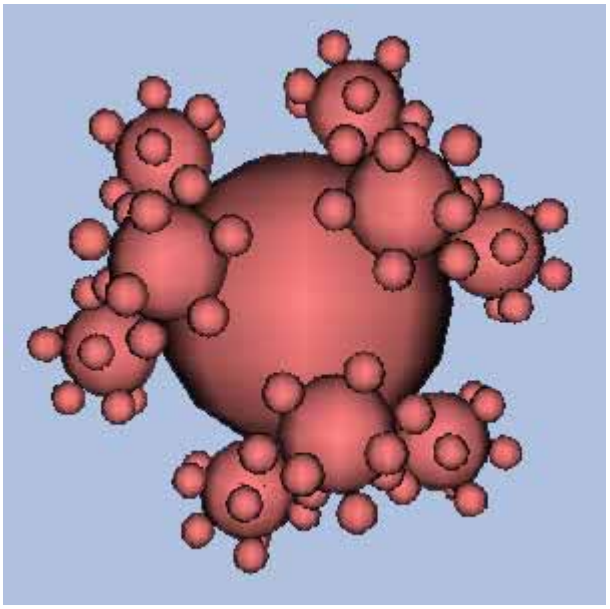

Fig. 3.11.-Imagen de esferas realizadas con un nivel de partición igual a nueve e iluminación de Gouraud. Los polígonos que forman las esferas no son visibles, sin embargo la silueta de las esferas continua viéndose poligonal y pueden observarse ligeras líneas brillantes en las aristas.

3.2.3.3 Iluminación de Phong

La técnica de iluminación de Phong, también conocida como sombreado por interpolación de normales, se basa en la misma técnica de dibujar los polígonos píxel por píxel en líneas de barrido, sin embargo, para cada punto se calcula su normal interpolando

trilinealmente las normales de los vértices de la arista que contienen a ese punto. Una vez obtenida la normal, se calcula la intensidad de color en ese punto utilizando el modelo de iluminación de luz especular.

La técnica de iluminación de Phong mejora varios problemas de Gouraud, como las luces enfocadas y las *Mach bands*, a un costo considerablemente alto de cálculos, debido a que para interpolar la normal de los vértices se debe realizar un cálculo para cada uno de los componentes x, y, z de la normal. Además, es necesario calcular para cada punto, el vector que va al observador. A pesar de la desventaja de tiempo de cálculos, la técnica de Phong genera imágenes muy realistas de los objetos (Fig. 3.11).

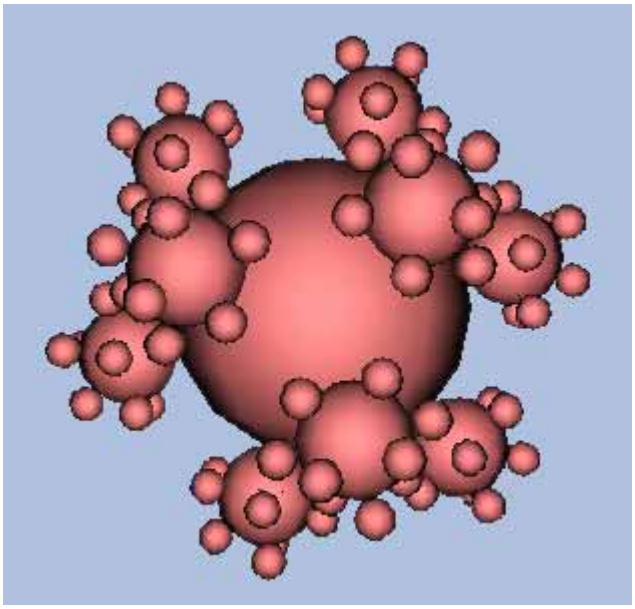

Fig. 3.12-Imagen de esferas realizadas con un nivel de partición igual a nueve e iluminación de Phong.