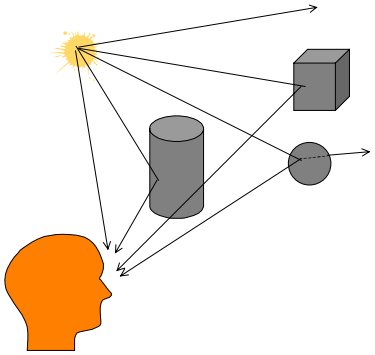
Iluminación y Sombreado

Desde una perspectiva física, una superficie puede emitir luz por su propia emisión, como focos de luz, o reflejar luz de otras superficies que la iluminan. Algunas superficies pueden reflejar y emitir luz. El color que se ve en un punto de un objeto está determinado por las múltiples interacciones entre las fuentes de luz y superficies reflectivas. Este es un proceso recursivo.

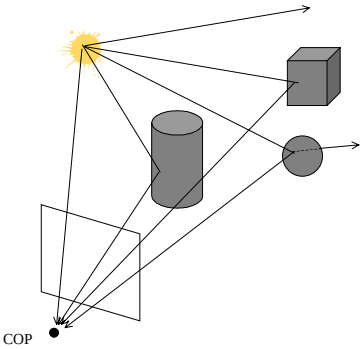
El problema consiste de dos aspectos:
1. Modelar las fuentes de luz en una escena.
2. Construir un modelo de reflexión que trate con las interacciones entre materiales y luz.

Para comprender el proceso de iluminación, se puede comenzar siguiendo los rayos de luz de un punto fuente, donde el observador ve solamente la luz que emite la fuente y que llega a los ojos; probablemente a lo largo de complejos caminos y múltiples interacciones con objetos en la escena, como se muestra en la siguiente figura:



- Si un rayo de luz entra al ojo directamente de la fuente, se verá el color de la fuente.
- Si un rayo de luz pega en una superficie que es visible al observador, el color visto se basará en la interacción entre la fuente y el material de la superficie: se verá el color de la luz reflejado de la superficie a los ojos.

En término de gráfica por computadora, se reemplaza el observador por el plano de proyección, como se ve en la siguiente figura:



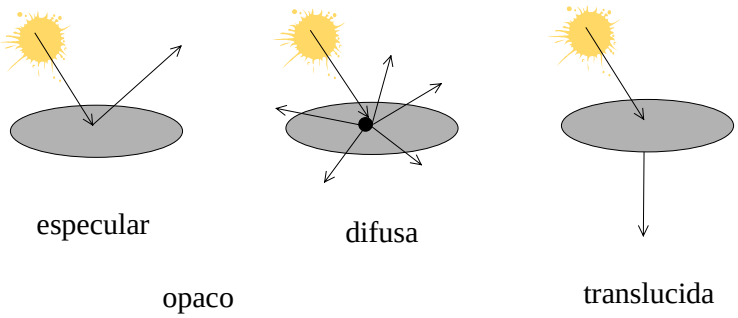
La ventana de recorte en este plano se mapea a la pantalla. El recorte del plano de proyección y su mapeo a la pantalla significa un número particular de pixeles de despliegue. El color de la fuente de luz y las superficies determina el color de uno o mas pixeles en el frame buffer.

Se debe considerar solo aquellos rayos que dejan las fuentes y llegan al ojo del observador, el COP, después de pasar por el rectángulo de recorte. Nótese que la mayoría de los rayos que dejan la fuente no contribuyen a la imagen y por lo tanto, no son de interés aquí.

La naturaleza de interacción entre los rayos y las superficies determina si un objeto aparece rojo o rosado, claro u obscuro, reluciente o no. Cuando la luz da en una superficie, parte se absorbe, y parte se refleja.

- Si la superficie es **opaca**, reflexión y absorción significará de toda la luz que dé en la superficie.
- Si la superficie es **translucida**, parte de la luz será transmitida a través del material y podrá luego interactuar con otros objetos.

Un objeto iluminado por luz blanca se ve rojo porque absorbe la mayoría de la luz incidente pero refleja luz en el rango rojo de frecuencias.
Un objeto relumbrante se ve así porque su superficie es regular, al contrario de las superficies irregulares.
El sombreado de los objetos también depende de la orientación de las superficies, caracterizado por el vector normal a cada punto. Las interacciones entre luz y materiales se puede clasificar en tres grupos, como se ve en la siguiente figura:



1. **Superficies especulares** (espejos) - se ven relumbrantes porque la mayoría de la luz reflejada ocurre en un rango de ángulos cercanos al ángulo de reflexión. Espejos son **superficies especulares perfectas**. La luz del rayo de luz entrante puede absorberse parcialmente, pero toda la luz reflejada aparece en un solo ángulo, obedeciendo la regla que el ángulo de incidencia es igual al ángulo de reflexión.
2. **Superficies Difusas** - se caracterizan por reflejar la luz en todas las direcciones. Paredes pintadas con *mate* son reflectores difusos. **Superficies difusas perfectas** dispersan luz de manera igual en todas las direcciones y tienen la misma apariencia a todos los observadores.
3. **Superficies translucidas** - permiten que parte de la luz penetre la superficie y emerja de otra ubicación del objeto. El proceso de **refracción** caracteriza al vidrio y el agua. Cierta luz incidente puede también reflejarse en la superficie.

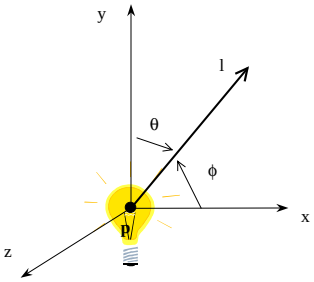
Fuentes de Luz

La luz puede dejar una superficie mediante dos procesos fundamentales:

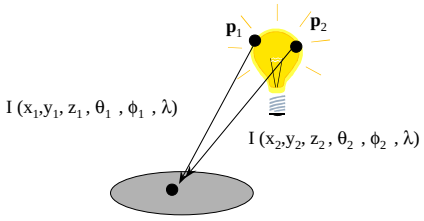
- Emisión propia
- Reflexión

Normalmente se piensa en una fuente de luz como un objeto que emite luz solo mediante fuentes de energía internas, sin embargo, una fuente de luz, como un foco, puede reflejar alguna luz incidente a esta del ambiente. Este aspecto no será tomado en cuenta en los modelos más sencillos.

Si se considera una fuente como en la siguiente figura, se le puede ver como un objeto con una superficie.



Cada punto (x,y,z) en la superficie puede emitir luz que se caracteriza por su dirección de emisión (θ,ϕ) y la intensidad de energía emitida en cada frecuencia λ. Por lo tanto, una fuente de luz general se puede caracterizar por la **función de iluminación** *I*(x, y, z, θ, ϕ, λ) de seis variables. Nótese que se requiere dos ángulos para especificar una dirección, y que se asume que cada frecuencia puede considerarse de manera independiente. Desde la perspectiva de una superficie iluminada por esta fuente, se puede obtener la contribución total de la fuente al integrar sobre la superficie de la fuente, un proceso que considera los ángulos de emisión que llegan a la superficie bajo consideración y que debe considerar también la distancia entre la fuente y la superficie.



Para una fuente de luz distribuida, como un foco de luz, la evaluación de este integral es difícil, usando métodos analíticos I numéricos. A menudo, es mas fácil modelar la fuente distribuida con polígonos, cada una de las cuales es una fuente simple, o aproximando a un conjunto de fuentes de punto.

Se considerarán cuatro tipos básicos de fuentes, que serán suficientes para generar las escenas más sencillas:
1. Luz ambiente
2. Fuentes de punto
3. Spotlights (Luces direccionales)
4. Luces distantes

Fuentes de Color

No solamente las fuentes de luz emiten diferentes cantidades de luz en diferentes frecuencias, pero también sus propiedades direccionales varían con la frecuencia. Por lo tanto, un modelos físicamente correcto puede ser muy complejo. Para la mayoría de las aplicaciones, se puede modelar fuentes de luz en base a tres componentes primarios, RGB, y puede usar cada uno de los tres colores fuentes para obtener el componente de color correspondiente que un observador humano vería.

Se describe una fuente mediante una función de **intensidad** o **luminancia** de tres componentes:

$$I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$$

Cada uno de los componentes es la intensidad de los componentes rojo, verde y azul independientes. Como las computaciones de color involucran tres cálculos similares pero independientes, se presentarán ecuaciones escalares sencillas, con el entendimiento de que pueden representar cualquiera de los tres colores.

Luz Ambiente

En algunos cuartos, las luces se diseñan y ubican para proveer iluminación uniforme en el cuarto. Tal iluminación se logra mediante fuentes grandes con difusores cuyo propósito es esparcir la luz en todas las direcciones. Se puede crear una simulación precisa de tal iluminación, modelando todas las fuentes distribuidas, y luego integrando la iluminación de estas fuentes en cada punto de una superficie reflectora. Hacer tal modelo y generar la escena sería un tarea formidable para un sistema gráfico, especialmente si se desea ejecución en tiempo real. De manera alternativa, se puede ver el efecto deseado de las fuentes: lograr un nivel de luz uniforme en el cuarto. Esta iluminación uniforme se llama **luz ambiente**. Si se sigue este segundo enfoque, se puede postular una intensidad ambiente en cada punto del ambiente. Por lo tanto, iluminación ambiente se caracteriza por una **intensidad $\mathbf{I_a}$**, que es idéntica en cada punto de la escena.
La fuente de ambiente tiene tres componentes:

$$I_a = \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix}$$

Aunque cada punto en la escena recibe la misma iluminación ambiente de $\mathbf{I_a}$, cada superficie la refleja de manera diferente.

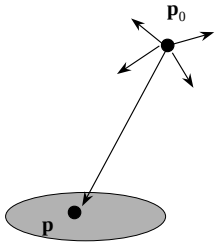
$$I_a(p) = I_a$$

Fuentes

Una **fuente de punto** ideal emite luz de manera igual en todas las direcciones. Se caracteriza una fuente ideal ubicada en un punto p_0 por un vector color de tres componentes:

$$I(p_0) = \begin{bmatrix} I_r(p_0) \\ I_g(p_0) \\ I_b(p_0) \end{bmatrix}$$

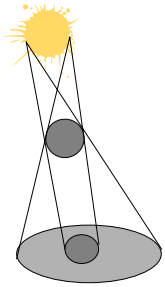
La intensidad de iluminación recibida de una fuente de punto es proporcional al inverso del cuadrado de la distancia entre la fuente y la superficie. Por lo tanto, en un punto *p*, como se ve en la siguiente figura:



La intensidad de la luz recibida de la fuente de punto está dada por el vector

$$I_p(p, p_0) = \frac{1}{|p - p_0|^2} I(p_0)$$

El uso de fuentes de punto en la mayoría de las aplicaciones se determina mas por su facilidad de uso que por su similitud a la realidad física. Escenas generadas solo con fuentes de punto tienden a tener un alto contraste; los objetes se ven brillantes u obscuros. En el mundo real, los tamaños grandes de las fuentes de luz contribuye a las escenas mas suaves, como se ve en la siguiente figura:



Algunas áreas están totalmente en la sombra, o en la **umbra**, mientras que otras están parcialmente en la sombra, o **penumbra**. Se puede mitigar el efecto de alto contraste al añadir luz ambiente a una escena.

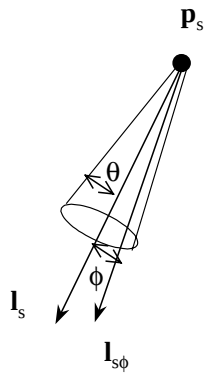
La distancia también contribuye al efecto abrupto en las fuentes de punto. Aunque el término inverso al cuadrado de la distancia es correcto para fuentes de punto, en la práctica normalmente se reemplaza por un término del tipo $(a+bd+cd^2)^{-1}$, donde d es la distancia y las constantes a, b, y c se escogen para suavizar la luz, lo que significaría una ecuación de la forma

$$I_p(p, p_0) = \frac{1}{a + b|p - p_0| + c|p - p_0|^2} I(p_0)$$

Nótese que, si la fuente de luz está lejana de las superficies en la escena, la intensidad de la luz de la fuente será suficientemente uniforme que el término de distancia será constante sobre las superficies.

Spotlights (direccionales)

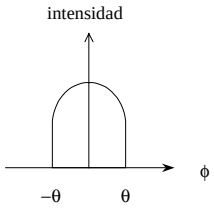
Los spotlights se caracterizan por un rango delgado de ángulos por los cuales se emite luz. Se puede construir un spotlight sencillo de una fuente de punto limitando los ángulos de donde la luz de la fuente se puede ver. Se puede usar un cono cuyo ápice está en $\mathbf{p}_s$, apuntando en la dirección $\mathbf{l}_s$, y cuyo ancho está determinado por el ángulo θ, como se muestra en la siguiente figura:



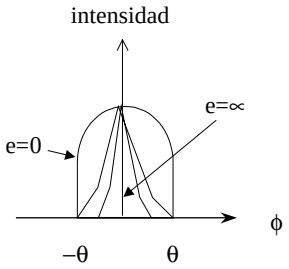
Si $\theta = 180$, el spotlight se vuelve una fuente de punto.

Spotlights mas realistas se caracterizan por una distribución de luz dentro del cono, normalmente con la mayoría de la luz concentrada en el centro del cono. Por lo tanto, la intensidad es una función del ángulo ϕ entre la dirección de

la fuente y un vector $\mathbf{I}_{s\phi}$ a un punto sobre la superficie (mientras este ángulo sea menos que θ), como se ve en la siguiente figura:



Aunque esta función puede definirse de varias maneras, normalmente se define mediante $\cos^e\phi$, donde el exponente *e* determina que tan rápidamente la intensidad de la luz disminuye, como se ve en la siguiente figura:



Los cosenos son por lo general funciones convenientes para cálculos de luz. Si $\mathbf{l}_{s\phi}$ (vector al punto en la superficie) y $\mathbf{l}_s$ (fuente de luz) son ambos vectores unidad, se puede computar el coseno mediante el producto punto:

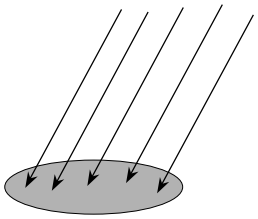
$$\cos\phi = \mathbf{l}_{s\phi} \cdot \mathbf{l}_s$$

un cálculo que requiere solo tres multiplicaciones y dos sumas.

$$I_s(p, p_0) = \frac{\cos^e \phi}{a + b|p - p_0| + c|p - p_0|^2} I(p_0)$$

Fuentes de Luz Distantes

La mayoría de los cálculos de sombreado requieren la dirección de un punto sobre la superficie a la fuente de luz. Según se mueve a lo largo de la superficie, se debe recomputar este vector para calcular la intensidad en cada punto, una computación que es una parte significativa del cálculo del sombreado. Sin embargo, si la fuente de luz está lejos de la superficie, el vector no cambiará mucho según se mueve de un punto a otro, al igual que la luz del sol da en todos los objetos cercanos entre si con el mismo ángulo. La siguiente figura ilustra que efectivamente se reemplaza una fuente de luz de punto con una fuente que ilumina objetos con rayos de luz paralelos, o sea, una fuente paralela.



En la práctica, los cálculos para fuentes de luz distantes son similares a los cálculos para proyecciones paralelas; se reemplaza la ubicación de la fuente de luz por una dirección de la fuente de luz. Por lo tanto, en coordenadas

homogéneas, una fuente de luz de punto en $\mathbf{p_0}$ se representará internamente como una matriz columna de cuatro dimensiones

$$p_0 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

donde la fuente de luz distante se representa mediante

$$p_0 = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

que es la representación de un vector.

$$I_d(p) = I_d$$

Intensidad Completa

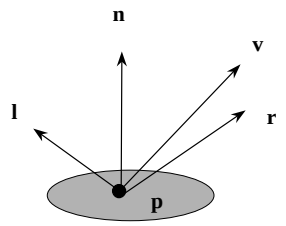
La intensidad completa exclusivamente for efectos de iluminación es la siguiente:

$$I(p) = \frac{1}{a + b|p - p_0| + c|p - p_0|^2}\left(I_p(p, p_0) + \cos^e \phi I_s(p, p_0)\right) + I_d(p) + I_a(p)$$

Modelo de Reflexión Phong

El modelo de reflexión de Phong es eficiente y suficientemente aproximado a la realidad física para producir buenas imágenes, bajo una variedad de condiciones de luz y propiedades de materiales.

El modelo usa cuatro vectores para calcular el color para un punto arbitrario ***p*** sobre la superficie, como se muestra en la siguiente figura:



Si la superficie es curva, los cuatro vectores pueden cambiar según se mueve de punto a punto.

- El vector **n** es la normal en **p**.
- El vector **v** tiene dirección de **p** al observador o COP.

- El vector **l** tiene dirección de una línea de **p** a un punto arbitrario sobre la superficie para una fuente de luz distribuida, o una fuente de luz de punto.
- El vector **r** tiene la dirección de un rayo perfectamente reflejado de **l**. La dirección de **r** está determinada por **n** y **l**.

El modelo Phong apoya los tres tipos de interacciones material-luz: ambiente, difusa y especular. Si se tiene un conjunto de fuentes puntos, con componentes independientes para cada uno de los tres colores primarios para cada uno de los tres tipos de interacciones material-luz; entonce, se puede describir la matriz de iluminación para una fuente de luz *i* para cada punto **p** sobre una superficie, mediante:

$$L_i = \begin{bmatrix} L_{ira} & L_{iga} & L_{iba} \\ L_{ird} & L_{igd} & L_{ibd} \\ L_{irs} & L_{igs} & L_{ibs} \end{bmatrix}$$

- La primera fila contiene las intensidades ambiente para rojo, verde y azul para la fuente *i*.
- La segunda fila contiene los términos difusos.
- La tercera fila contiene los términos especulares.

(Aún no se ha aplicado ninguna atenuación por la distancia.)

Se construye el modelo asumiendo que se puede computar la cantidad de cada una de las luces incidentes reflejada en el punto de interés. Por ejemplo, para el término difuso rojo de la fuente *i*, L_{ird}, se puede computar el término de reflexión, R_{ird}, y su contribución a la intensidad I_{ird} en **p** sería $I_{ird} = R_{ird} L_{ird}$. El valor de R_{ird} depende de las propiedades del material, la orientación de la superficie, la dirección de la fuente de luz y la distancia entre la fuente de luz y el observador. Para cada punto, se computa una matriz de términos de reflexión:

$$R_i = \begin{bmatrix} R_{ira} & R_{ird} & R_{irs} \\ R_{iga} & R_{igd} & R_{igs} \\ R_{iba} & R_{ibd} & R_{ibs} \end{bmatrix}$$

Se puede computar la contribución para cada fuente de color agregando los componentes de ambiente, difuso y especular. Por ejemplo, la intensidad roja vista en el punto **p** de la fuente *i* es:

$$I_{ir} = R_{ira} L_{ira} + R_{ird} L_{ird} + R_{irs} L_{irs} = I_{ira} + I_{ird} + I_{irs}$$

Se obtiene la intensidad total sumando todas las contribuciones de todas las fuentes y del ambiente global. Por lo tanto, el término rojo sería:

$$I_r = \sum_i (I_{ira} + I_{ird} + I_{irs}) + I_{ar}$$

Donde I_{ar} es el componente rojo de la luz ambiente global (sin reflexión).

Se puede simplificar la notación viendo que las computaciones son iguales para cada fuente para cada color primario. Difieren dependiendo si se considera los términos ambiente, difuso o especular. SE puede omitir los subíndices *i*, *r*, *g*, y *b*:

$$I = I_a + I_d + I_s = R_a L_a + R_d L_d + R_s L_s$$

Donde la computación será hecha para cada una de las primarias y cada fuente, y el término del ambiente global puede agregarse al final.

Reflexión Ambiente

La intensidad de la luz ambiente L_a es la misma sobre cada punto de la superficie. Parte de la luz es absorbida y parte es reflejada. La cantidad reflejada está dada por el **coeficiente de reflexión de ambiente k_a**, $R_a = k_a$. Como sólo se refleja una fracción positiva de luz, se debe tener

$$0 \le k_a \le 1$$

y por lo tanto

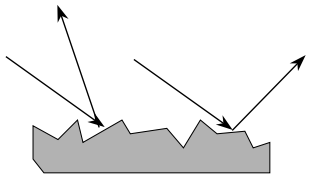
$$I_a = k_a L_a$$

Aquí, L_a puede ser cualquiera de las fuentes de luz individuales, o puede el término ambiente global.

Una superficie tiene tres coeficientes ambiente, k_{ar}, k_{ag} y k_{ab}, que pueden ser distintas. Por ejemplo, una esfera se vería amarilla bajo luz ambiente blanca si su coeficiente ambiente azul es pequeño y sus coeficientes rojo y verde son grandes.

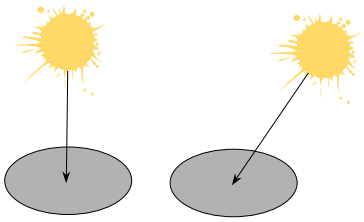
Reflexión Difusa

Un reflector difuso perfecto esparce la luz que refleja de manera igual en todas las direcciones, viéndose igual para todos los observadores. Sin embargo, la cantidad de luz reflejada depende del material, dado que parte de la luz es absorbida, y de la posición de la fuente de luz relativa a la superficie. Reflexiones difusas son caracterizadas por superficies rugosas, como se ve en la siguiente figura (corte trasversal):

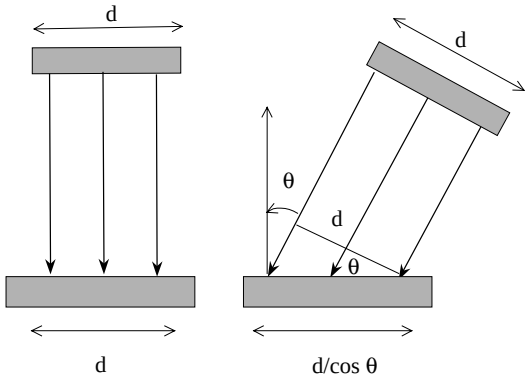


Los rayos de luz que pegan en la superficie con ángulos levemente diferentes se reflejarían en ángulos marcadamente diferentes. Superficies difusas perfectas son tan rugosas que no hay un ángulo preferido de reflexión. Tales superficies son a veces conocidas como **superficies Lambercianas**, pudiéndose modelar matemáticamente por la ley de Lambert.

Se considera una superficie plana difusa iluminada por el solo, como se muestra en la siguiente figura:



La superficie se vuelve mas brillante al mediodía, y menos durante la madrugada y la puesta, dado que, según la ley de Lambert, solo se ve el componente vertical de la luz entrante. Para comprender esta ley, se considera una fuente de luz paralela pequeña pegando en un plano, como se muestra en la siguiente figura:



Según la fuente baja en el cielo (Artificial), la misma cantidad de luz se esparce sobre una área mas grande, y la superficie parece oscurecerse.

Se puede caracterizar reflexiones difusas matemáticamente. La ley de Lambert dice que:

$$R_d \propto \cos\theta$$

Donde θ es el ángulo entre la normal **n** en el punto de interés y la dirección de la fuente de luz **l**. Si **l** y **n** son ambos vectores unidad, entonces

$$\cos\theta = \mathbf{l} \cdot \mathbf{n}$$

Si se agrega un coeficiente de reflexión k_d que representa la fracción de luz difusa entrante que es reflejada, se tiene el siguiente término de reflexión

$$I_d = k_d (\mathbf{l} \cdot \mathbf{n}) L_d \qquad 0 \le k_d \le 1$$

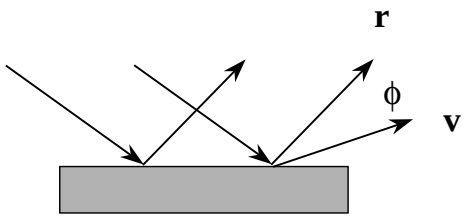
Si se desea incorporar el término de distancia, para considerar la atenuación de la luz según esta viaja una distancia d desde la fuente a la superficie, se puede agregar el término cuadrático de atenuación:

$$I_d = \frac{k_d}{a + bd + cd^2} (\mathbf{l} \cdot \mathbf{n}) L_d$$

Reflexión especular

Si se emplea solo reflexiones ambiente y difusas, las imágenes serán sombreadas y aparecerán tridimensionales, pero todas las superficies se verán sin vida. Lo que hace falta son la reflexión de secciones más brillantes en los objetos. Esto ocasiona un color diferente del color del ambiente reflejado y luz difusa. Una esfera roja, bajo luz blanca, tendrá un resplandecer blanco que es la reflexión de parte de la luz de la fuente en la dirección del observador.

Mientras que una superficie difusa es rugosa, una superficie especular es suave. Mientras mas lisa se la superficie, mas se parece a un espejo, como se ve en la siguiente figura.

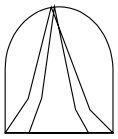


Según la superficie se hace mas lisa, la luz reflejada se concentra en un rango mas pequeño de ángulos, centrado alrededor del ángulo de un reflector perfecto: un espejo o una superficie especular perfecta. Modelar superficies especulares realísticas puede ser complejo, ya que el patrón por el cual se esparce no es simétrico, dependiendo de el largo de onda de la luz incidente y cambia con el ángulo de reflexión.

Phong propuso un modelo aproximado que puede computarse con solo un pequeño incremento en el trabajo para superficies difusas. El modelo agrega un término para reflexión especular. Se considera la superficie como rugosa para el término difuso u lisa para el término especular. La cantidad de luz que el observador ve depende del ángulo ϕ entre **r**, la dirección de un reflector perfecto, y **v**, la dirección del observador. El modelo de Phong usa la ecuación

$$I_s = k_s L_s \cos^{\alpha}\phi \qquad 0 \le k_s \le 1$$

El coeficiente k_s ($0 \le k_s \le 1$) es la fracción reflejada de la luz especular entrante. El exponente α es el coeficiente de brillantez. La siguiente figura muestra como, según se incrementa α, la luz reflejada se concentra en una región mas delgada, centrada en el ángulo de un reflector perfecto.



En el límite, según α tiende a infinito, se obtiene un espejo; valores entre 100 y 500 corresponden a la mayoría de las sueprficies metálicas, y valores menores (<100) corresponden a materiales que muestran brillantez gruesa.

La ventaja computacional del modelo de Phong es que, si se normaliza **r** y **v** a valores unitarios, se puede usar el producto punto, y el término especular se vuelve

$$I_s = k_s L_s (\mathbf{r} \cdot \mathbf{v})^{\alpha}$$

Se puede agregar el término de distancia, como se hizo con las reflexiones difusas. Finalmente, se refiere al modelo Phong, incluyendo el término de distancia, a la siguiente ecuación:

$$I = \frac{1}{a + bd + cd^2} (k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{r} \cdot \mathbf{v})^{\alpha}) + k_a L_a$$

Esta fórmula se computa para cada fuente de luz y para cada primaria.

No tiene mucho sentido asociar una cantidad diferente de luz ambiente con cada fuente, o permitir que sean diferentes los componentes de luz especulares y difusos. Como no se puede resolver la ecuación completa, se deben utilizar múltiples trucos para lograr un realismo visual. Por ejemplo, en un ambiente con muchos objetos, cuando se prende una luz, parte de la luz pegará en la superficie directamente. Estas contribuciones a la imagen pueden modelarse con componentes especulares y difusos de la fuente. Sin embargo, mucho del resto de la luz de la fuente será esparcida mediante múltiples reflexiones de otros objetos y contribuirá a la luz recibida en la superficie bajo consideración. Se puede aproximar este término teniendo un componente de ambiente asociado con la fuente. El sombreado que se debe asignar a este término depende del color de la fuente y el color de los objetos en el cuarto, una consecuencia desafortunada del uso de modelos aproximados. Hasta cierto punto, el mismo análisis se usa para

luz difusa. Luz difusa se refleja entre superficies, y el color visto en una superficie particular depende de las demás superficies en el ambiente. Al usar componentes difusos y especulares separados con las fuentes de luz, se puede tratar de aproximar un efecto global con cálculos locales.

El modelo de Phong se ha hecho en espacio de objetos. El sombreado, sin embargo, no se hace hasta que los objetos hayan pasado por las transformaciones modelo-vista y proyección. Estas transformaciones pueden afectar los términos de coseno en el modelo.

Computación de Vectores

Los modelos de iluminación y reflexión derivados son lo suficientemente generales para aplicarse a superficies curvas o planas, vistas paralelas o perspectivas, y a superficies distantes o cercanas. La mayoría de los cálculos para generar una escena involucran la determinación de los vectores requeridos y productos punto. Para cada caso especial, son posibles las simplificaciones. Por ejemplo, si la superficie es un polígono plano, la normal es la misma en todos los puntos sobre la superficie. Si la fuente de luz está lo suficientemente lejos de la superficie, la dirección de la luz es la misma para todos los puntos.

Primero se examinará la computación de vectores para el caso general, y luego se verá técnicas adicionales para superficies planas.

Vectores Normales

Para superficies lisas, el vector normal a la superficie existe en cada punto y da la orientación local. Su cálculo depende de la representación matemática de la superficie. Dos casos sencillos, el plano y la esfera, ilustran como computar normales y donde aparecen dificultades.

Un plano puede describirse por la ecuación

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

Esta ecuación también puede escribirse en término de la normal al plano **n**, y un punto $\mathbf{p_0}$, que esté sobre el plano

$$\mathbf{n} \cdot (\mathbf{p} - \mathbf{p_0}) = 0$$

donde **p** es cualquier punto (x,y,z) sobre el plano. Comparando las dos formas, se ve que el vector **n** está dado en coordenadas homogéneas por

$$n = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix}$$

Sin embargo, si en lugar se dan tres puntos no colineares, $\mathbf{p_0}$, $\mathbf{p_1}$ y $\mathbf{p_2}$, que están sobre el plano y son suficientes para determinarlo de manera única. Los vectores $\mathbf{p_2} - \mathbf{p_0}$ y $\mathbf{p_1} - \mathbf{p_0}$ son paralelos al plano, y se puede usar su producto cruz para encontrar la normal

$$\mathbf{n} = (\mathbf{p_2} - \mathbf{p_0}) \times (\mathbf{p_1} - \mathbf{p_0})$$

Se debe tener cuidado en el orden de multiplicación de los vectores en el producto cruz. Revirtiendo el orden cambia la superficie de apuntar afuera a apuntar adentro, y eso puede afectar el cálculo de de luz. Algunos sistemas gráficos utilizan los tres primeros vértices en la especificación de un polígono para determinar la normal automáticamente. OpenGL, no lo hace, pero forzando al usuario a computar normales crea mayor flexibilidad en como se aplica el modelo de luz.

Para superficies curvas, el cómputo de normales depende en la representación de la superficie. Se puede ver algunas posibilidades considerando como representar una esfera unitaria centrada en el origen. La especificación básica para una esfera con radio 1 se da mediante la **ecuación implícita**

$$f(x,y,z) = x^2 + y^2 + z^2 - 1 = 0$$

o en forma vectorial

$$f(\mathbf{p}) = \mathbf{p} \cdot \mathbf{p} - 1 = 0$$

($\mathbf{p} \cdot \mathbf{p} = 1$, ya que cos0 = 1, y los vectores son unitarios)

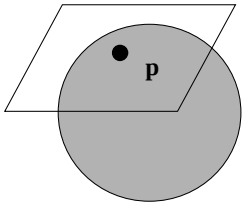
La normal está dada por el **vector gradiente**, representado por la matriz columna

$$n = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \\ 2z \end{bmatrix} = 2p$$

La esfera también puede representarse de **forma paramétrica** (la mas utilizada en la gráfica por computadora), donde los valores *x*, *y*, *z* de un punto sobre la esfera se representan independientemente en término de dos parámetros *u* y *v*:

$x = x(u, v) = \cos u \sin v$
$y = y(u, v) = \cos u \cos v$
$z = z(u, v) = \sin u$

Según *u* y *v* varían en el rango $-\pi/2 < u < \pi/2$, $-\pi < v < \pi$, se obtienen todos los puntos sobre la esfera. Cuando se usa la forma paramétrica, se puede obtener la normal mediante el plano tangente, mostrado en la siguiente figura, en el punto $\mathbf{p}(u, v) = [x(u, v) \;\; y(u, v) \;\; z(u, v)]^T$ sobre la superficie:



El plano tangente da la orientación local de la superficie en un punto. Se puede derivar tomando los términos lineares de la serie de expansión de Taylor de la superficie en **p**. El resultado es, en **p**, líneas en las direcciones de los vectores representados por

$$\frac{\partial p}{\partial u} = \begin{bmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial u} \end{bmatrix}, \quad \frac{\partial p}{\partial v} = \begin{bmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial v} \end{bmatrix}$$

yacen en el plano tangente. Se puede usar su producto cruz para obtener la normal

$$n = \frac{\partial p}{\partial u} \times \frac{\partial p}{\partial v}$$

Para la esfera unitaria

$$n = \cos u \begin{bmatrix} \cos u \sin v \\ \cos u \cos v \\ \sin v \end{bmatrix} = (\cos u) p$$

Solo interesa la dirección **n**; por lo cual se puede dividir por cos*u* para obtener la normal unitaria de la esfera

n = **p**

Dentro de un sistema gráfico, normalmente se trabaja con un conjunto de vértices, y el vector normal debe aproximarse de algún conjunto de puntos cercano al punto donde se necesita la normal. LA arquitectura de tubería de sistemas gráficos de tiempo real hace este cálculo difícil, porque se procesa un vértice a la vez, y el sistema gráfico puede que no tenga la información lista para computar la aproximación de la normal en un punto dado. Por lo tanto, los sistemas gráficos a menudo dejan la computación de normales al programa de usuario.

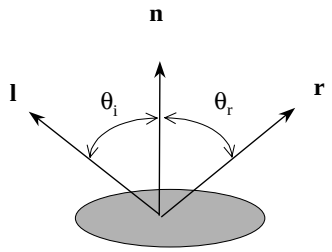
En OpenGL, se puede asociar una normal con un vértice mediante funciones como

```
glNormal3f(nx, ny, nz);
glNormal3fv(pointer_to_normal);
```

Si se define una normal antes de una secuencia de vértices mediante llamadas `glVertex`, la normal será asociada con todos los vértices, y será usada para los cálculos de luz sobre todos los vértices.

Angulo de Reflexión

Una vez calculada la normal en un punto, se puede utilizar la normal y la dirección de la fuente de luz para computar la dirección de una reflexión perfecta. Un espejo ideal se caracteriza en que *el ángulo de incidencia es igual al ángulo de reflexión*, como se ve en la siguiente figura.



- El **ángulo de incidencia** es el ángulo entre la normal y la fuente de luz (asumiendo que es una fuente de punto).
- El **ángulo de reflexión** es el ángulo entre la normal y la dirección en que la luz se refleja.

En dos dimensiones, existe un solo ángulo satisfaciendo la condición del ángulo.
En tres dimensiones, existen un infinito número de ángulos satisfaciendo la condición, por lo cual debe agregarse:
En un punto **p** *sobre la superficie, el rayo de luz entrante, el rayo de luz reflejado, y la normal en el punto deben todos yacer sobre el mismo plano.*
Estas dos condiciones son suficientes para determinar **r** de **n** y **l**. Para simplificar los cálculos se supone que los vectores **l** y **n** están normalizados, además de el vector resultante **r**

| **l** | = | **n** | = | **r** | = 1

Si $\theta_i = \theta_r$ entonces

$$\cos\theta_i = \cos\ \theta_r$$

Usando el producto punto, la condición del ángulo es

$$\cos\theta_i = \mathbf{l} \cdot \mathbf{n} = \cos\ \theta_r = \mathbf{n} \cdot \mathbf{r}$$

La condición coplana implica que **r** se puede escribir como una combinación lineal de **l** y **n**

$$\mathbf{r} = \alpha\mathbf{l} + \beta\mathbf{n}$$

Tomando el producto punto con **n**, se encuentra que

(1) $$\mathbf{n} \cdot \mathbf{r} = \alpha\mathbf{l} \cdot \mathbf{n} + \beta = \mathbf{l} \cdot \mathbf{n}$$

Se obtiene una segunda condición entre α y β del requisito que **r** sea un vector unitario

(2) $$1 = \mathbf{r} \cdot \mathbf{r} = \alpha^2 + 2\alpha\beta\,\mathbf{l} \cdot \mathbf{n} + \beta^2$$

Resolviendo estas dos ecuaciones se obtiene, de (1)

$$\beta = \mathbf{l} \cdot \mathbf{n} - \alpha\mathbf{l} \cdot \mathbf{n} = (1-\alpha)\,\mathbf{l} \cdot \mathbf{n}$$

reemplazando en (2)

$$\begin{aligned}\alpha^2 + 2\alpha\beta\,\mathbf{l} \cdot \mathbf{n} + \beta^2 &= \alpha^2 + 2\alpha(1-\alpha)(\mathbf{l} \cdot \mathbf{n})^2 + (1-\alpha)^2\,(\mathbf{l} \cdot \mathbf{n})^2 = 1\\ &= \alpha^2 + (2\alpha + (1-\alpha))(1-\alpha)(\mathbf{l} \cdot \mathbf{n})^2 = \alpha^2 + (1+\alpha)(1-\alpha)(\mathbf{l} \cdot \mathbf{n})^2\\ &= \alpha^2 + (1-\alpha^2)(\mathbf{l} \cdot \mathbf{n})^2 = \alpha^2(1-(\mathbf{l} \cdot \mathbf{n})^2) + (\mathbf{l} \cdot \mathbf{n})^2 = 1\end{aligned}$$

o

$$\alpha^2\,(1-(\mathbf{l} \cdot \mathbf{n})^2) = 1-\,(\mathbf{l} \cdot \mathbf{n})^2$$

Resolviendo para α se obtiene

$$\alpha^2 = 1$$

Si $\alpha = 1$, entonces, de (1), $\beta = 0$, lo que lleva a la identidad.
Si $\alpha = -1$, entonces, de (1), $\beta = 2(\mathbf{l} \cdot \mathbf{n})$, por lo cual de

$$\mathbf{r} = \alpha\mathbf{l} + \beta\mathbf{n}$$

se obtiene

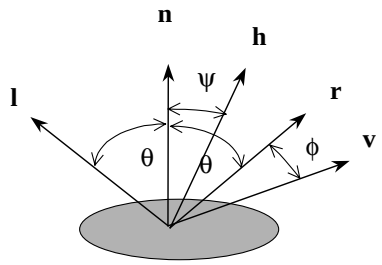
$$\mathbf{r} = 2(\,\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$

Vector Medio

Si se usa el modelo de Phong con reflexiones especulares, el producto **r** · **v** debe recalcularse en cada punto sobre la superficie. Se puede obtener una buena aproximación usando el vector unitario medio entre el vector del observador y el de la fuente de luz

$$h = \frac{l + v}{|l + v|}$$

La siguiente figura muestra los cinco vectores



Se define el ángulo ψ como el ángulo entre **v** y **h**, el **ángulo medio**. Cuando **v** yace en el mismo plano que **l**, **n**, y **r**, se obtiene:

$$\theta + \psi = \theta - \psi + \phi$$

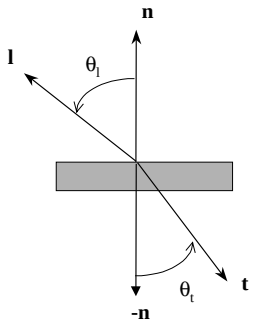
obteniéndose

$$2\psi = \phi$$

Si se reemplaza **r** · **v** con **n** · **h**, se evita calcular **r**. Sin embargo, el ángulo ψ es mas pequeño que ϕ, y, si se usa el mismo exponente *e* en $(\mathbf{n} \cdot \mathbf{h})^e$ usado en $(\mathbf{r} \cdot \mathbf{v})^e$, el tamaño de los reflejos especulares será menor. Se puede mejorar el problema reemplazando el valor del exponente *e* con el valor *e*', para que $(\mathbf{n} \cdot \mathbf{h})^{e'}$ se acerque mas a $(\mathbf{r} \cdot \mathbf{v})^e$. Claramente, evitar la recomputación de **r** es una ventaja.

Luz Transmitida

Cálculos de sombreado similares se aplican si el modelo permite la transmisión de luz. Se considera una superficie que transmite toda la luz que le da, como se ve en la siguiente figura.



Si la velocidad de la luz difiere en los dos materiales, la luz se curva en la superficie. Si η_l y η_t son los índices de refracción, una medida de la velocidad relativa de la luz en los dos materiales de los dos lados de la superficie. La **ley de Snell** dice que

$$\frac{\sin\theta_l}{\sin\theta_t} = \frac{\eta_t}{\eta_l}$$

Se puede encontrar la dirección de la luz transmitida **t** de la siguiente forma. Se conoce $\cos\theta_l$ de **n** y **l**; si estos vectores están normalizados, es simplemente su producto punto. Dado $\cos\theta_t$, se puede encontrar $\sin\theta_t$, y luego $\sin\theta_l$. Si $\eta = \eta_t / \eta_l$

$$\frac{\sin^2\theta_l}{\sin^2\theta_t} = \eta^2$$

$$\sin^2\theta_l = \left(1-\cos^2\theta_l\right) = \eta^2\sin^2\theta_t = \eta^2\left(1-\cos^2\theta_t\right)$$

$$\left(1-\cos^2\theta_t\right) = \frac{1}{\eta^2}\left(1-\cos^2\theta_l\right)$$

$$\cos\theta_t = \left(1-\frac{1}{\eta^2}\left(1-\cos^2\theta_l\right)\right)^{\frac{1}{2}}$$

Al igual que con la luz reflejada, los tres vectores deben ser coplanos

(1) $\mathbf{t} = \alpha\mathbf{n} + \beta\mathbf{l}$

Si además se impone la condición que **t** es un vector unitario, se puede hacer una derivación similar a la del vector reflejado, para encontrar

(2) $1 = \mathbf{t}\cdot\mathbf{t} = \alpha^2 + 2\alpha\beta\,\mathbf{l}\cdot\mathbf{n} + \beta^2$

Resolviendo estas dos ecuaciones se obtiene, de (1)

$$\mathbf{n}\cdot\mathbf{t} = \alpha + \beta\,\mathbf{l}\cdot\mathbf{n}$$

$$\alpha = \mathbf{n}\cdot\mathbf{t} - \beta\,\mathbf{l}\cdot\mathbf{n}$$

De la ecuación (2)

$$\begin{aligned}(\mathbf{n}\cdot\mathbf{t} - \beta\,\mathbf{l}\cdot\mathbf{n})^2 + 2(\mathbf{n}\cdot\mathbf{t} - \beta\,\mathbf{l}\cdot\mathbf{n})\beta\,\mathbf{l}\cdot\mathbf{n} + \beta^2 &= 1\\ = (\mathbf{n}\cdot\mathbf{t})^2 - 2\beta(\mathbf{n}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{n}) - \beta^2(\mathbf{l}\cdot\mathbf{n})^2 + 2\beta(\mathbf{n}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{n}) - 2\beta^2(\mathbf{l}\cdot\mathbf{n})^2 + \beta^2&\\ = (\mathbf{n}\cdot\mathbf{t})^2 - \beta^2(\mathbf{l}\cdot\mathbf{n})^2 + \beta^2&\\ = (\mathbf{n}\cdot\mathbf{t})^2 + \beta^2(1 - (\mathbf{l}\cdot\mathbf{n})^2) = 1&\end{aligned}$$

$\beta^2 = (1 - (\mathbf{n}\cdot\mathbf{t})^2)/(1 - (\mathbf{l}\cdot\mathbf{n})^2) = (1 - \cos^2\theta_t)/(1 - \cos^2\theta_l) = \sin^2\theta_t/\sin^2\theta_l = 1/\eta^2$

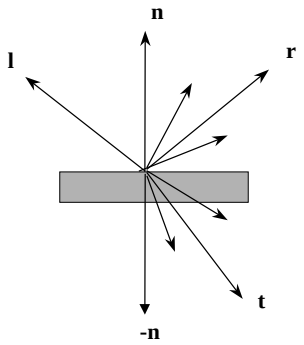
$\beta = 1/\eta$

$\alpha = \mathbf{n}\cdot\mathbf{t} - \beta\,\mathbf{l}\cdot\mathbf{n} = \cos\theta_t - \cos\theta_l/\eta$

$$t = -\frac{1}{\eta}l - \left(\cos\theta_t - \frac{1}{\eta}\cos\theta_l\right)n$$

Los primeros dos signos negativos en la ecuación son una consecuencia del hecho que **t** apunta alejándose del lado de atrás de la superficie. El ángulo para el cual el término de la raíz cuadrada en la expresión para $\cos\theta_t$ se vuelve cero ($\sin\theta_l = \eta$) se llama el **ángulo crítico**. Si la luz pega en la superficie en este ángulo, la luz transmitida es en la dirección de la superficie. Si θ_l se incrementa mas, toda la luz se refleja, y nada se transmite.

Una expresión mas general para lo que ocurre en una superficie de transmisión se ve en la siguiente figura.



Parte de la luz se transmite, parte se refleja, y el resto se absorbe. De la luz transmitida, parte se esparce de manera similar a las reflexiones especulares, excepto que aquí la luz está concentrada en la dirección de **t**. Por lo tanto, un modelo de transmisión podría incluir un término proporcional a $\mathbf{t} \cdot \mathbf{v}$ para observadores del lado de transmisión de la superficie. También se puede usar la analogía del ángulo medio para simplificar el cálculo de este término.

Sombreado Poligonal

Asumiendo que se puede computar vectores normales, dado un conjunto de fuentes de luz y un observador, los modelos de luz e iluminación desarrollados pueden aplicarse a cada punto de una superficie. Lamentablemente, aunque se tenga ecuaciones sencillas para determinarlos vectores normales, como en el ejemplo de la esfera, la cantidad de computaciones requeridas puede ser muy grande. SE han visto muchas de las ventajas de usar modelos poligonales para los objetos. Una ventaja adicional, para polígonos planos, es que se puede reducir bastante el trabajo requerido para el sombreado. LA mayoría de los sistemas gráficos, incluyendo OpenGL, explota las posibles eficiencias para polígonos planos, descomponiendo superficies curvas en muchos polígonos planos pequeños, cada uno teniendo un vector normal bien definido.

Se considerarán tres maneras de sombrear polígonos:
- Sombreado plano o constante
- Sombreado interpolativo o Gouraud
- Sombreado Phong

Sombreado Plano (Flat)

Los tres vectores, **l**, **n** y **v**, pueden variar según se va entre puntos sobre una superficie.
- Para un polígono plano, **n** es constante.
- Si se asume un observador distante (la bandera `GL_DISTANT_VIEWER` en OpenGL), **v** es constante sobre el polígono.
- Si la fuente de luz es distante, **l** es constante.

Distante se puede interpretar como una fuente o un observador en el infinito; en particular, si los polígonos son pequeños, las distancias relativas para el infinito no tienen que ser muy grandes. Los ajustes necesarios, como cambiar la *ubicación* de la fuente u observador a la *dirección* de la fuente u observador, de forma correspondiente, puede hacerse a las ecuaciones de sombreado y a su implementación.

Si los tres vectores son constantes, entonces el cálculo de sombreado se lleva a cabo una sola vez para cada polígono, y se asignará la misma sombra a cada punto en el polígono. Esta técnica se conoce como **sombreado plano** o **constante**. En OpenGL, se especifica sombreado plano mediante

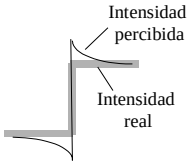
```
glShadeModel(GL_FLAT);
```

Si se usa sombreado plano, OpenGL usará las normales asociadas con el primer vértice de un polígono

Sencillo para el cálculo de sombreado. Para primitivas como un strip de triángulo, OpenGL usa la normal del tercer vértice para el primer triángulo, la normal del cuarto vértice para el segundo, etc. Reglas similares se aplican para otras primitivas, como strips cuadriláteros.

Sombreado plano mostrará diferencias de sombreado entre los polígonos. Si las fuentes de luz y el observador están cerca del polígono, los vectores **v** y **l** serán diferentes para cada polígono. Sin embargo, si los polígonos se diseñaron para modelar una superficie suave, sombreado plano no será el mas apropiado, ya que se verán aunque sea diferencias de sombreado pequeñas entre polígonos adyacentes, como se ve en la siguiente figura

El sistema visual humano tiene una sensibilidad remarcada para pequeñas diferencias de intensidad de color, causada por la propiedad de **inhibición lateral**. Si se ve una secuencia incremental de intensidades, se percibe los incrementos de brillantez reforzados en un lado del intervalo de intensidad y decremento de brillantez reforzado del otro lado, como se ve en la siguiente figura



Se ven bandas, llamadas **bandas de Mach**, a los largo de los bordes. Este fenómeno es una consecuencia de como se conectan los conos al nervio óptico en el ojo, y no hay nada que se pueda hacer al respecto, fuera de buscar técnicas de sombreado mas suaves que no produzcan grandes diferencias en sombreado en los bordes de los polígonos.

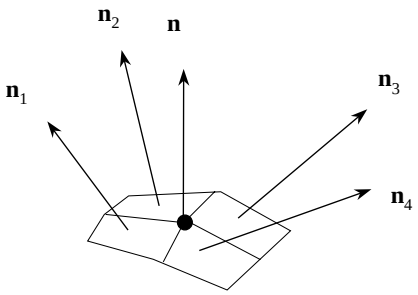
Sobreado Interpolativo y Gouraud

En el ejemplo del cubo rotante, se vio que OpenGL interpola colores asignados a los vértices a través del polígono. Si se asigna el modelo de sombreado para que sea suave, mediante

```
glShadeModel(GL_SMOOTH);
```

OpenGL interpolará colores para las primitivas, como las líneas. Si se permite sombreado y luz suaves, y se asigna a cada vértice la normal del polígono siendo sombreado, se calcularía la luz en cada vértice, determinando el color del vértice, usando las propiedades materiales y los vectores **v** y **l** se computan para cada vértice. Si la fuente de luz es distante, y el observador es distante o no hay reflexiones especulares, entonces el sombreado interpolativo sombreara un polígono con un color constante.

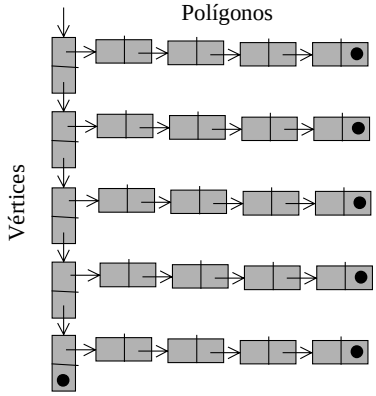
Como múltiples polígonos se juntan en vértices interiores, cada uno teniendo su propia normal, la normal en los vértices es discontinua. Aunque esta situación pudiera complicar las matemáticas, Gouraud se dio cuenta que la normal en el vértice de puede *definir* de manera que se obtenga sombras mas suaves mediante interpolación. Si se tiene un vértice interior donde cuatro polígonos se juntan, cada uno con su normal, como se ve en la siguiente figura



En el **sombreado Gouraud**, se define la normal en un vértice como el promedio normalizado de las normales de los polígonos que comparten el vértice. En este ejemplo, la **normal del vértice** está dada por

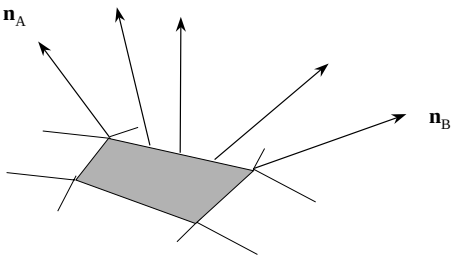
$$n = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$$

Desde la perspectiva de OpenGL, el sombreado Gouraud es deceptivamente sencillo. Se necesita solamente asignar correctamente las normales de vértices. La literatura a menudo no distingue entre el sombreado interpolativo y el Gouraud. Sin embargo, existe un problema, encontrar las normales para promediar. Si el programa es lineal, especificar una lista de vértices (y otras propiedades), no se tiene la información necesaria de cuales polígonos comparten un vértice. Lo que se requiere es una estructura de datos para representar un "mesh". Atravesar esta estructura de datos puede generar los vértices con las normales promediadas. Tal estructura de datos debe contener, de manera mínima, polígonos, vértices, normales, y propiedades de los materiales. Una posible estructura se muestra en al siguiente figura, teniendo como elemento clave los polígonos que se juntan en cada vértice

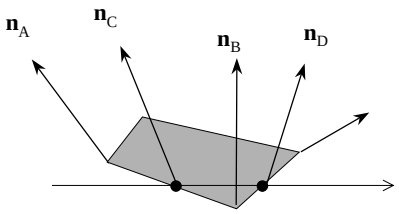


Sombreado Phong

Incluso la suavidad introducida por el sombreado Gouraud no puede prevenir la apariencia de bandas Mach. Phong propuso que, en lugar de interpolar intensidades de los vértices, según se hace en el sombreado Gouraud, se interpole normales a lo largo del polígono. Para un polígono que comparte lados y vértices con otros polígonos, como se ve en la siguiente figura.



Se puede computar normales en los vértices interpolando sobre las normales de los polígonos que comparten el vértice. Luego, se puede usar interpolación bilineal, para interpolar las normales sobre el polígono, como se ve en la siguiente figura



Se puede usar las normales interpoladas en los vértices *A* y *B* para interpolar a lo largo de lado entre ellas:

$$\mathbf{n}(\alpha) = (1- \alpha)\mathbf{n}_A + \alpha\, \mathbf{n}_B$$

Se puede hacer una interpolación similar en todos los lados. LA normal en cualquier punto interior se puede obtener de los puntos en los lados mediante

$$\mathbf{n}(\alpha,\beta) = (1- \beta)\mathbf{n}_C + \beta\, \mathbf{n}_D$$

Una vez obtenidas las normales en cada punto, se puede hacer cálculos de sombreado independientes. Normalmente, este proceso se combinará con la rasterización del polígono, para que la línea entre *C* y *D* proyecte a una línea de rastreo en el frame buffer.

Sombreado Phong producirá imágenes mas suaves que con el sombreado Gouraud, pero a un costo computacional mayor. Existen varias implementaciones en hardware para el sombreado Gouraud, pero no así mismo para sombreado Phong.

Aproximación de una esfera por subdivisión recursiva

La esfera es un ejemplo de superficie curva. Se muestra aquí una aproximación poligonal a una esfera, sirviendo como base para escribir programas sencillos que ilustran las interacciones entre los parámetros de sombreado y aproximaciones poligonales a superficies curvas. Para esto se utiliza la **subdivisión recursiva**, una poderosa técnica para generar aproximaciones a curvas y superficies a cualquier nivel de exactitud. (Aunque OpenGL no apoya la construcción de una esfera, las librerías GLU y GLUT si las incluyen, construidas mediante superficies cuadráticas y aproximaciones poligonales, respectivamente.)

El punto de comienzo es el tetrahedro, aunque se puede comenzar con cualquier polígono regular cuyas facetas se puedan dividir inicialmente en triángulos. El tetrahedro regular está compuesto de cuatro triángulos equiláteros, determinado por cuatro vértices. Se comienza aquí con los siguientes cuatro vértices: (0,0,1), (0,$2\sqrt{2}/3$,-1/3), (-$\sqrt{6}/3$, -$\sqrt{2}/3$,-1/3), ($\sqrt{6}/3$,-$\sqrt{2}/3$,-1/3). Los cuatro puntos yacen en un círculo unitario, centrado en el origen.

Se obtiene la primera aproximación dibujando un marco alambrado para el tetrahedro. Se utiliza el tipo punto, definido anteriormente, para definir los cuatros vértices globales:

```
point v[4]={{0.0, 0.0, 1.0}, {0.0, 0.942809, -0.33333},
      {-0.816497, -0.471405, -0.333333},
      {0.816497, -0.471405, 0.333333}};
```

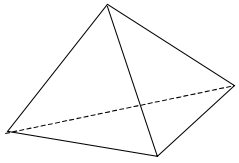
Se puede dibujar triángulos mediante la función

```
void triangle( point a, point b, point c)
{
      glBegin(GL_LINE_LOOP);
            glVertex3fv(a);
            glVertex3fv(b);
            glVertex3fv(c);
      glEnd();
}
```

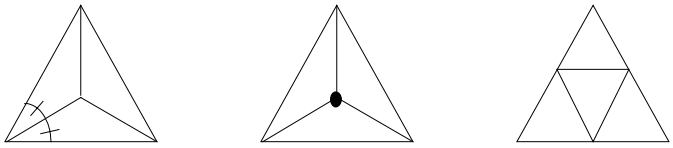
El tretrahedro se puede dibujar mediante

```
void tetrahedron(void)
{
      triangle(v[0], v[1], v[2]);
      triangle(v[3], v[2], v[1]);
      triangle(v[0], v[3], v[1]);
      triangle(v[0], v[2], v[3]);
}
```

El orden de los vértices obedece a la regla de mano derecha, por lo cual se puede convertir para dibujar polígonos fácilmente. Si se añade el código típico de inicialización, el programa generaría una imagen como en la siguiente figura; un poliedro regular simple, pero una pobre aproximación a una esfera.



Se puede obtener una aproximación mas cercana a una esfera subdividiendo cada faceta del tetrahedro en triángulos mas pequeños. Subdividiendo en triángulos asegura que las nuevas facetas sean planas. Existen al menos tres formas de subdivisión, como se muestra en la siguiente figura.



Bisección de ángulos Cómputo del centrum Bisección de lados

- Se puede bisectar cada ángulo del triángulo, y luego dibujar los tres bisectores, los cuales se encuentran en un punto común, generando de esta forma tres nuevos triángulos.
- Se puede computar el centro de masa (centrum) de los vértices simplemente promediándolos, y luego dibujando líneas de este punto a los tres vértices, generando de nuevo tres triángulos.

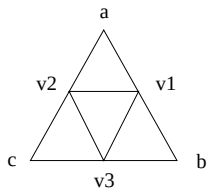
Sin embargo, estas técnicas no preservan los triángulos equiláteros que constituyen los tetrahedros regulares.

- En su lugar, recordando una posible construcción del gasket de Sierpinski, se puede conectar los bisectores de los lados del triángulo, formando cuatro triángulos equiláteros. Esta será la técnica utilizada aquí.

Después de subdividir una faceta mediante la bisección de lados, los cuatro nuevos triángulos estarán aún en el mismo plano que el triángulo original. Se puede mover los nuevos vértices creados por la bisección a la esfera unitaria al normalizar cada vértice bisectado, usando una función de normalización sencilla, como

```
void normal(point p)
{
    double sqrt();
    float d =0.0;
    int i;
    for(i=0; i<3; i++) d+=p[i]*p[i];
    d=sqrt(d);
    if(d>0.0) for(i=0; i<3; i++) p[i]/=d;
}
```

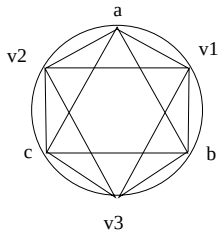
Ahora se puede subdividir un solo triángulo, definido por lo vértices numerados a, b, y c, como se muestra en la siguiente figura,



mediante el código

```
point v1, v2, v3;
int j;
for(j=0; j<3; j++) v1[j]=a[j]+b[j];
normal(v1);
for(j=0; j<3; j++) v2[j]=a[j]+c[j];
normal(v2);
for(j=0; j<3; j++) v3[j]=b[j]+c[j];
normal(v3);
triangle(a, v1, v2);
triangle(c, v2, v3);
triangle(b, v3, v1);
triangle(v1, v3, v2);
```

El resultado se muestra en la siguiente figura



Se puede usar este código en la rutina del tetrahedro para generar 16 triángulos en lugar de cuatro, pero sería mejor poder repetir el proceso de subdivisión *n* veces para generar sucesivamente aproximaciones mas cercanas a la esfera. Llamando la rutina de subdivisión recursivamente, se puede controlar el número de subdivisiones.

Primero, se hace que la rutina del tetrahedro dependa de la profundidad de recursión añadiendo el argumento *m*

```
void tetrahedron(int m)
{
```

```
        divide_triangle(v[0], v[1], v[2], m);
        divide_triangle(v[3], v[2], v[1], m);
        divide_triangle(v[0], v[3], v[1], m);
        divide_triangle(v[0], v[2], v[3], m);
}
```

La función `divide_triangle` se llamará a si misma para subdividir adicionalmente si *m* es mayor que cero, pero generará triángulos si *m* ha sido reducida a cero. El código es el siguiente

```
void divide_triangle(point a, point b, point c, int m)
{
    point v1, v2, v3;
    int j;
    if(m>0)
    {
        for(j=0; j<3; j++) v1[j]=a[j]+b[j];
        normal(v1);
        for(j=0; j<3; j++) v2[j]=a[j]+c[j];
        normal(v2);
        for(j=0; j<3; j++) v3[j]=b[j]+c[j];
        normal(v3);
        divide_triangle(a, v1, v2, m-1);
        divide_triangle(c, v2, v3, m-1);
        divide_triangle(b, v3, v1, m-1);
        divide_triangle(v1, v3, v2, m-1);
    }
    else(triangle(a,b,c));
}
```

Fuentes de Luz en OpenGL

OpenGL apoya los cuatro tipos de fuentes de luz descritas, y permite al menos ocho fuentes de luz en un programa. Cada una debe ser especificada y habilitada individualmente. Aunque se tiene que especificar muchos parámetros, son exactamente los parámetros requeridos por el modelo Phong. Las funciones de OpenGL

```
glLightfv(source, parameter, pointer_to_array);
glLightf(source, parameter, value);
```

permiten asignar los parámetros de ofrma vectorial o escalar, respectivamente. Existen cuatro parámetros vectorailes que se pueden asignar:

- posición (o dirección) de la fuente de luz
- cantidad de luz ambiente asociada con la fuente
- cantidad de luz difusa asociada con la fuente
- cantidad de luz especular asociada con la fuente

Por ejemplo, si se quiere especificar la primera fuente de luz `GL_LIGHT0`, y ubicarla en el punto (1.0,2.0,3.0), se guardaría su posición como un punto en coordenadas homogéneas:

```
GLfloat light_0_pos[]={1.0, 2.0, 3.0, 1.0};
```

Con el cuarto componente asignado a cero, la fuente de punto se vuelve una fuente distante con una dirección de vector

```
GLfloat light_0_dir[]={1.0, 2.0, 3.0, 0.0};
```

Para una sola fuente de luz, si se quiere un componente especular blanco, y componentes ambiente y difuso rojo, se tendría el código

```
GLfloat light_0_pos[]={1.0, 2.0, 3.0, 1.0};
```

```
GLfloat light_0_ambient[]={1.0, 0.0, 0.0, 1.0};
GLfloat light_0_diffuse[]={1.0, 1.0, 1.0, 1.0};
GLfloat light_0_specular[]={1.0, 1.0, 1.0, 1.0};

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

glLightfv(GL_LIGHT0, GL_POSITION, light_0_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_0_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_0_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_0_specular);
```

Nótese que se debe habilitar iluminación y la fuente de luz particular. El último elemento de los distintos componentes, es el factor alfa.

También se puede agregar un término de ambiente global que es independiente de cualquiera de las fuentes. Por ejemplo, si se quiere agregar una pequeña cantidad de luz blanca, se usaría el siguiente código

```
GLfloat global_ambient[]={0.1, 0.1, 0.1, 1.0};

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
```

Los términos de distancia se basan en el modelo de atenuación de distancia:

$$f(d) = \frac{1}{a + bd + cd^2}$$

el cual contiene términos constante, lineal y cuadrático. Estos términos se asignan mediante glLightf; por ejemplo

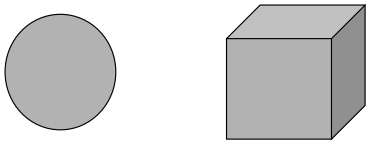
```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
```

Se puede convertir una fuente pòsicional a un spotlight al escoger la dirección del spotlight (`GL_SPOT_DIRECTION`), el exponente (`GL_SPOT_EXPONENT`), y el ángulo (`GL_SPOT_CUTOFF`). Los tres se pueden especificar mediante `glLightf` y `glLightfv`.

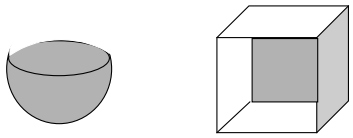
Existen otros dos parámetros de luz provistos por OpenGL que vale la pena mencionar: `GL_LIGHT_MODEL_LOCAL_VIEWER` y `GL_LIGHT_MODEL_TWO_SIDED`. Los cálculos de iluminación pueden consumir mucho tiempo. Si se asume que el observador está a una distancia infinita de la escena, entonces el cálculo de reflexiones es más fácil, porque la dirección al observador de cualquier punto en la escena no cambia. La omisión en OpenGL es hacer esta aproximación, ya que su efecto en muchas escenas es mínimo. Si se prefiere que se hagan los cálculos completos de luz, usando la posición real del observador, se cambiaría el modelo usando

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
```

Anteriormente se ha visto que una superficie tiene una cara frontal y otra posterior. Para polígonos, se determina el frente y atrás según el orden en que se especifican los vértices, utilizando la regla de mano derecha. Para la mayoría de los objetos, solo se ven las caras frontales, por lo cual no es de interés como OpenGL sombrea las superficies mirando hacia atrás. Por ejemplo, para objetos convexos, como una esfera o un paralelepípedo, como se muestra en la siguiente figura, el observador nunca ve la cara de atrás, sin importar donde está posicionado.



Sin embargo, si se quita un lado del cubo, o se rebana la esfera, como se ve en la siguiente figura, un observador ubicado apropiadamente pudiera ver una cara de atrás, por lo cual se debe sombrear las caras de adelante y atrás correctamente.



Se puede asegurar que OpenGL maneje ambas caras correctamente invocando la función

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDED, GL_TRUE);
```

Fuentes de luz son objetos, al igual que polígonos y puntos. Por lo tanto, las fuentes de luz están afectadas por las transformaciones modelo-vista de OpenGL. Se les puede definir en la posición deseada, o definirlas en una posición conveniente y moverlas a la posición deseada mediante transformaciones modelo-vista. La regla básica gobernando la ubicación de objetos es que los vértices se convierten en coordenadas de ojo mediante la transformación modelo-vista que está en efecto en el momento que los vértices son definidos. Por lo tanto, mediante una ubicación cuidadosa de las especificaciones de las fuentes de luz relativas a la definición de los demás objetos geométricos, se puede crear

- fuentes de luz que se mantengan estacionarias mientras los objetos se mueven
- fuentes de luz que se mueven mientras los objetos se mantienen estacionarios, y
- fuentes de luz que se mueven con los objetos.

Especificación de Materiales en OpenGL

Las propiedades materiales en OpenGL corresponden directamente con las fuentes de luz apoyadas y con el modelo de reflexión de Phong. También se puede especificar propiedades diferentes de mateirales para las caras frontales y posteriores de una superficie. Todos los parámetros de reflexión se especifican mediante las funciones

```
glMaterialfv(face, type, pointer_to_array);
glMaterialf(face, type, value);
```

Por ejemplo, se pudiera definir coeficientes de reflectividad ambiente, difuso y especular (k_a, k_d, k_s) para cada uno de los colores primarios de tres formas

```
GLfloat mat_ambient[]={0.2, 0.2, 0.2, 1.0};
GLfloat mat_diffuse[]={1.0, 0.8, 0.0, 1.0};
GLfloat mat_specular[]={1.0, 1.0, 1.0, 1.0};
```

Aquí, se definió una pequeña cantidad de reflectividad ambiente blanca, propiedades difusas amarillas, y reflexiones especulares blancas. Se asignan las propiedades materiales para las caras frontales y posteriores mediante

```
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular);
```

Nótese que, si los coeficientes especulares y difusos son iguales (como es a menudo el caso), se puede especificar ambas usando `GL_DIFFUSE_AND_SPECULAR` para el parámetro `type`. Para especificar propiedades diferentes de caras frontales y posteriores, se usa `GL_FRONT` y `GL_BACK`. El brillo de una superficie, el exponente en el término de reflexión especular, se especifica mediante glMaterialf; por ejemplo

```
glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 100.0);
```

Las propiedades materiales se mantienen mientras los valores de modo no cambien, afectando a superficies solo definidas después del cambio.

OpenGL también permite definir superficies que tengan componentes emisivos que caracterizan fuente auto luminosas. Este método es útil si se quiere que una fuente de luz aparezca en la imagen. Este término no está afectado por ninguna otra de las fuentes de luz, y no afecta ninguna otra superficie. Agrega un color fijo a las superficies y está especificada de manera similar a las demás propiedades materiales. Por ejemplo

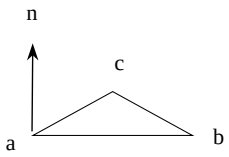
```
GLfloat emission[]={0.0, 0.3, 0.3};

glMaterialfv (GL_FRONT_AND_BACK, GL_EMISSION, emission);
```

define una pequeña cantidad de emisión azul-verde (cian).

Sombreando el modelo de la esfera

Se puede ahora sombrear las esferas. Si se desea sombrear las esferas aproximadas con el modelo de sombreado de OpenGL, se debe asignar normales. Una sencilla, pero ilustrativa, selección es el sombreado plano para cada triángulo, usando los tres vértices para determinar una normal, y luego asignar esta normal al primer vértice, como se ve en la siguiente figura



Siguiendo el enfoque del ejemplo, se usa el producto cruz, y luego se normaliza el resultado.

$$n = \frac{ab \times ac}{|ab \times ac|}$$

$$ab \times ac = \begin{bmatrix} x & y & z \\ b[0]-a[0] & b[1]-a[1] & b[2]-a[2] \\ c[0]-a[0] & c[1]-a[1] & c[2]-a[2] \end{bmatrix}$$

Una función de producto cruz es

```
cross(point a, point b, point c, point d)
{
      d[0]=(b[1]-a[1])*(c[2]-a[2])-(b[2]-a[2])*(c[1]-a[1]);
      d[1]=(b[2]-a[2])*(c[0]-a[0])-(b[0]-a[0])*(c[2]-a[2]);
      d[2]=(b[0]-a[0])*(c[1]-a[1])-(b[1]-a[1])*(c[0]-a[0]);
      normal(d);
}
```

Asumiendo que las fuentes de luz han sido definidas y habilitadas, se puede cambiar la rutina del triángulo para producir las esferas sombreadas:

```
void triangle( point a, point b, point c)
{
      point n;
      cross(a,b,c,n);
      glBegin(GL_LINE_LOOP);
            glNormal3fv(n);
            glVertex3fv(a);
```

```
                glVertex3fv(b);
                glVertex3fv(c);
        glEnd();
}
```

En el resultado del sombreado plano, aunque se incrementen el número de subdivisiones para que el interior de la esfera parezca suave, aún se notan los bordes de los polígonos alrededor de la esfera. Este tipo de borde se conoce como **silueta** (**silhouette edge**).

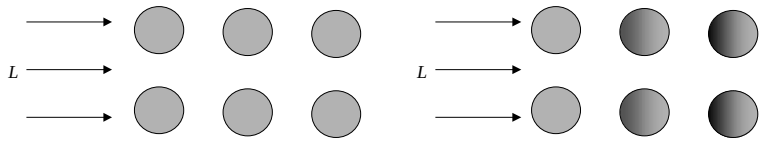
Fácilmente se puede aplicar sombreado interpolativo al modelo de esfera, ya que se conoce que la normal en cada punto **p** sobre la superficie está en la dirección del origen a **p**. Se puede asignar la normal verdadera a cada vértice, y OpenGL interpolará las sombras en estos vértices a lo largo de cada triángulo. Por lo tanto, se puede cambiar `triangle` a

```
void triangle( point a, point b, point c)
{
        point n;
        int j;
        glBegin(GL_POLYGON);
                for(j=0; j<3; j++) n[j]=a[j];
                normal(n);
                glNormal3fv(n);
                glVertex3fv(a);
                for(j=0; j<3; j++) n[j]=b[j];
                normal(n);
                glNormal3fv(n);
                glVertex3fv(b);
                for(j=0; j<3; j++) n[j]=c[j];
                normal(n);
                glNormal3fv(n);
                glVertex3fv(c);
        glEnd();
}
```

Aunque usando las normales verdaderas produce imágenes mas realistas que sombreado plano, el ejemplo no es general, ya que se ha usado normales que se conocen analíticamente. Tampoco se ha provisto una imagen con sombreado Gouraud verdadera. Si se supone que se quiere una imagen con sombreado Gouraud de la esfera aproximada; en cada vértice se necesita conocer las normales de todos los polígonos incidentes en el vértice. El código provisto aquí no tiene una estructura de datos con esa información.

Rendering Global

Existen limitaciones impuestas por el modelo local de iluminación usado. Por ejemplo, si se considera un arreglo de esferas iluminadas por una fuente distante, como se muestra en la siguiente figura



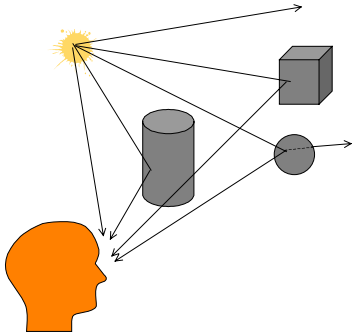
Modelo de Iluminación Local Modelo de Iluminación Global

Las esferas cercanas a la fuente bloquean parte de la luz de la fuente de llegar a las otras esferas. Sin embargo, si se usa el modelo local, cada esfera se sombreará independientemente, apareciendo de la misma forma al observador. Si estas esferas son especulares, en una escena real, parte de la luz será esparcida entre las esferas. El modelo de luz presentado no puede manejar esta situación, además de no poder producir sombras.

Si estos efectos son importantes, se tiene que usar técnicas de generación de imágenes mas sofisticadas (y mas lentas), tales como **trazo de rayos** (**ray tracing**) y **radiosidad** (**radiosity**). Estas dos técnicas son complementarias. Ray tracing funciona bien para superficies altamente especulares, tales como escenas compuestas de objetos altamente reflectivos y translucidos como bolas de vidrio. Radiosidad se aplica mejor a escenas con superficies perfectamente difusas, como interiores de edificios.

Ray Tracing

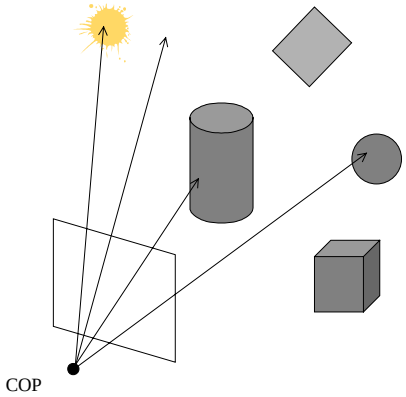
En muchas formas, ray tracing es una extensión al enfoque de rendering con un modelo de iluminación local. Está basado en la observación previa que, de los rayos de luz saliendo de una fuente, los únicos que contribuyen a la imagen son aquellos que entran el lente de la cámara sintética y pasan por el centro de proyección. La siguiente figura muestra varias de las posibles interacciones con una fuente de luz de punto y superficies especulares perfectas.



Los rayos pueden entrar al lente de la cámara (o al ojo humano):

- directamente de la fuente,
- de las interacciones con una superficie visible a la cámara,
- después de múltiples reflexiones,
- después de la transmisión a través de uno o mas objetos.

La mayoría de los rayos de dejan la fuente no entran en el lente y no contribuirán a la imagen. Por lo tanto, tratar de seguir todos los rayos de una fuente de luz es una tarea muy extenuante. Sin embargo, se se invierte la dirección de los rayos, y se consideran solo aquellos que comienzan en el centro de proyección, se sabe que estos **rayos irradiados** (**cast rays**) deben contribuir a la imagen. Por lo tanto, se comienza el trazo de luz como se muestra en la siguiente figura.



Se incluye el plano de la imagen, en término de áreas de pixel, ya que se debe asignar un color a cada pixel, y por lo tanto, se debe irradiar al menos un rayo para cada pixel. Cada rayo irradiado:

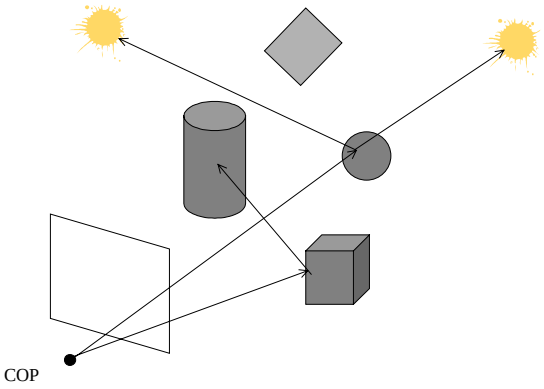
- intersectará una superficie,

- intersectará una fuente de luz,
- se irá al infinito sin pegarle a nada.

Los pixeles correspondientes al último caso pueden asignarse con el color de fondo. Los rayos que pegan en una superficie, inicialmente suponiendo que todos los objetos son opacos, requieren calcular un sombreado para el punto de intersección. Si simplemente se calcula la sombra en el punto de intersección, usando el modelo de Phong, se produciría la misma imagen que utilizando el modelo local. Sin embargo, se puede hacer mucho mas.

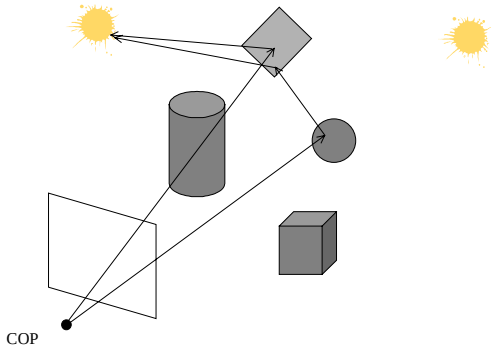
Nótese que el proceso descrito hasta el momento requiere los mismos pasos que en la tubería de generación de imágenes: modelado de objetos, proyección y determinación de superficies visibles. Sin embargo, el orden en que se llevan a cabo los cálculos es diferente. La tubería de generación de imágenes trabaja en base a un vértice a la vez; mientras que el trazador de rayos trabaja en base a un pixel a la vez.

En el trazo de rayos, en lugar de aplicar inmediatamente el modelo de reflexión, primero se revisa si el punto de intersección entre el rayo irradiado y la superficie está iluminado. Se calcula los **rayos de sombra** (**shadow rays**) o **rayos de sondeo** (**feeler rays**) del punto sobre la superficie a cada fuente, como se muestra en la siguiente figura.



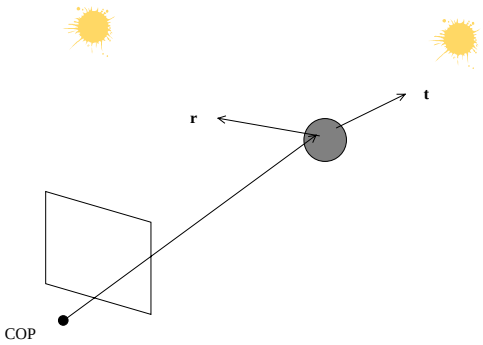
Si un rayo de sombra intersecta una superficie antes de llegar a la fuente, la luz está bloqueada de llegar al punto bajo consideración y este punto está en sombra, por lo menos de esa fuente. No se tienen que hacer cálculos de intersección para fuentes que estén bloqueadas de un punto sobre una superficie. Si todas las superficies son opacas y no se considera la luz esparcida entre superficies, se tiene una imagen que tiene sombras agregadas a lo ya hecho sin ray tracing. El precio pagado es el costo de hacer un cálculo de superficies ocultas para cada punto de intersección entre un rayo irradiado y una superficie.

Supongamos que las superficies son altamente reflectivas, como las que se muestran en la siguiente figura.



Se peude seguir el rayo de sombra según pega entre las diferentes superficies, hasta que se vaya al infinito o intersecte alguna fuente. Tales cálculos normalmente se hacen de forma recurrente, y toman en consideración cualquier absorción de luz en las superficies.

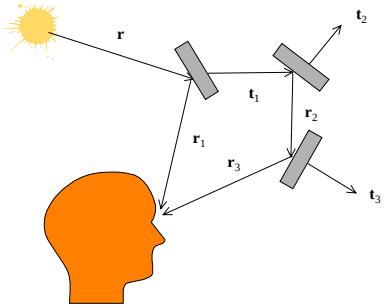
Ray tracing es particularmente bueno para manejar superficies que sean reflectantes y transmitivas, como se ve en la siguiente figura.



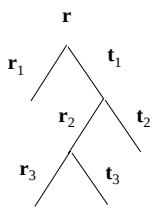
Usando el modelo básico, se sigue una rayo irradiado a una superficie con la propiedad que, si el rayo de uan fuente pega en un punto, entonces la luz de la fuente es parcialmente absorbida, y parte de esta luz contribuye al término de reflexión difusa. El resto de la luz entrante se divide entre un rayo transmitido y un rayo reflectado. Desde la perspectiva de el rayo irradiado, si una fuente de luz es visible en el punto de intersección, entonces se requiere llevar a cabo tres tareas:

1. Se computa la contribución de una fuente de luz en el punto, usando el modelo de reflexión estándar.
2. Se debe irradiar un rayo en la dirección de una reflexión perfecta.
3. Se debe irradiar un rayo en la dirección del rayo transmitido.

Estos dos rayos son tratados como el rayo irradiado original, o sea, se intersectan, si es posible, con otras superficies, terminan en una fuente, o se van al infinito. En cada superficie que estos rayos intersectan, se pueden generar rayos adicionales mediante reflexión o transmisión de la luz. La siguiente figura muestra un rayo irradiado, $\mathbf{r}_1$, y el camino que puede seguir a través de un ambiente sencillo.



La siguiente figura muestra el **árbol de rayos** (**ray tree**) generado, mostrando que rayos deben ser trazados, siendo construido dinámicamente por el proceso de ray tracing.



La forma mas fácil de describir un ray tracer es recursivamente, mediante una sola función que traza un rayo y se llama a si misma para los rayos reflejados y transmitidos. La mayoría del trabajo en ray tracing está en el cálculo de intersecciones entre rayos y superficies. Es difícil de implementar un ray tracer que pueda manejar una variedad de

objetos, ya que, según se agregan mas tipos de objetos, computar interseccioens se vuelve problemático. Por lo tanto, la mayoría de los ray tracers básicos apoyan solo sueprficies planas y cuadráticas.

Aunque el ray tracer presentado aquí utiliza el modelo de Phong para incluir un término difuso en el punto de intersección entre un rayo y una superficie, se ignora la luz que se esparce de manera difusa en este punto. Si se tratara de seguir esa luz, se tendría tantos rayos que seguir que el ray tracer pudiera nunca terminar su ejecución. Por lo tanto, ray tracers son maas apropiados para ambientes altamente reflectivos.

Radiosity

Radiosidad, del otro lado, es ideal para escenas consistiendo solamente de superficies difusas perfectas. Aquí, Se puede obtener un balance de energía global que determina el color para cada superficie poligonal.

Se supone aquí que se tiene una escena sencilla, donde las superficies son perfectamente difusas. Si se procesa esta imagen con una fuente de luz distante, cada superficie poligonal será de un color constante. Si esta fuera una escena real, sin embargo, parte de las reflexiones difusas se reflejarían, ocasionando luz adicional en parte de las demás superficies. El modelo de sombreado sencillo no considera las **interacciones difuso-difuso**.

La ecuación de generación de imágenes resultaría en el sombreado correcto para todas los puntos en las superficies. Si se supone que todas las superficies son perfectamente difusas, se puede simplificar la ecuación a un punto donde exista un método numérico para su solución, **radiosidad** (**radiosity**).

El método de radiosidad básico parte la escena en pequeños polígonos planos, o **patches**, cada una de las cuales se asume es perfectamente difusa y se sombreara con una sombra constante. Lo necesario es encontrar estas sombras. Existen dos pasos para el método:

1. Se consideran los patches en pares para determinar los **factores de forma** (**form factors**) que describen como la energía de luz que deja un patch afecta la otra. Una vez determinados los factores de forma, la ecuación de rendering, que comienza como una ecuación integral, se puede reducir a un conjunto de ecuaciones lineales para radiosidades, esencialmente la reflectividad de las facetas.
2. Una vez resueltas estas ecuaciones, se puede sombrear la escena usando sombreado constante. Aunque la cantidad de cálculos requeridos para computar factores de forma es enorme, es un problema $O(n^2)$ para n patches, una vez se determina las radiosidades de cada patch, estos son independientes de la ubicación del observador, dada la suposición que todas las superficies son perfectamente difusas. Por lo tanto se puede sombrear la imagen rápidamente como cualquier escena que use el modelo de iluminación local.