UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnicatura en diseño
y programación de videojuegos

UNLVIRTUAL

MANIPULACION DE OBJETOS EN 2D

Unidad 4: Modelado de la iluminación y el color

Docente
Walter Sotil

Tutores
Emmanuel Rojas Fredini, Cristian Yones

CONTENIDOS

1. Introducción

Los dispositivos de salida gráfica permiten visualizar las primitivas geométricas en color. En esta unidad veremos los fundamentos de la percepción del color, los diferentes modelos de color que utilizaremos para la medida y la representación del color de los objetos que enviamos a los dispositivos gráficos.

Además, abordaremos la problemática de cómo generar una escena incorporando modelos de iluminación y su implementación en OpenGL.

2. Fundamentos de la teoría del color

El *color* es una sensación que percibe el cerebro cuando interpreta las señales nerviosas que recibe de los fotorreceptores ubicados en la retina del ojo. Éstos, a su vez, interpretan las ondas electromagnéticas, cuya longitud de onda está dentro del espectro visible por el hombre. Las ondas electromagnéticas representan lo que conocemos como luz. Como se observa en la siguiente figura, la luz es sólo una parte del espectro electromagnético.

Color
Es una sensación que percibe el cerebro cuando interpreta las señales nerviosas que recibe de los fotorreceptores ubicados en la retina del ojo.

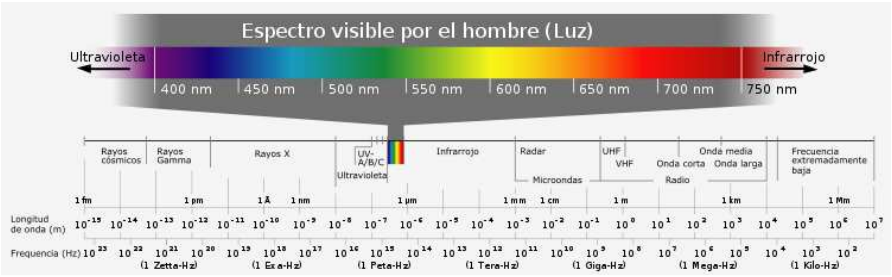

Figura 1. Espectro visible por el hombre.

Todo objeto absorbe una parte de la luz que recibe y refleja otra. La parte reflejada es captada por el ojo e interpretada en el cerebro. Cada material absorbe y refleja distintas proporciones de cada longitud de onda. Por lo tanto, un color es una distribución continua o un espectro de longitudes de onda electromagnética. Por ejemplo, percibimos que una manzana iluminada con luz blanca (que contiene todos los colores) es roja porque la superficie de dicha manzana refleja la luz que tiene la longitud de onda correspondiente al color rojo y absorbe la mayor parte de la energía de los demás colores. Esto nos permite tener idea del material que constituye al objeto observado, ya que distintos materiales reflejan y absorben distintas cantidades de cada longitud de onda.

3. Receptores del ojo humano y percepción del color

Cuando un objeto recibe luz, la parte que se refleja es la que produce la sensación del color en nuestro cerebro.

La *visión* es un sentido que consiste en la habilidad de detectar la luz y de interpretarla.

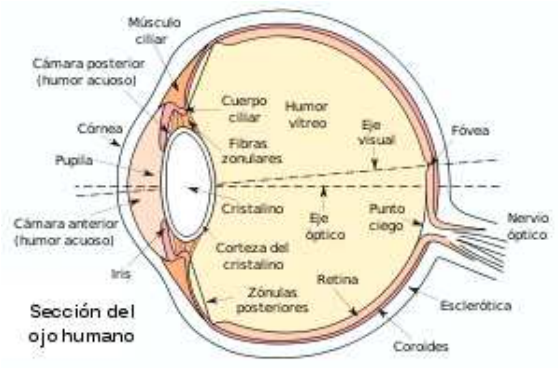

Figura 2. Sección del ojo humano.

La primera parte del sistema visual se encarga de formar la imagen óptica del estímulo visual en la retina (el sistema óptico). Los primeros en intervenir son los fotorreceptores, los cuales capturan la luz que incide sobre ellos. Existen fotorreceptores de dos tipos: los conos y los bastones. Otras células de la retina se encargan de transformar dicha luz en impulsos electroquímicos y de transportarlos hasta el nervio óptico. Desde allí, se proyectan al cerebro, donde se realiza el proceso de formar los colores y reconstruir las distancias, movimientos y formas de los objetos observados.

Las células sensoriales de la retina reaccionan de forma distinta a la luz y a cada longitud de onda. Los bastones son los que poseen mayor sensibilidad y permiten ver en la oscuridad, pero no distinguen el color, sólo pueden percibir una escala de grises con distintas intensidades entre el blanco y el negro. Los conos son menos sensibles pero permiten distinguir las diferencias cromáticas, y adquieren mayor relevancia cuando los niveles de iluminación son suficientemente elevados.

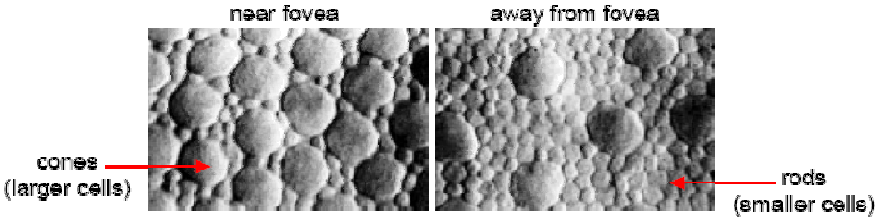

Figura 3. Conos y bastones.

Hay tres tipos de conos: denominados S (Short, corto), *M* (Medium, mediano) y *L*(Long, largo). Cada uno de ellos capta solamente las longitudes de onda señaladas en el gráfico. Posteriormente, transformadas en el cerebro, se corresponden aproximadamente con el azul, verde y rojo.

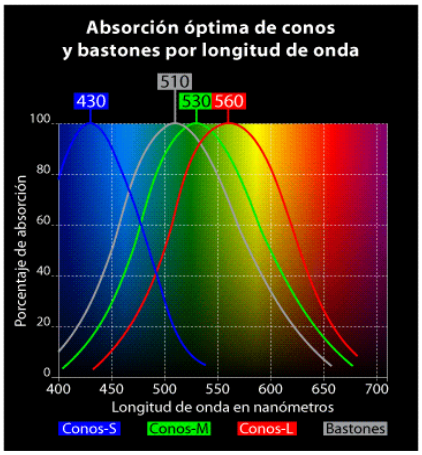

Figura 4. Espectro de absorción de cada tipo de cono.

La información de los conos y bastones es procesada por otras células del ojo, que dan como resultado dos dimensiones o canales de pares antagónicos cromáticos (rojo-verde y azul-amarillo) y una dimensión acromática o canal de claroscuro.

Dicho de otra manera, estas células se excitan o inhiben ante la mayor intensidad de la señal del rojo frente al verde y del azul frente a la del amarillo (suma de rojo y verde), generando además un canal acromático relativo a la luminosidad. La información de este procesamiento se traslada, a través del nervio óptico, al cerebro, donde la actividad neuronal determina los atributos que representan al color: luminosidad, tono y saturación.

4. Síntesis del color

La luz que llega a los ojos puede provenir en forma directa de la fuente o indirectamente cuando llega reflejada por un objeto. Por lo tanto, podemos definir dos formas en que se sintetiza el color: la síntesis aditiva y la sustractiva. La primera se basa en la luz directa, mientras que la segunda, en la luz reflejada.

4.1 Síntesis aditiva

Se llama *síntesis aditiva* al proceso que consiste en obtener un color de luz determinado por la suma de otros colores. Este proceso de reproducción aditiva normalmente utiliza luz roja, verde y azul (colores primarios) para producir el resto de colores. Combinando uno de estos colores primarios con otro en proporciones iguales, se obtienen los colores aditivos secundarios (cian, magenta y amarillo). Variando la intensidad de cada luz de color podemos ver el espectro completo de estas luces. La ausencia de los tres da el negro, en tanto que la suma de los tres da el blanco. Estos tres colores se corresponden con los tres picos de sensibilidad de los tres tipos de conos en nuestros ojos. Los televisores y los monitores de computadoras son las aplicaciones prácticas más comunes de la síntesis aditiva.

Síntesis aditiva
Proceso que consiste en obtener un color de luz determinado por la suma de otros colores. Los televisores y los monitores de computadoras son las aplicaciones prácticas más comunes de este tipo de síntesis del color.

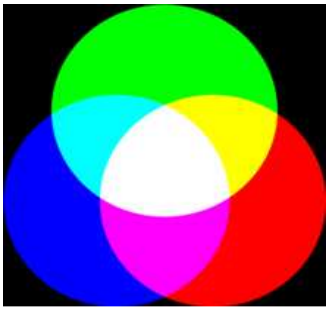

Figura 5. Síntesis aditiva.

4.2 Síntesis sustractiva

Se llama *síntesis sustractiva* al proceso en el cual a la energía de radiación se le sustrae algo por absorción. En este proceso, el color de partida siempre suele ser el acromático blanco y los colores primarios son el amarillo, el magenta y el cian. Cada uno de estos colores tiene la misión de absorber el campo de radiación de cada tipo de conos. Actúan como filtros porque el amarillo no deja pasar las ondas que forman el azul, el magenta no deja pasar el verde y el cian no permite pasar al rojo. La aplicación práctica más común de este tipo de síntesis del color es la impresión y la pintura.

Síntesis sustractiva
Proceso en el cual a la energía de radiación se le sustrae algo por absorción. La aplicación práctica más común de este tipo de síntesis del color es la impresión y la pintura.

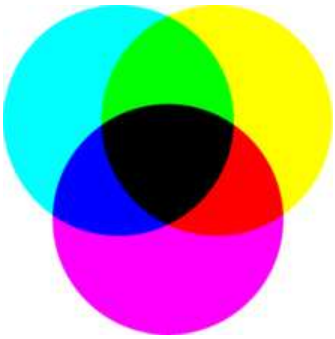

Figura 5. Síntesis sustractiva.

5. Modelos de representación del color

Los *modelos de color* describen matemáticamente cómo pueden ser representados los colores. Existen fórmulas y/o algoritmos para pasar de un modelo a otro, pero no veremos todos, dada la complejidad que presentan algunos.

Modelos de color
Describen matemáticamente cómo pueden ser representados los colores. Los más utilizados son RGB, CMY o CMYK, YUV, HSV y HSL.

A continuación, veremos algunos de los modelos de color más utilizados.

RGB (acrónimo de Red, Green, Blue): este modelo hace referencia a un color en términos de la intensidad de los colores primarios con que se forma y es el más utilizado en trabajos digitales. Se basa en la síntesis aditiva antes vista. En la imagen podemos ver la representación de los colores como coordenadas en el espacio.

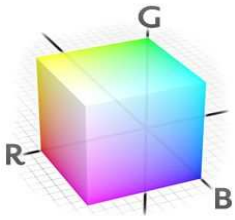

Figura 6. Modelo de color RGB.

CMY o CMYK (acrónimo de Cyan, Magenta, Yellow y Key): es un modelo de colores sustractivo que se utiliza en la impresión. En el mismo se hace referencia a un color en términos de la intensidad de los colores primarios sustractivos con que se forma. La cuarta componente (llamada Key o black) aparece porque en la impresión no es práctico formar el color negro mezclando los tres colores primarios sustractivos. Para pasar CMY a RGB se utiliza la siguiente fórmula:

R = 1-C

G = 1-M

B = 1-Y

YUV: este modelo codifica el color en un canal de luminancia (Y) y dos de crominancia (U y V). Esta forma de codificación tiene en cuenta la percepción humana, ya que los cambios en los canales de crominancia son menos perceptibles por el ojo y por lo tanto se le puede dar menos importancia que al canal Y. Por esta propiedad, este modelo es muy utilizado a la hora de comprimir imágenes o video. Por ejemplo, es usado en los

sistemas PAL y NTSC de difusión de televisión. En la siguiente figura, vemos una imagen con sus tres canales separados:

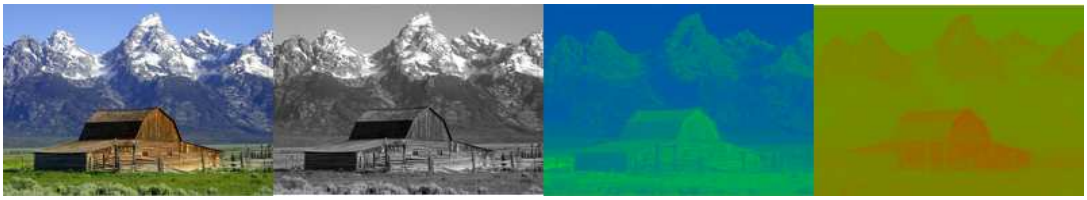

Figura 7. Modelo de color YUV.

HSV (del inglés Hue, Saturation, Value – Tonalidad, Saturación, Valor): sus tres componentes tienen el siguiente significado:

Tonalidad: el tipo de color (como rojo, azul o amarillo). Se representa como una magnitud que va de 0º a 360°, aunque para algunas aplicaciones se normaliza del 0 al 100%. Cada valor corresponde a un color. Ejemplos: 0 es rojo, 60 es amarillo y 120 es verde.

Saturación: se representa como la distancia al eje de intensidad negro-blanco. Los valores posibles van del 0 al 100%. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará.

Valor: es la intensidad del color. Representa la altura en el eje blanco-negro. Los valores posibles van del 0 al 100%. La magnitud 0 siempre es negro. Dependiendo de la saturación, 100 podría ser blanco o un color más o menos saturado.

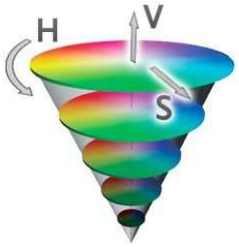

Figura 8. Modelo de color HSV.

HSL: es similar al modelo *HSV,* pero refleja mejor la noción intuitiva de la saturación y la luminancia como dos parámetros dependientes. Esto lo convierte en un modelo más adecuado para los artistas. En la imagen se puede ver la diferencia entre los dos modelos.

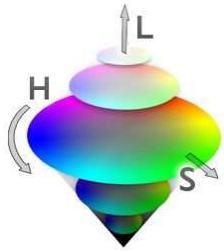

Figura 9. Modelo de color HSL.

6. Modelado de colores y transparencias en OpenGL

6.1 Manipulación de colores con OpenGL

OpenGL define un color utilizando las intensidades separadas de componentes rojo, verde y azul. Por lo tanto, un color se especifica con tres valores numéricos. Debido a que, en general, en el color-buffer se utilizan 24 bits de profundidad para RGB (8 bits para cada componente), se pueden modelar los colores disponibles mediante un cubo de 256 unidades en cada eje. A este volumen lo denominamos *espacio de color RGB.*

El color que se asigna a cada vértice de las entidades a dibujar se puede especificar mediante glColor3{b s i f d ub us ui}(R,G,B). Es decir, usando por ejemplo glColor3f(1.0f,0.0f,0.0f) o glColor3ub(255,0,0) se puede definir el color rojo.

Dependiendo de la configuración, si se utiliza –por ejemplo– *glColor3f,* el rango de cada componente puede variar de 0.0 a 1.0 y los valores para cada una pueden ser cualquier valor de coma flotante válido entre 0 y 1. En la siguiente figura se muestran algunos colores normales con sus valores por componente.

Color compuesto	*Componente roja*	*Componente verde*	*Componente azul*
Negro	0.0	0.0	0.0
Rojo	1.0	0.0	0.0
Verde	0.0	1.0	0.0
Amarillo	1.0	1.0	0.0
Azul	0.0	0.0	1.0
Magenta	1.0	0.0	1.0
Cian	0.0	1.0	1.0
Gris oscuro	0.25	0.25	0.25
Gris claro	0.75	0.75	0.75
Blanco	1.0	1.0	1.0

Figura 10. Algunos colores comunes por componentes.

También es posible obtener gradientes de color. Bastará con que le asignemos un color distinto a cada vértice del polígono para que OpenGL rendererice la imagen con el gradiente de color resultante entre ellos. Para ello, OpenGL realiza en forma automática una interpolación de colores.

Es clave recordar aquí que OpenGL funciona como una máquina de estados. Es decir, seleccionamos un modo o estado y éste permanece hasta que se cambie. Por lo tanto, una vez seleccionado el color de dibujo con *glColor,* éste permanecerá seleccionado (estado actual) hasta que se modifique.

6.2 Canal *alpha*

El modelo RGBA de color dota a cada punto de una cuarta componente *A,* denominada canal *alpha*. Para ello, se usa el comando glColor4{b s i f d ub us ui} (R,G,B,Alpha).

El canal alpha sirve para decidir qué debe hacerse con ese punto: si contribuye o no junto con otros puntos a colorear un píxel. Si activamos el canal alpha, obtendremos mezclas de colores para el píxel. Cada polígono contribuirá en un cierto porcentaje a colorear el píxel, de forma que podamos obtener interesantes efectos como emular un cristal tintado o un papel de celofán rojo.

El canal alpha especifica el grado de opacidad de un objeto (por defecto se utiliza valor 1.0).

Canal *Alpha*
Especifica el grado de opacidad de un objeto. Así, un objeto es totalmente opaco si alpha vale 1.0.

El *blending* mezcla el fragmento entrante con el alojado en el píxel, dependiendo del alpha de cada uno de ellos y la operación de mezcla seleccionada.

Figura 11. Blending: transparencias.

En la operación de mezcla clásica, cuando entra un fragmento semitransparente, es: D = As S + (1-As) D. En esta ecuación, *S* (source) es cada uno de los valores R, G o B del fragmento entrante, mientras que *D* (destination) corresponde, a la derecha, al píxel almacenado en el color buffer y, a la izquierda, al resultado que será almacenado en el píxel. *As* es el valor alpha del píxel entrante, que se utiliza aquí como nivel de opacidad del fragmento que entra. Debemos entender que un objeto es totalmente opaco si alpha vale 1.0. Si alpha vale 0.0 significa un 0% de opacidad, o sea, un 100% de transparencia.

En general, el blending se define mediante la función glBlendFunc(sfactor,dfactor), con un factor multiplicativo para *S* y otro para *D.* La tabla de factores es la siguiente (Tabla 7.1 del *Red Book*):

Constant	Relevant Factor	Computed Blend Factor
GL_ZERO	source or destination	(0, 0, 0, 0)
GL_ONE	source or destination	(1, 1, 1, 1)
GL_DST_COLOR	source	(R_d, G_d, B_d, A_d)
GL_SRC_COLOR	destination	(R_s, G_s, B_s, A_s)
GL_ONE_MINUS_DST_COLOR	source	(1, 1, 1, 1)-(R_d, G_d, B_d, A_d)
GL_ONE_MINUS_SRC_COLOR	destination	(1, 1, 1, 1)-(R_s, G_s, B_s, A_s)
GL_SRC_ALPHA	source or destination	(A_s, A_s, A_s, A_s)
GL_ONE_MINUS_SRC_ALPHA	source or destination	(1, 1, 1, 1)-(A_s, A_s, A_s, A_s)
GL_DST_ALPHA	source or destination	(A_d, A_d, A_d, A_d)
GL_ONE_MINUS_DST_ALPHA	source or destination	(1, 1, 1, 1)-(A_d, A_d, A_d, A_d)
GL_SRC_ALPHA_SATURATE	source	(f, f, f, 1); f=min(As, 1-Ad)

En el ejemplo mostrado más arriba, la configuración utilizada sería la siguiente: glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA). En el *Red Book* se pueden ver algunos trucos y usos alternativos del blending.

7. Sombreado de superficies y modelos de iluminación

Sombreado de superficies
Consiste en calcular el color con el que se debe pintar un objeto en la pantalla. No se debe confundir con proyección de sombras. Ésta es el resultado de la oclusión de la luz por parte de un objeto, mientras que el sombreado es el efecto que causa la luz sobre la superficie del objeto.

El *sombreado de superficies* es un proceso que se desarrolla a la hora de renderizar escenas y consiste en calcular el color con el que se debe pintar un objeto en la pantalla, teniendo en cuenta el *material* de dicho objeto, su posición en la escena y las fuentes de luz, entre otras cosas.

Es importante no confundir *proyección de sombras* con sombreado. La primera es el resultado de la oclusión de la luz por parte de un objeto, mientras que la segunda es el efecto que causa la luz sobre la superficie del objeto. En este curso sólo abordaremos el sombreado de objetos. En la siguiente figura se puede visualizar la diferencia entre una esfera sin sombreado y otra con:

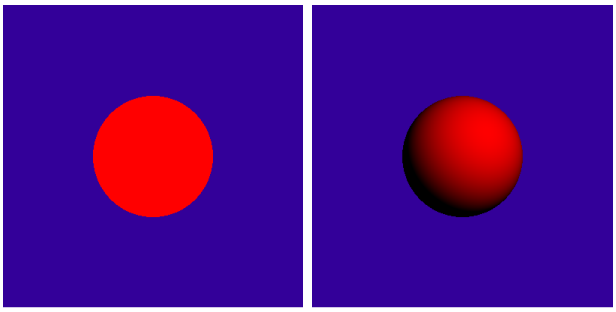

Figura 12. Izquierda: esfera sin cálculo de sombreado. Derecha: esfera con sombreado.

Modelos de iluminación
Posibilitan el cálculo del color de cada punto de la escena. Existen modelos de iluminación local y de iluminación global. En la industria de los videojuegos normalmente se utilizan los primeros.

Para sombrear un objeto es importante entender la relación del mismo con la luz y cómo modelar esa relación. Es aquí donde aparecen los *modelos de iluminación,* que tienen la finalidad de posibilitar el cálculo del color de cada punto de la escena.

Desde un punto de vista físico, una superficie puede emitir luz, como el sol, o puede reflejar luz que recibe de una fuente o de otros objetos. El color que se ve en un punto de un objeto, entonces, está determinado por las múltiples interacciones entre las fuentes de luz y las superficies reflectoras, que pueden estar constituidas por distintos materiales y por lo tanto reaccionar de diferentes maneras a la luz. Este es un proceso recursivo muy complejo y difícil de simular en una máquina, por lo que generalmente se simplifica perdiendo un poco de calidad.

Aquí podemos hacer la distinción entre dos tipos de modelos de iluminación:

Iluminación local: aquí, a la hora de calcular el color con el que se debe pintar un píxel de un objeto, sólo se tiene en cuenta el material de dicho objeto, la luz ambiente, la posición de las fuentes de luz (el ángulo de incidencia de la luz) y la posición del observador (el ángulo con el eje visual).

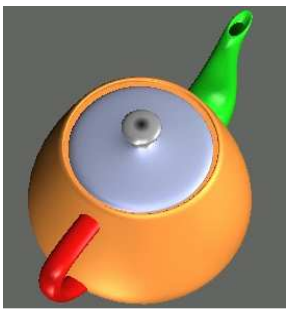

Figura 13. Modelo de iluminación local.

Iluminación global: en este modelo se tiene en cuenta la luz reflejada por otros objetos, dando como resultado imágenes mucho más realistas que las que se obtienen con los

modelos de iluminación local. La desventaja es que es muy costoso en cálculo. Algunos de estos modelos de iluminación son *Radiosity, Ray Casting, Ray Tracing, Photon Map,* entre otros.

Figura 14. Modelo de iluminación global.

En la industria de los videojuegos normalmente se utilizan modelos locales de iluminación, ya que las máquinas actuales no cuentan con la potencia de cálculo necesaria para renderizar escenas en tiempo real utilizando modelos de iluminación global.

A continuación, veremos el modelo de iluminación local de *Phong* (desarrollado por Bui Tuong Phong en su tesis de doctorado, en 1973), que es actualmente el más utilizado. Este modelo de iluminación local, junto con otra mínima incorporación, es el implementado en OpenGL.

8. Modelo de iluminación de Phong

Modelo de iluminación de Phong
Es un modelo empírico en el cual se realiza el cálculo de la intensidad de cada punto de la escena en base a la suma de tres tipos de iluminación: ambiente, difusa y especular.

Es un modelo empírico en el cual se realiza el cálculo de la intensidad de cada punto de la escena en base a la suma de tres tipos de iluminación:

- Iluminación ambiente.
- Iluminación difusa.
- Iluminación especular.

Según el modelo de iluminación local de Phong, la iluminación recibida por el ojo (cámara) desde un punto de una superficie es la combinación lineal de varias componentes. Como explicaremos más adelante, depende de las direcciones del ojo, de la luz y de la normal a la superficie en el punto. El conjunto de constantes multiplicativas que integran la combinación lineal son las propiedades del material. Finalmente, todo se multiplica por la intensidad de la luz incidente.

Se tiene, entonces, la siguiente ecuación: Iphong = Iambiental + Idifusa + Iespecular

En esta ecuación, *Iphong* representa la intensidad del punto, pero con este único valor sólo podemos construir una imagen en escala de grises. Si queremos obtener una imagen a color, debemos trabajar con las tres componentes RGB (o también con RGBA), es decir, tendremos un valor de intensidad para cada canal de color. Por este motivo, al especificar las características de un material o de una fuente de luz, debemos hacerlo como ternas de componentes, una por cada canal, como veremos más adelante.

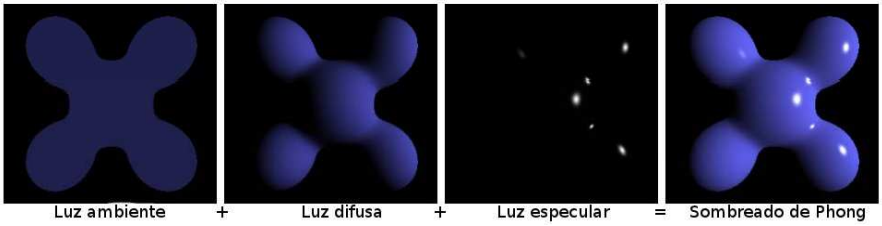

Figura 15. Modelo de iluminación local de Phong.

A continuación, analizaremos cómo se calcula cada una de las tres contribuciones, haciendo su aporte a la intensidad final del punto.

En estos cálculos entran en juego cuatro vectores:

- *L:* vector unitario con dirección a la fuente de luz.
- *N:* vector unitario normal a la superficie.
- *R:* vector unitario con dirección adonde la luz se refleja. Es coplanar a *N* y *L*. Forma con *N* un ángulo igual al que forma *L*.
- *V:* vector unitario con dirección al observador.

Tenemos la intensidad de la luz, que se representa con tres constantes: *Ia* (iluminación ambiente), *Id* (iluminación difusa) e *Ie* (iluminación especular).

Además, poseemos tres coeficientes empíricos que dependen de las propiedades ópticas del material: *Ka, Kd* y *Ke.*

La intensidad de luz y las constantes del material son tres componentes para ambiente, difusa y especular, y cada una en tres componentes roja, verde y azul. En total son nueve números para cada luz y nueve para cada material.

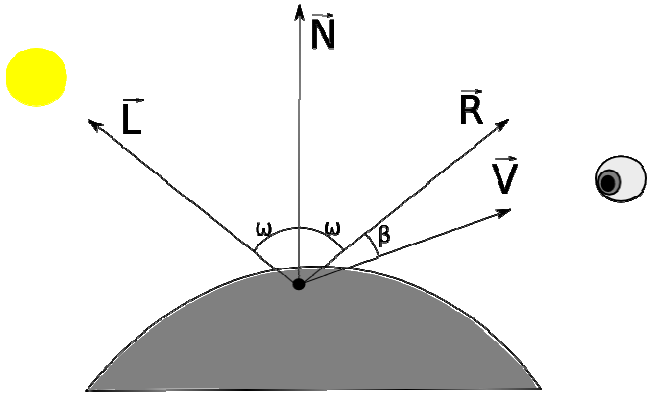

Figura 16. Vectores intervinientes en el modelo de iluminación local de Phong.

Como se expresó anteriormente, se realiza el cálculo de la intensidad de cada punto de la escena en base a la suma de tres tipos de iluminación:

Iluminación ambiente: no proviene de una dirección concreta, por lo que incide sobre todas las partes del objeto con la misma intensidad. Simula el proceso de reflexión de la luz sobre los demás objetos de la escena. Evita que las zonas sin luz directa se visualicen totalmente en negro. El cálculo de esta componente de la iluminación es el siguiente:

Iambiental = Ka . Ia

Iluminación difusa: un reflector difuso perfecto esparce la luz que refleja de idéntica manera en todas las direcciones, viéndose igual para todos los observadores. Sin embargo, la cantidad de luz reflejada depende de la posición de la fuente de luz con respecto a la superficie y del material en cuestión. Tales superficies son a veces conocidas como *superficies Lambertianas,* pudiéndose modelar matemáticamente por la ley de Lambert. La misma sostiene que la cantidad de luz reflejada es proporcional al coseno del ángulo entre el vector normal a la superficie y el vector dirección de la luz. Si estos dos vectores son unitarios se puede definir:

$$Idifusa = Kd \cdot Id \cdot \cos(\omega) = Kd \cdot Id \cdot (L*N)$$

Iluminación especular: procede de una dirección concreta y se refleja "principalmente" en una única dirección, produciendo brillos intensos (por ejemplo, los objetos metálicos). Como la difusa, sólo afecta a las partes del objeto en las que la luz incide directamente. La iluminación depende del ángulo entre la dirección de incidencia de la luz y la posición del observador. Esto se puede modelar matemáticamente siguiendo la fórmula de Phong:

$$Iespecular = Ke \cdot Ie \cdot \cos^{q}(\beta) = Ke \cdot Ie \, (R*V)^{q}$$

Donde *q* es el coeficiente de brillo. En el límite, según *q* tiende a infinito, se obtiene un espejo.

Si juntamos todas las ecuaciones vistas en esta sección, el modelo que define la intensidad del punto nos queda:

$$Iphong = Ia \cdot Ka + Id \cdot Kd \, (L*N) + Ie \cdot Ke \, (R*V)^{q}$$

Atenuación de la luz

Cuando la luz viaja por el espacio se va atenuando según una función de la distancia *d* entre el foco de luz y el objeto. Para lograr una aproximación empírica que permita simular el efecto de esta atenuación, se utiliza una función de atenuación con la siguiente forma:

$$f(d) = 1 / (a + b \cdot d + c \cdot d^2)$$

Aquí, los coeficientes *a*, *b* y *c* pueden manipularse para obtener distintos efectos de iluminación. Esta función de atenuación sólo afectará a las componentes difusa y especular de la luz, en el modelo de Phong, ya que la luz ambiente es independiente de la fuente.

En OpenGL se puede agregar, además, la emisión, que no es parte del modelo original, simulando materiales luminosos, así como una atenuación de la luz reflejada en función de la distancia al objeto (fog o neblina).

9. Tipos de sombreado

Al momento de renderizar un objeto, se parte de una malla formada por varios polígonos (normalmente triángulos), donde cada vértice de los mismos tiene un vector normal a la superficie representada por el polígono. Para poder aplicar el modelo de iluminación desarrollado a cada punto de una superficie, deberíamos contar con una normal para cada punto de la misma. Lamentablemente, aunque tengamos ecuaciones sencillas para determinar los vectores normales, como en una esfera, la cantidad de cálculos requeridos puede ser muy grande.

Por este motivo, se toma una de las siguientes opciones para sombrear un polígono:

Sombreado plano o constante (Flat Shading): este método aplica el color o lo calcula según el modelo de iluminación una sola vez por cada polígono. El color calculado en uno de los vértices (generalmente el último) se utiliza para sombrear todos los puntos del polígono.

Sombreado interpolativo o Gouraud (Gouraud Shading): a cada vértice del polígono se asigna el color o se calcula usando el modelo de iluminación. La intensidad de cada componente en los puntos intermedios se calcula por interpolación.

Sombreado Phong (Phong Shading): en este método de sombreado se interpolan las normales y luego se aplica el modelo de iluminación en cada punto del polígono. Es el que mejores resultados otorga si lo que se intenta es simular una superficie curva, pero también es el más costoso en cálculos.

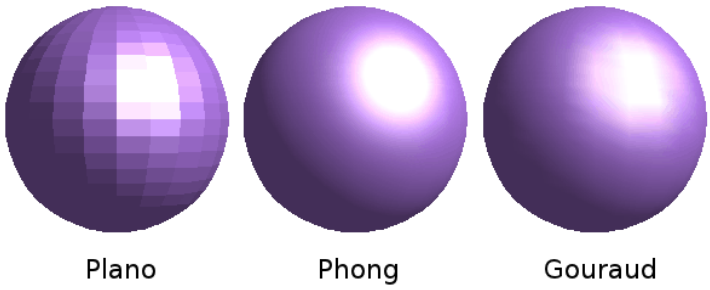

Figura 17. Tipos de sombreado.

En OpenGL sólo se encuentran implementados los dos primeros, es decir, sombreado plano y Gouraud.

Se puede elegir el tipo de sombreado llamando a la función `glShadeModel(`tipo de sombreado`).`

Para sombreado plano se utiliza glShadeModel(GL_FLAT).

Para sombreado interpolativo se usa glShadeModel(GL_SMOOTH). Este es el sombreado seteado por defecto.

El sombreado de Phong (no confundir con el modelo de iluminación de Phong) no se encuentra implementado en OpenGL. Para utilizarlo se debe programar un shader, pero este tema queda fuera del alcance del presente curso.

10. Fuentes de luz en OpenGL

10.1 Incorporación de fuentes de luz

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

Las funciones de OpenGL

```
                                .nter_to_array)
                                ıe)
```

permiten asignar los parámetros de forma vectorial o escalar, respectivamente.

Existen cuatro parámetros vectoriales 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 quisiéramos especificar la primera fuente de luz GL_LIGHT0 y ubicarla en el punto (1.0, 2.0, 3.0), su posición se guardaría 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 luz de punto se transforma en una fuente distante, con una dirección de vector (1.0, 2.0, 3.0):

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

Para una sola fuente de luz, si deseáramos una componente especular blanca y componentes ambiente y difusa roja, tendríamos 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, 0.0, 0.0, 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);
```

Notemos que debemos habilitar la iluminación y la fuente de luz particular. El último argumento es el canal alpha.

También se puede agregar un término de ambiente global que es independiente de cualquiera de las fuentes. Por ejemplo, si queremos agregar una pequeña cantidad de luz blanca, podríamos usar 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 antes Visto:

$f(d) = 1 / (a + b \cdot d + c \cdot d^2)$

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);
```

Las fuentes de luz son objetos, al igual que polígonos y puntos. Por lo tanto, están afectadas por las transformaciones modelo-vista de OpenGL. Se las puede definir en la posición deseada o en una posición conveniente y moverlas a la posición deseada mediante transformaciones modelo-vista (GL_MODELVIEW).

10.2 Ahorro de cálculos en el sombreado

Existen otros dos parámetros de luz provistos por OpenGL que vale la pena mencionar:

GL_LIGHT_MODEL_LOCAL_VIEWER y GL_LIGHT_MODEL_TWO_SIDE.

Si se asume que el observador está a una distancia infinita de la escena, el cálculo de reflexiones es más fácil porque la dirección al observador desde cualquier punto en la escena no cambia (*V* es constante). El efecto de esta aproximación en muchas escenas es mínimo. En OpenGL se puede activar con la línea:

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,
GL_FALSE);
```

Si preferimos que se hagan los cálculos completos de luz, usando la posición real del observador, deberíamos cambiar el modelo, usando:

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

Por otro lado, GL_LIGHT_MODEL_TWO_SIDE indica a OpenGL si es necesario o no hacer los cálculos de iluminación y sombreado en la cara posterior de las superficies. Para polígonos se determina el frente y el reverso, según el orden en que se especifican los vértices, utilizando la regla de mano derecha.

Por ejemplo, para objetos cerrados como una esfera o un cubo, sólo se ven las caras frontales, por lo cual, en general, no es de interés sombrear las superficies posteriores.

Para activar o desactivar esta opción, se usa la función:

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, 0 o 1)
```

10.3 Especificación de materiales en OpenGL

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

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

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

Por ejemplo, se pueden definir coeficientes de reflectividad ambiente, difuso y especular (Ka, Kd y Ke) para cada uno de los colores primarios:

```
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_DIFFUSE, mat_diffuse);

glMaterialfv(GL_FRONT_AND_BACK,GL_SPECULAR, mat_specular);
```

```
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
```

Si los coeficientes especulares y difusos son iguales, se pueden 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 coeficiente de brillantez q en el término de reflexión especular, se especifica mediante glMaterialf, por ejemplo:

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

GL_SHININESS puede tomar valores en el rango [0, 128]. El valor por defecto es 0.

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

10.4 Emisión de luz

OpenGL también permite definir superficies con componentes emisivos que caracterizan fuentes auto luminosas. El método es útil si queremos que una fuente de luz aparezca en la imagen. Esta componente emisiva no está afectada por ninguna otra fuente de luz y no afecta a 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).

BIBLIOGRAFÍA

Cátedra Computación Gráfica, FICH - UNL, 2010 [en línea] [CIMEC]
www.cimec.org.ar/cg

Hearn, D.; Baker, P. *Computer Graphics, C version.* Second Edition. Pearson Prentice Hall, 1997.

Neider J.; Davis,T.; Woo M. "OpenGl Programming Guide: The Official Guide to Learning OpenGL". Addison-Wesley Publishing Company, 1997 [en línea] [OpenGL]
http://www.opengl.org/documentation/red_book/

The OpenGL Light Bible, 2011 [en línea] [falloutsoftware]
http://www.falloutsoftware.com/tutorials/gl/gl8.htm