

# 4.- Iluminación.

---

4.1.- Iluminación en el mundo real y en OpenGL	32
4.2.- Creación de fuentes de luz	33
4.3.- Seleccionar un modelo de iluminación	37
4.4.- Seleccionar propiedades del material	38

---

En este tema veremos como controlar la iluminación en una escena. Con OpenGL podemos manipular la luz y los objetos en una escena para crear diferentes clases de efectos. Cuando iluminamos una escena el resultado de la iluminación depende de qué luz usamos y como los objetos de la escena reflejan o absorben la luz. Un ejemplo perfecto es el mar, el cual en un día soleado tiene un color diferente al que tiene en un día nuboso. Ciertos objetos reflejan la luz (paredes, escritorios, papel), mientras que otros la transmiten (celofán, vidrio).

Las secciones que veremos en este capítulo son las siguientes:

- Comportamiento de la luz en el mundo real y cómo OpenGL modela este comportamiento.
- Definir y colocar las fuentes de luz.
- Definir los elementos de un modelo de iluminación.
- Describir las propiedades de los objetos, materiales.

En este capítulo veremos las funciones de OpenGL que nos facilitan estas operaciones.

---

## 4.1.- ILUMINACIÓN EN EL MUNDO REAL Y EN OPENGL.

Cuando miramos a una superficie, la percepción del color por parte de nuestros ojos depende de la distribución de energía de los fotones que llegan. Estos fotones vienen desde una fuente o combinación de fuentes de luz, algunos de los cuales son absorbidos y otros son reflejados por la superficie. Las superficies pueden ser brillantes y reflejan la luz normalmente en ciertas direcciones, mientras que otras superficies disipan la luz que les llega igualmente en todas las direcciones.

OpenGL aproxima la luz e iluminación como si pudiera romperse en tres componentes (rojo, verde y azul). El color de las fuentes de luz está caracterizado por la cantidad de luz roja, verde y azul que emiten, y el material de las superficies está caracterizado por el porcentaje de luz (componentes rojo, verde y azul) que reflejan en varias direcciones. En el modelo de iluminación de OpenGL, la luz en una escena proviene de varias fuentes que pueden apagarse o encenderse individualmente. Algunas luces vienen de una dirección o posición determinada, mientras que otras aparecen disipadas a lo largo de toda la escena.

### Luz emitida, ambiente, difusa y especular

En el modelo de iluminación de OpenGL la luz se considera dividida en cuatro componentes independientes: emitida, ambiente, difusa y especular. Las cuatro componentes se procesan independientemente para posteriormente juntarlas.

La luz **emitida**, es la más elemental, proviene de un objeto y no se ve afectada por ninguna fuente de luz. La luz **ambiente** es aquella que proviene de una fuente que ha sido tan disipada por el entorno que es imposible determinar su dirección. Cuando una luz ambiente golpea una superficie, esta es disipada igualmente en todas direcciones. La luz **difusa** procede de una dirección de forma que es más brillante si da de lleno en la superficie que si se desvía de la superficie. Una vez que golpea una superficie se disipa igualmente en todas direcciones, así que aparece igualmente brillante, no importando donde el ojo está localizado. Cualquier luz que viene de una posición o dirección particular probablemente tenga un componente difuso. La luz **especular** procede de una dirección particular, y tiende a rebotar en una dirección determinada. El metal o plástico brillante tienen un alto componente especular, mientras que la tiza o una alfombra no tienen casi nada de este componente. Podemos ver la luz especular como lo la luz brillante.

OpenGL permite establecer los valores de rojo, verde y azul (RGB) para cada componente independientemente.

### Materiales

Para el modelo de iluminación de OpenGL el color de un material depende de los porcentajes de luz incidente roja, verde y azul que éste refleja. Una bola roja refleja toda la luz roja y absorbe toda la verde y azul. Si vemos esa bola con luz blanca (compuesta de igual cantidad de rojo, verde y azul), toda la roja es reflejada y vemos la bola roja. Si la bola la

vemos con una luz roja también aparece roja. Si la vemos con luz verde, aparece negra, toda la luz verde es absorbida y como no recibe luz roja no refleja luz roja.

Como la luz, los materiales tienen diferentes componentes de color: ambiente, difusa, especular que determinan las reflectancias del material. Un reflejo ambiente se combina con la componente ambiente de la luz que le incide, lo mismo ocurre con la difusa y especular.

### Valores RGB para luz y materiales

Los componentes de color para la luz y materiales tienen distinto significado. Para la luz, los números corresponden al porcentaje de intensidad para cada color. Si los R, G, B son todos uno es una luz blanca lo más brillante posible. Si los tres son 0.5 el color es todavía blanco pero de intensidad media, de forma que aparece gris. Si R y G son 1 y B es 0 (completamente roja y verde sin azul), la luz aparece amarilla. Para los materiales, los números corresponden a las proporciones de luz reflejada de esos colores. Así si R=1 y G=0.5 y B=0 para un material, ese material refleja toda la luz roja, la mitad de la verde, y ninguna de la luz azul.

### Definición de vectores normales para cada vértice

La normal de un objeto determina su orientación relativa a las fuentes de luz. Para cada vértice, OpenGL usa las normales para determinar cuánta luz recibe un vértice de cada fuente de luz. Las normales en las funciones de la librería *glaux*, por ejemplo *auxSolidSphere*, están definidas como parte de la propia rutina.

## 4.2.- CREACION DE FUENTES DE LUZ

Las propiedades de una fuente de luz son el color, posición y dirección. El comando usado para especificar todas las propiedades de una luz es **glLight\*()**, tiene tres argumentos para identificar la luz que se está utilizando, la propiedad y el valor deseado para esa propiedad.

Después de definir las características de las luces, hay que encenderlas con el comando **glEnable()**. También hay que llamar a este comando con el parámetro **GL\_LIGHTING**, de esta forma se prepara OpenGL para desarrollar los cálculos de iluminación.

---

```
void glLight{if}[v] (GLenum light, GLenum pname, TYPE param )
```

---

Crea la luz especificada por *light*, la cual puede ser **GL\_LIGHT0**, **GL\_LIGHT1**, ...**GL\_LIGHT7**. Las características de la luz quedan definidas por *pname*. El argumento *param* indica los valores que establece la característica *pname*; si se utiliza la versión de

vector(v) *pname* es un puntero a un grupo de valores, o el propio valor si la versión sin vector es la que estamos usando(i,f).

Valores por defecto para el parámetro *pname* de **glLight\*()**.

Nombre del parámetro	Valor por defecto	Significado
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	RGBA-intensidad de luz ambiente
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	RGBA-intensidad de luz difusa
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	RGBA-intensidad de luz especular
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(x, y, z, w) posición de la luz
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(x, y, z, w) dirección de la luz
GL_SPOT_EXPONENT	0.0	exponente "spotlight"
GL_SPOT_CUTOFF	180.0	ángulo "spotlight"
GL_CONSTANT_ATTENUATION	1.0	factor de atenuación constante
GL_LINEAR_ATTENUATION	0.0	factor de atenuación lineal
GL_QUADRATIC_ATTENUATION	0.0	factor de atenuación cuadrático

Los valores por defecto de GL\_DIFFUSE y GL\_SPECULAR en la tabla anterior se aplican solo para GL\_LIGHT0. Para otras luces, el valor por defecto para GL\_AMBIENT y GL\_DIFFUSE es (0.0, 0.0, 0.0, 1.0).

Ejemplo:

```
GLfloat light_ambient[]={0.0, 0.0, 0.0, 1.0};
GLfloat light_diffuse[]={1.0, 1.0, 1.0, 1.0};
GLfloat light_specular[]={1.0, 1.0, 1.0, 1.0};
GLfloat light_position[]={1.0, 1.0, 1.0, 0.0};
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

## Color

OpenGL permite asociar a cualquier luz tres parámetros relacionados con el color: GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR. El parámetro GL\_AMBIENT se refiere a la intensidad RGBA de la luz ambiente que una fuente de luz añade a la escena. El valor por defecto es que no hay luz ambiente. El parámetro GL\_DIFFUSE es el que está más relacionado con el concepto que tenemos normalmente de "color de una luz"; se refiere a la intensidad RGBA de la luz difusa que una fuente de luz añade a la escena. El valor por defecto es (1.0, 1.0, 1.0, 1.0) para GL\_LIGHT0 que produce una luz blanca brillante. El valor por defecto para GL\_LIGHT1... GL\_LIGHT7 es (0.0, 0.0, 0.0, 0.0). El parámetro GL\_SPECULAR afecta a el color del brillo especular en un objeto. Un objeto como un vaso tiene una luz especular que es el color de la luz que está brillando en él, normalmente blanca. Si queremos crear un efecto realista, el parámetro GL\_SPECULAR y GL\_DIFFUSE se deben establecer al mismo valor.

## Posición y atenuación

Podemos clasificar la fuente de luz que incide sobre una escena en dos tipos: direccional y puntual. Una **fuente direccional** es aquella que está situada en el infinito, de forma que los rayos de luz pueden considerarse paralelos en el momento que alcanzan un objeto. El sol es una fuente de luz de este tipo. La fuente de luz puntual está situada cerca de la escena, su posición exacta determina el efecto en la escena. Una lampara de escritorio es una fuente de luz de este segundo tipo.

Una fuente de luz direccional:

```
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Si el valor  $w$  de `GL_POSITION` ( $x,y,z,w$ ) es 0.0 la fuente de luz es direccional y los tres primeros valores determinan la dirección. Esta dirección es transformada por la matriz de *modelview* como si describiera un vector normal. Por defecto `GL_POSITION` es (0,0,1,0) que define una luz direccional que apunta hacia el eje  $z$  negativo. No tiene sentido atenuar con la distancia la intensidad de una fuente de luz direccional.

Si el valor de  $w$  es distinto de cero, la luz es puntual, y los valores ( $x,y,z$ ) especifican la localización de la luz en coordenadas de objeto homogéneas. Esta localización es transformada por la matriz de *modelview* y almacenada en coordenadas de vista. Una luz puntual normalmente ilumina en todas las direcciones, pero podemos restringirlo a un cono de luz definiendo un “*spotlight*”. En una fuente de luz puntual si que podremos atenuar la luz con la distancia, para ello se define: la distancia entre la posición de la luz y el vértice,  $k_c$ =constante de atenuación,  $k_l$ =atenuación lineal y  $k_q$ =atenuación cuadrática. Normalmente  $k_c$  vale 1 y las demás 0.

La ambiente, difusa y especular son todas atenuadas, solo la emisión y la global ambiente no son atenuadas.

### Luz dirigida (Spotlights)

Podemos restringir la figura de una fuente de luz para que emita en forma de cono. Para crear una spotlight tenemos que determinar la extensión del cono de luz que deseamos. Para especificar el ángulo entre el eje del cono y un rayo al lo largo del borde usamos el parámetro `GL_SPOT_CUTOFF`. Por defecto la característica de spotlight está deshabilitada porque este ángulo es 180 (emite en todas las direcciones).

```
glLightfv(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
```

También es necesario especificar la dirección, lo que determina el eje del cono de luz. Por defecto ésta dirección apunta hacia el eje  $z$  negativo.

```
GLfloat spot_direction[] = {-1.0, -1.0, 0.0}
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
```

La dirección está especificada en coordenadas homogéneas, la dirección de un cono de luz es transformada por la matriz de modelo-vista como si fuera un vector normal y el resultado es almacenado en coordenadas de vista.

Además de el ángulo y dirección se puede controlar de dos formas la intensidad de distribución de luz dentro del cono. Primero puedes establecer el factor de atenuación descrito anteriormente que se multiplica por la intensidad de la luz. También puedes establecer el `GL_SPOT_EXPONENT` para controlar como de concentrada esta la luz. La intensidad de luz es mayor en el centro del cono.

### Varias luces

Podemos tener al menos ocho fuentes de luz en la escena (puede haber más, depende de la implementación de OpenGL. OpenGL necesita realizar cálculos para determinar cuanta luz recibe cada vértice de cada fuente de luz. El incremento del número de luces influye en una ejecución más lenta.

Las constantes usadas para referirse a las ocho luces: `GL_LIGHT0`, `GL_LIGHT1` .. `GL_LIGHT7`. Anteriormente hemos establecido parámetros relacionados con `GL_LIGHT0`. Si quieres una luz adicional, necesitas especificar sus parámetros; recuerda que los valores por defecto son diferentes para las otras luces que no son `GL_LIGHT0`.

Un ejemplo que define una “spotlight” blanca atenuada:

```
GLfloat light1_ambient[ ]={0.2, 0.2, 0.2, 1.0};
GLfloat light1_diffuse[ ]={1.0, 1.0, 1.0, 1.0};
GLfloat light1_specular[ ]={1.0, 1.0, 1.0, 1.0};
GLfloat light1_position[ ]={-2.0, 2.0, 1.0, 1.0};
GLfloat spot_direction[ ]={-1.0, -1.0, 0.0};

glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
glLightfv(GL_LIGHT1, GL_POSITION, light1_position);

glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.5);
glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.5);
glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.2);

glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction
);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);

glEnable(GL_LIGHT1);
```

### Controlando la posición y dirección de una luz

OpenGL trata la posición y dirección de una fuente de luz justo como trata la posición de una primitiva geométrica. Una fuente de luz está sujeta a la misma matriz de transformaciones que una primitiva. Cuando `glLight*()` se llama para especificar la posición y dirección de una fuente de luz, la posición y dirección se transforma por la actual matriz de modelo-vista y se almacena en coordenadas de vista, esto quiere decir que podemos cambiar la posición y orientación de una luz cambiando los contenidos de la matriz. Podemos conseguir los tres efectos siguientes, cambiando el punto del programa en que la posición de la luz se establece con respecto a las transformaciones de modelado y vista:

- una posición fija  
se establece la posición de la luz después de las transformaciones de modelo-vista
- una luz que se mueve a lo largo de un objeto estacionario  
se establece la posición de la luz después de las transformaciones de modelado que ellas mismas cambian para modificar la posición de la luz. Dentro de un bucle de eventos, necesitamos desarrollar la transformación de modelado y resetear la posición de la luz.
- una luz que se mueve con el punto de vista  
se establece la posición de la luz antes de las transformaciones de vista, de forma que la transformación de vista afecta a la luz y al punto de vista.

### 4.3.- SELECCIONAR UN MODELO DE ILUMINACION

Un modelo de iluminación de OpenGL tiene tres componentes:

- La intensidad de la luz global ambiente  
Cada fuente de luz puede contribuir a la luz ambiente de la escena. Puede haber otra luz ambiente que no viene de una fuente en particular. Para especificar la intensidad RGBA de esa luz global usamos el parámetro `GL_LIGHT_MODEL_AMBIENT`:

```
GLfloat lmodel_ambient[ ]={0.2, 0.2, 0.2, 1.0};  
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lmodel_ambient);
```

- Posición del punto de vista: local a la escena o distancia infinita.  
La ubicación del punto de vista afecta al calculo para el brillo producidas por la reflectancia especular. Mas específicamente, la intensidad del brillo en un vértice particular depende de la normal a ese vértice, la dirección desde el vértice a la fuente de luz, y la dirección desde el vértice al punto de vista.  
Con un punto de vista infinito, la dirección entre él y cualquier vértice en la escena permanece constante. Un punto de vista local tiende a proporcionar resultados mas realistas, pero la dirección ha tenido que ser calculada para cada vértice, la ejecución se ve afectada negativamente con un punto de vista local. Por defecto se asume un punto de vista infinito.

Para cambiar a un punto de vista local:

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

Esta llamada localiza el punto de vista en (0.0,0.0 ,0.0) en coordenadas de vista. Para volver a un punto de vista infinito pasamos como argumento GL\_FALSE.

- Cálculos de iluminación en dos caras.  
Los cálculos de iluminación se realizan para todos los polígonos, ya sean éstos con cara-frontal o cara-posterior. Cuando de un objeto sólo se ve la cara frontal no tiene sentido iluminar la cara trasera. Pero si se realiza un corte al objeto la superficie interior podría visualizarse y podríamos iluminar la parte posterior.

Para cambiar a una iluminación por dos caras:

```
glLightModeli(LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

**OpenGL invierte las normales a la superficie para polígonos con caras posteriores; esto significa que las normales a la superficie de las caras posterior y anterior miran hacia el viewer mas que apuntar alejándose. Como resultado todos los polígonos son iluminados correctamente.**

Para volver a un a desactivar la iluminación por dos caras se pasa como argumento GL\_FALSE.

### Habilitar la iluminación

Con OpenGL necesitas habilitar o deshabilitar explícitamente la iluminación. Si no está habilitada el color actual es mapeado al vértice pero los cálculos de normales, fuentes de luz, modelo de iluminación y propiedades de material no se producen. Para ello:

```
glEnable(GL_LIGHTING);
```

Para deshabilitarlo se llama a glDisable con GL\_LIGHTING como argumento.

También es necesario habilitar cada fuente de luz que se defina, después de haber especificado los parámetros para esa fuente.

## 4.4.- SELECCIONAR PROPIEDADES DEL MATERIAL

1. Las propiedades del material de los objetos en la escena vienen definidas por: el color ambiente, difuso y especular, el brillo, y el color de cualquier luz emitida. La mayoría de las propiedades del material son conceptualmente similares a las estudiadas para las fuentes de luz. El comando utilizado es glMaterial\*().

```
void glMaterial{if}[v] (GLenum face, GLenum pname, TYPE param )
```

Especifica la propiedad del material a usar en los cálculos de iluminación. El parámetro *face* puede ser `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK` para indicar que a que cara del objeto aplicar el material. La propiedad se establece a través del argumento *pname* y los valores con que la característica *pname* se establece a través del argumento *param*, este último parámetro puede ser un puntero a un grupo de valores o un único valor dependiendo de la versión que utilicemos con vector (v) o con entero (i). La versión sin vector trabaja solo para establecer `GL_SHININESS`. `GL_AMBIENT_AND_DIFFUSE` permiten establecer los colores del material ambiente y difuso simultáneamente para el mismo valor `RGBA`.

Valores por defecto para el parámetro *pname* de `glMaterial*()`.

**Tabla 4.1.**

Nombre del parámetro	Valor por defecto	Significado
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	color ambiente del material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	color difuso del material
GL_AMBIENT_AND_DIFUSE		color ambiente y difuso del material
GL_SPECULAR	(0.0,0.0, 0.0, 1.0)	color especular del material
GL_SHININESS	0.0	exponente especular
GL_EMISSION	(0.0, 0.0,0.0, 1.0)	color de emisión del material
GL_COLOR_INDEXES	0.0	ambiente, difusa y índices de color especular

### Reflejo difuso y ambiente

Los parámetros `GL_DIFFUSE` y `GL_AMBIENT` establecidos por `glMaterial*()` afectan a el color de la luz difusa y ambiente que refleja un objeto.

El reflejo difuso determina el color que se percibe del objeto. Éste se ve afectado por el color de la luz difusa que incide sobre el y el ángulo de la luz relativa a la dirección normal. La posición del punto de vista no afecta al reflejo difuso.

El reflejo ambiente afecta completamente al color del objeto. El reflejo difuso es más vivo donde un objeto es iluminado directamente, el reflejo ambiente se nota más donde el objeto no recibe iluminación directa. El reflejo ambiente total de un objeto se ve afectado por la luz ambiente global y la luz ambiente individual de fuentes de luz. Igual que el reflejo difuso la posición del punto de vista no afecta al reflejo ambiente.

Para objetos del mundo real, el reflejo difuso y ambiente son del mismo color, por ejemplo:

```
GLfloat mat_amb_diff[]={0.1, 0.5, 0.8, 1.0}
glMaterialfv(GL_FRONT_AND_BACK,
GL_AMBIENT_AND_DIFFUSE,mat_amb_diff);
```

### Reflejo especular

El reflejo especular desde un objeto produce brillos. La cantidad de reflejo especular visto por un observador depende de la localización del punto de vista, es más vivo a lo largo del ángulo directo de reflejo. OpenGL permite establecer el color RGBA de un reflejo especular con `GL_SPECULAR`, y controlar el tamaño y luminosidad con `GL_SHININESS` (valores en el rango [0.0, 128.0]).

```
GLfloat mat_specular[]={1.0, 1.0, 1.0, 1.0}
GLfloat low_shininess[]={5.0}
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
```

## Emisión

Normalmente se usa esta propiedad para simular bombillas u otras fuentes de luz en la escena.

Un color verdoso para GL\_EMISSION:

```
GLfloat mat_emission[]={0.3, 0.2, 0.2, 0.0}  
glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
```

## Otra alternativa para cambiar las propiedades de un material

Esta técnica intenta minimizar los costos de ejecución asociados a `glMaterial*()`, utiliza la función `glColorMaterial()`:

```
void glColorMaterial{if}[v] (GLenum face, GLenum mode )
```

Establece propiedad del material especificado por *mode* y de las caras del material especificadas por *face* para trazar el valor del color actual todo el tiempo. Un cambio al color actual (usando `glColor*()`) inmediatamente lo actualiza para el material especificado. El parámetro *face* puede ser se trazado por puede ser L\_FRONT, GL\_BACK, GL\_FRONT\_AND\_BACK (por defecto). El parámetro *mode* puede ser GL\_AMBIENT, GL\_DIFFUSE, GL\_AMBIENT\_AND\_DIFFUSE (por defecto), GL\_SPECULAR o GL\_EMISSION.

Después de llamar a `glColorMaterial()`, necesitas llamar a `glEnable()` con GL\_COLOR\_MATERIAL como parámetro, para después cambiar el actual color con `glColor()` ( o otras propiedades del material con `glMaterial*()`).

```
glColorMaterial(GL_FRONT, GL_DIFFUSE);  
glEnable(GL_COLOR_MATERIAL);  
glColor3f(0.2,0.5,0.8);  
// Dibujo de objetos  
glColor3f(0.9,0.0,0.2);  
//Dibujo de otros objetos aquí  
glDisable(GL_COLOR_MATERIAL);
```

Se puede usar `glColorMaterial()` cuando necesitas cambiar un parámetro simple de un material para la mayoría de los vértices de la escena.