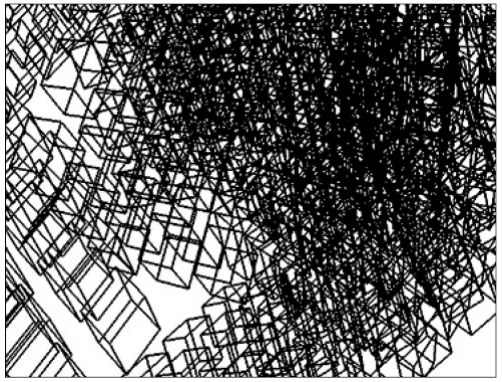
Iluminación

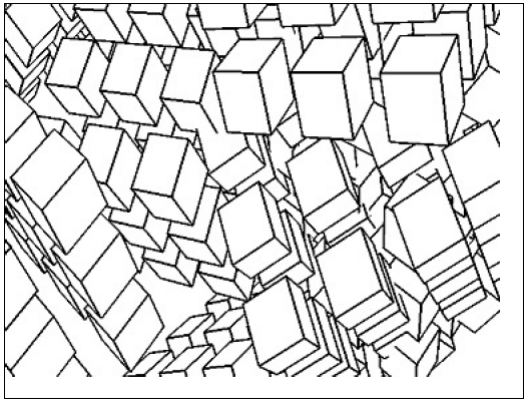
P. J. Martín, A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Renderización

- Renderización de escenas
- Renderización y rasterización
- Jerarquía en los niveles de realismo
  - Renderización basada en armazón de hilos

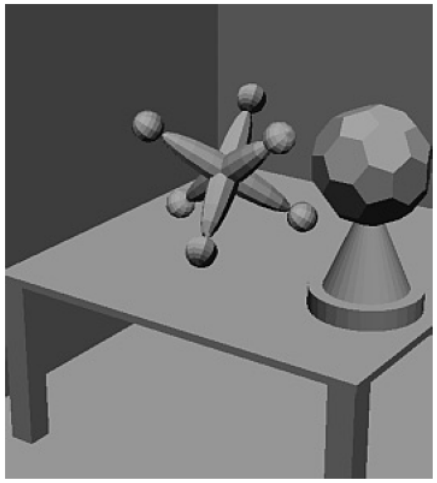


  - Renderización basada en armazón de hilos con eliminación de superficies ocultas

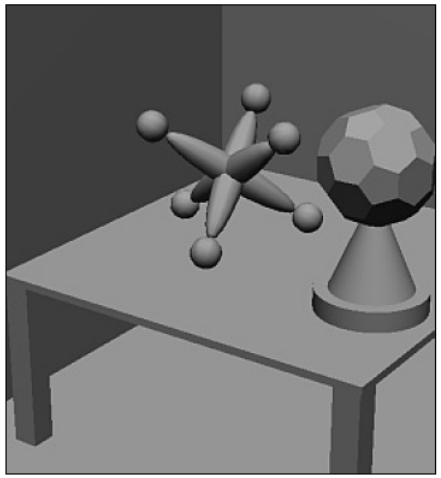


Renderización

- Iluminación plana (flat shading)

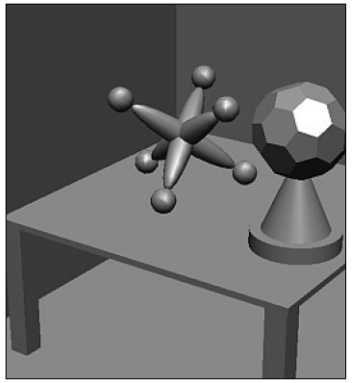


- Iluminación suave (smooth shading)

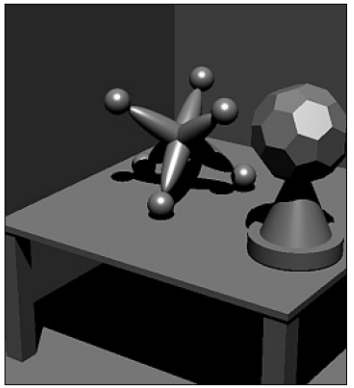


Renderización

- Iluminación con efectos especulares



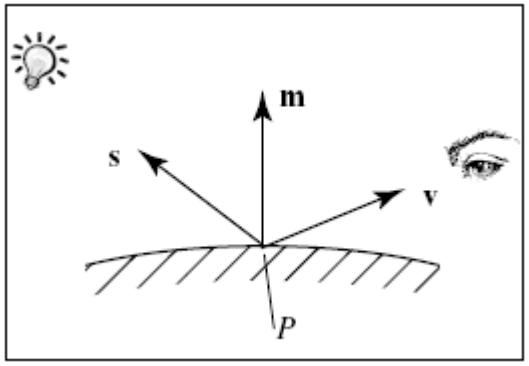
- Iluminación con sombras



- Iluminación con texturas

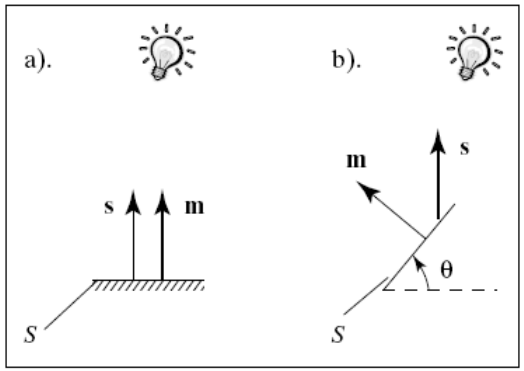
Shading models

- Un modelo de sombreado dicta cómo se refleja y dispersa la luz, cuando incide sobre una superficie.
- Simplificación con respecto al modelo físico de propagación de la luz (absorción de luz, transmisión de la luz a través de los objetos, intensidad de la luz emitida).
- Fenómenos principales que se tienen en cuenta cuando la luz incide en una superficie:
  - La dispersión difusa, responsable del color de la superficie
  - El reflejo especular, responsable del material de la superficie
- Ingredientes geométricos:
  - Vector normal (**m**) a la superficie en el punto P
  - Vector desde (**v**) P al ojo de la cámara
  - Vector desde (**s**) P a la fuente de luz

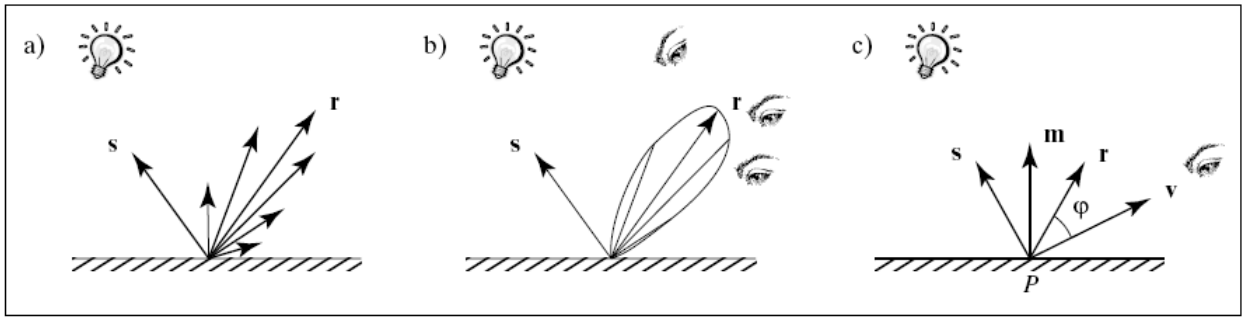


Componentes difusa, especular

- Dispersión difusa
  - Luz que vuelve a irradiar la superficie en todas las direcciones.
  - Depende del ángulo que forman los vectores **s** y **m**.
  - Modelo de Lambert.



- Reflexión especular
  - Luz que refleja la superficie.
  - El vector de máxima reflexión **r(s, m)** coincide con el rayo reflejado. La reflexión depende pues del ángulo que forman los vectores **r** y **v**.
  - Modelo de Phong.



… y ambiente

- Luz que alcanza una superficie aunque no esté expuesta a la fuente de luz.
- Las tres componentes (difusa, especular y ambiente) se modelan como ternas de colores RGB.
- Para fuentes de luz lejanas, el vector **s** varía poco, luego la componente difusa cambia poco a lo largo de la superficie.
- Para fuentes de luz cercana, los vectores **r** y **s** varían mucho, luego la componente especular cambia mucho a lo largo de la superficie.

Fuentes de luz en OpenGL

- El modelo de sombreado en OpenGL se define con el comando:

```
glShadeModel(GL_FLAT);     //Para el sombreado plano
glShadeModel(GL_SMOOTH);    //Para el sombreado suave
```

- OpenGL permite definir hasta 8 fuentes de luz

```
GL_LIGHT0, GL_LIGHT1, …
```

- El modo de iluminación se activa/desactiva con los comandos:

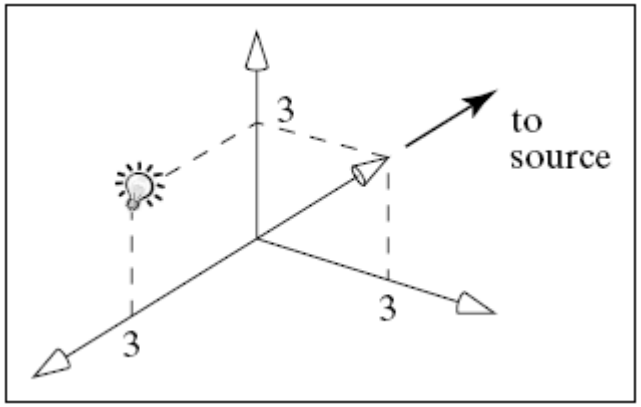
```
glEnable(GL_LIGHTING)/glDisable(GL_LIGHTING)
```

- Una fuente de luz particular se enciende/apaga con los comandos:

```
glEnable(GL_LIGHT0)/glDisable(GL_LIGHT0)
```

Fuentes de luz en OpenGL

- Las fuentes de luz en OpenGL son de dos tipos:
  - Direccionales (o remotas): se asume que sus rayos de luz llegan paralelos a la escena. Se dan mediante el vector del origen a la fuente de luz
  - Posicionales (o locales): los objetos de la escena son más o menos iluminados según su exposición. Se dan mediante el punto en el que se encuentra la fuente de luz.



Fuentes de luz en OpenGL

- La posición de una fuente de luz se define con el comando:

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

glLightfv(GL_LIGHT0, GL_POSITION, v);
```

- La forma en que OpenGL distingue el carácter de una fuente de luz es por la cuarta componente de la posición con que se definen. Si es 1, la fuente es local, mientras que si es 0, es direccional.
- Por defecto, todas las fuentes de luz tienen posición (0, 0, 1, 0), es decir, son direccionales y miran a la parte negativa del eje Z.

Fuentes de luz en OpenGL

- Para definir en OpenGL las componentes difusa, especular y ambiente de una fuente de luz se usan los comandos:

```
GLfloat amb0[]={0.2, 0.4, 0.6, 1.0};
GLfloat dif0[]=…;
GLfloat esp0[]=…;
glLightfv(GL_LIGHT0, GL_AMBIENT, amb0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif0);
glLightfv(GL_LIGHT0, GL_SPECULAR, esp0);
```

- Los valores por defecto son los siguientes:

(0, 0, 0, 1) para la componente ambiente de todas las luces (represente la oscuridad)

(1, 1, 1, 1) para las componentes difusa y especular de la luz `GL_LIGHT0` (representa el mayor brillo)

(0, 0, 0, 1) para las componentes difusa y especular de las restantes luces

- En OpenGL se puede definir la atenuación de las fuentes de luz especificando el factor de atenuación.
- El factor de atenuación de una fuente real de luz se modela mediante la fórmula:

$$factor\quad atenuacion = \frac{1}{k_c + k_l d + k_q d^2}$$

$$k_c = GL\_CONSTANT\_ATTENUATION$$
$$k_l = GL\_LINEAR\_ATTENUATION$$
$$k_d = GL\_QUADRATIC\_ATTENUATION$$

siendo *d* es la distancia.

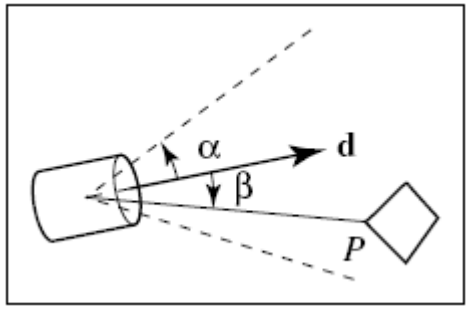
- En OpenGL se pueden definir las tres constantes de atenuación mediante los siguientes comandos:

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, …);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, …);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, …);
```

- Por defecto, $k_c=1$, $k_l=0$, $k_q=0$, lo que supone que el factor de atenuación es 1 y, por tanto, que no habrá atenuación.

Focos

- En OpenGL se pueden definir focos de luz como fuentes de luz posicionales caracterizadas por:



  es decir, por una dirección de emisión (el vector **d**), y una amplitud de emisión (el ángulo α).

- Los puntos que se encuentran fuera del cono de emisión de luz no reciben luz del foco.

  Los puntos que se encuentran dentro del foco de emisión reciben una cantidad de luz que varía según el ángulo β en un factor que es una potencia ε del valor cos(β).

Focos

- Los comandos de OpenGL para especificar un foco son:

```
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 4.0);
GLfloat dir[]={2.0, 1.0, -4.0};
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dir);
```

  que especifican un foco con α=45º, ε=4 y **d**=(2, 1, -4).

- Por defecto, α=180º, ε=0 y **d**=(0, 0, -1), que supone que el foco apunta al lado negativo del eje Z, es omnidireccional y la luz se distribuye uniformemente por todo el cono de emisión.

Tipos de luces

- **Luces estáticas**
  - No cambian su posición en la escena.
  - La posición debe restablecerse cada vez que cambie la matriz de modelado-vista.
  - La posición se fija pues en `display()`.
- **Luces dinámicas**
  - Luces que cambian de posición, independientemente de la cámara.
    - **Focos**.
    - La posición se fija después de (post-)multiplicar por la matriz que coloca el objeto que las contiene.
    - La posición se fija en el método `dibuja()` del objeto que las contiene.
  - Luces que se mueven con la cámara.
    - **Luz de minero**.
    - La posición se establece una vez, después de que la matriz de modelado-vista se ha fijado.
    - La posición se fija en `init()`.

Materiales

- OpenGL simula materiales por la forma en que reflejan la luz roja, verde y azul. Por ejemplo, una superficie verde bajo una luz roja resulta ser ...
- En OpenGL, los materiales se especifican con las mismas tres componentes (ambiente, difusa y especular) que las fuentes de luz.
- Cada componente del material indica cómo refleja la superficie, la correspondiente componente de la fuente de luz. Por ejemplo, componente especular alta de una superficie aparecerá brillante bajo una fuente de luz con componente especular alta.
- Como las componentes ambiente y difusa son las responsables del color, se suelen tomar iguales.
- Como la componente especular solo es responsable de los reflejos que emite la superficie, se suele tomar gris de forma que no altere el color de la luz incidente.

- Algunas de las componentes siguientes para simular materiales comunes son los siguientes coeficientes de reflexión:

Material	ambient: $\rho_{ar}, \rho_{ag}, \rho_{ab}$	diffuse: $\rho_{dr}, \rho_{dg}, \rho_{db}$	specular: $\rho_{sr}, \rho_{sg}, \rho_{sb}$	exponent: f
Black Plastic	0.0 0.0 0.0	0.01 0.01 0.01	0.50 0.50 0.50	32
Brass	0.329412 0.223529 0.027451	0.780392 0.568627 0.113725	0.992157 0.941176 0.807843	27.8974
Bronze	0.2125 0.1275 0.054	0.714 0.4284 0.18144	0.393548 0.271906 0.166721	25.6
Chrome	0.25 0.25 0.25	0.4 0.4 0.4	0.774597 0.774597 0.774597	76.8
Copper	0.19125 0.0735 0.0225	0.7038 0.27048 0.0828	0.256777 0.137622 0.086014	12.8
Gold	0.24725 0.1995 0.0745	0.75164 0.60648 0.22648	0.628281 0.555802 0.366065	51.2
Pewter	0.10588 0.058824 0.113725	0.427451 0.470588 0.541176	0.3333 0.3333 0.521569	9.84615
Silver	0.19225 0.19225 0.19225	0.50754 0.50754 0.50754	0.508273 0.508273 0.508273	51.2
Polished Silver	0.23125 0.23125 0.23125	0.2775 0.2775 0.2775	0.773911 0.773911 0.773911	89.6

- El comando de OpenGL para definir el material es el siguiente:

```
glMaterialfv(I, II, III);
```

donde:

I puede tomar alguno de los siguientes valores:

- `GL_FRONT`, para definir coeficientes de reflexión (ambiente, difusa o especular) para caras frontales
- `GL_BACK`, para caras traseras
- `GL_FRONT_AND_BACK`, para caras frontales y traseras

III es un vector **a** definido previamente, con tres componentes en el modelo RGB y una componente α:

```
GLfloat a[]=(…, …, …, 1.0);
```

- Por último, `II` puede ser:
  - `GL_AMBIENT`, para definir la componente ambiente
  - `GL_DIFFUSE`, para definir la componente difusa
  - `GL_SPECULAR`, para definir la componente especular
  - `GL_AMBIENT_AND_DIFFUSE`, para definir iguales y a la vez las componentes ambiente y difusa
  - `GL_EMISSION`, para simular objetos que emiten luz. Su luz no ilumina, sólo los hace parecer más brillantes. El valor que tiene por defecto es (0,0,0,1)
  - `GL_SHININESS,` para hacer parecer más brillante la superficie de un objeto. Es un valor entre 0 y 128. El valor 128 da aspecto metálico.

Materiales por registro de color

- Las componentes se pueden establecer mediante lo que se llama el *registro de color*. Para ello se usa el comando `glColor()` en lugar del comando `glMaterial()`.
- El código para definir los materiales usando el registro de color es el siguiente:

```
glEnable(GL_COLOR_MATERIAL);
   //Se activa el registro de color
glColorMaterial(I, II);
glColor3f(…, …, …);
glBegin(GL_TRIANGLES);  //O lo que sea
   glVertex3f(…, …, …);
glEnd();
```

donde **I** se define como antes, y **II** también, pero sin GL_SHININESS.

Modelos de iluminación locales vs globales

- Modelo local
  - El color de un vértice depende del material y de las fuentes de luz
  - No se tienen en cuenta sombras ni luces reflejadas
  - Solo interacción objeto-fuentes de luz
  - Fenómenos ópticos comunes (sombras, reflejos, caústicas) difíciles de reproducir
- Modelo global
  - El color de un vértice depende del material y de la luz que le llegue de las fuentes de luz
  - Interacción objeto-fuentes de luz y también objeto-objeto
  - Fenómenos ópticos comunes no son tan costosos de reproducir
  - Ray tracing, radiosidad

Componentes globales del modelo de iluminación en OpenGL (I)

- Color de luz ambiente global

```
GLfloat amb[]={…, …, …, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);
```

  El valor por defecto es (0.2, 0.2, 0.2, 1.0). La escena es pues siempre visible, a menos que no haya fuentes de luz y esta componente ambiente global se haya definido negra.

- Punto de vista de la cámara

```
glLightModeli(GL_LIGHT_LOCAL_VIEWER,GL_FALSE);
```

  El segundo parámetro es un booleano que especifica si el punto de vista es remoto o no.

Color en OpenGL

- Para asignar color a las primitivas de OpenGL se puede usar la luz o el color.
- El comando de OpenGL para asignar color a los vértices es:

```
glColor3f(…, …, …);
```

  que especifica el color **primario** de todos aquellos vértices cuya declaración aparezca después de este comando y mientras no aparezca otro.
- En OpenGL 1.4 y superiores se puede especificar también el color **secundario**, que es una forma de añadir color especular y que se añade después de aplicar la textura.

```
glSecondaryColor3f(…, …, …);
```

  No tiene componente alfa. Para usarlo ha de activarse con el comando:

```
glEnable(GL_COLOR_SUM);
```

Componentes globales del modelo de iluminación en OpenGL (II)

- Sombreado de ambos lados de las caras de una malla, invirtiendo normales para las caras traseras

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE,GL_TRUE);
```

  El valor por defecto es `GL_FALSE`.

- Aplicación de la luz especular antes o después de las texturas

```
glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL,
              GL_SEPARATE_SPECULAR_COLOR);
```

  Este comando produce un color primario (sin las componentes especulares de las luces) y otro secundario (con ellas); combina el primero con la textura y después añade el segundo. Por defecto, las texturas se aplican después de la luz.