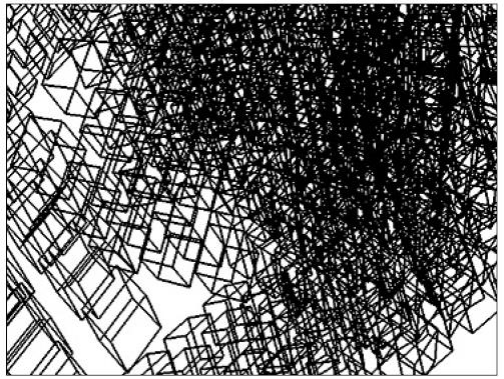
Iluminación

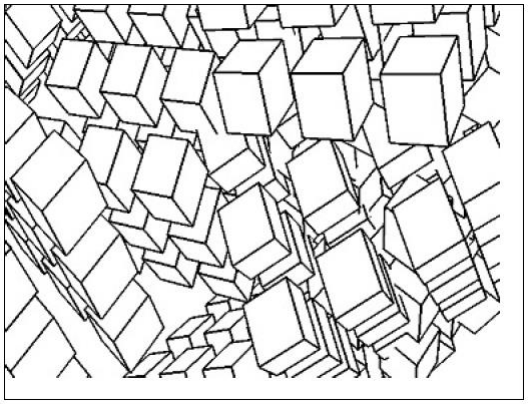
A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Renderización

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

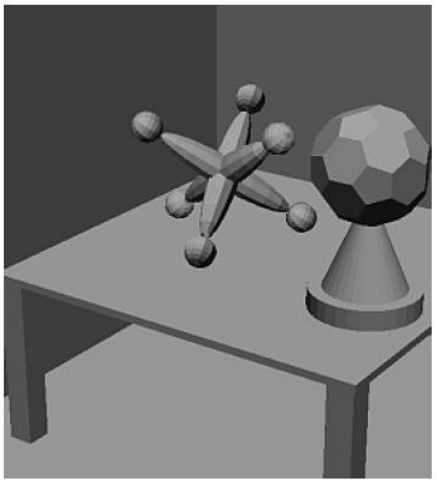


  - Renderización basada en armazón de hilos con eliminación de superficies ocultas

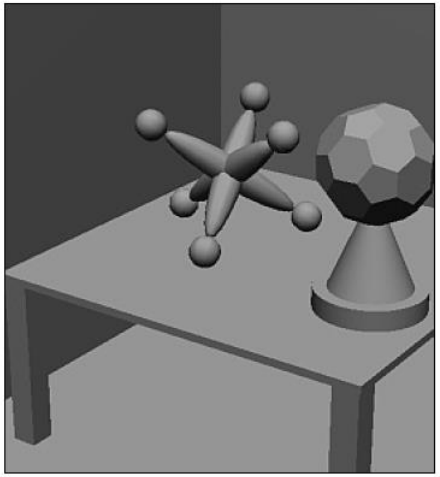


Renderización

- Iluminación plana (flat shading)

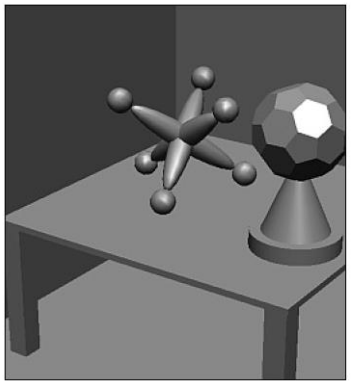


- Iluminación suave (smooth shading)

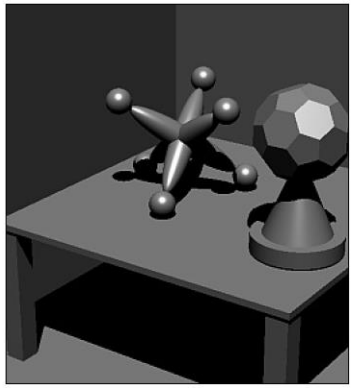


Renderización

- Iluminación con efectos especulares



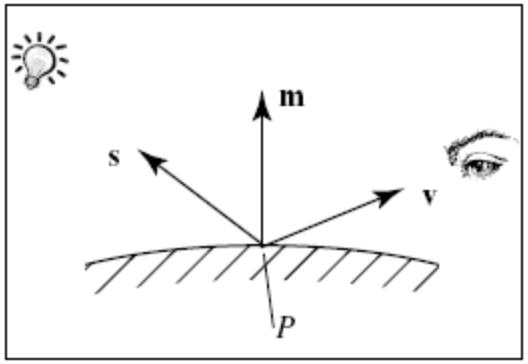
- Iluminación con sombras



- Iluminación con texturas

Lighting models

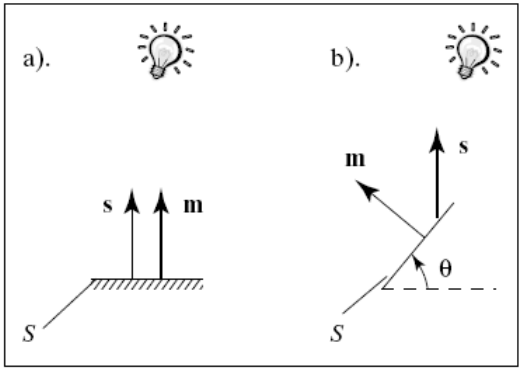
- Un modelo de iluminación dicta cómo se refleja y dispersa la luz, cuando incide sobre una superficie.
- Simplificación con respecto al modelo físico de propagación de la luz (absorción de luz, transmisión de la luz a través de los objetos, intensidad de la luz emitida).
- Fenómenos principales que 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, ...

- Dispersión difusa
  - Luz que, habiendo incidido sobre una superficie, se vuelve a irradiar desde ella, en todas las direcciones.
  - Su intensidad depende del ángulo que forman los vectores **s** y **m**.
  - Para su implementación se usa el modelo de Lambert:
    - `dvec3 diffuse = max(0, cos θ)*diffLight*diffMaterial`

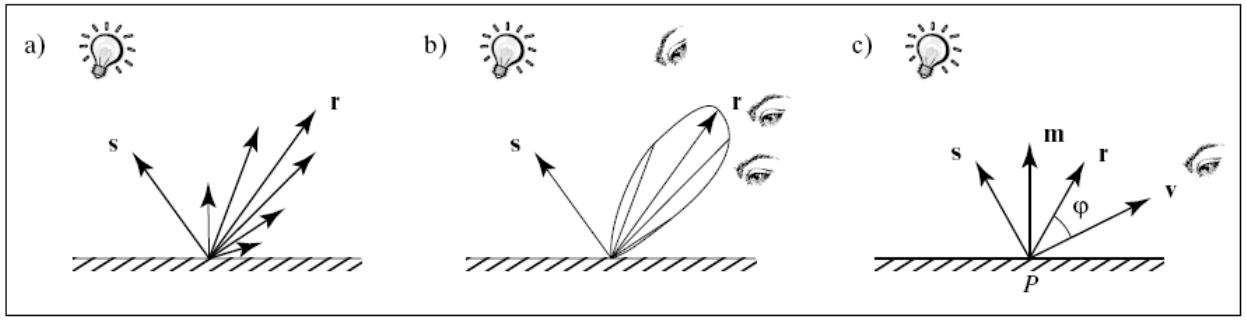
    donde `cos θ = dot(m, s)` (si los vectores están normalizados)
  - Para fuentes de luz lejanas, el vector **s** varía poco, luego la componente difusa cambia poco a lo largo de la superficie



… especular

- Reflexión especular
  - Luz que refleja la superficie.
  - El vector de máxima reflexión **`r(s, m)`** coincide con el rayo reflejado. La intensidad de la luz reflejada depende pues del ángulo que forman los vectores **`r`** y **`v`**.
  - Para su implementación se usa el modelo de Phong:
    - **`dvec3 specular = max(0, cos`** ^{f} **`ϕ)*specLight*specMaterial`**

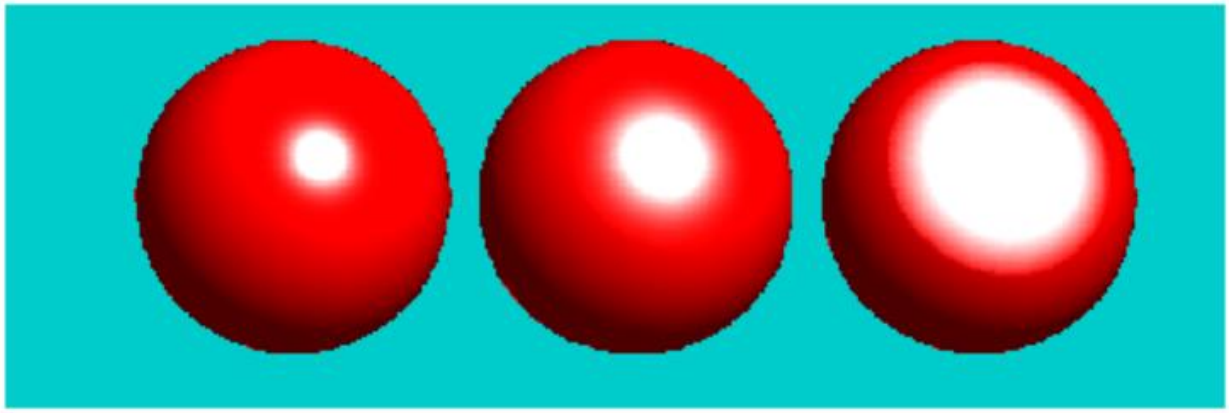
    donde **`cos ϕ = dot(r, v)`** (si los vectores están normalizados) y **`f=1,…,∞`** (espejo) (es un atributo del material)
  - Para fuentes de luz cercanas, los vectores **`r`** y **`s`** varían mucho luego la componente especular cambia mucho a lo largo de la superficie.



… especular

- Modelo de Phong:
  - **dvec3 specular = max(0, cos $^{f}\phi$)*specLight*specMaterial**

  donde **cos ϕ = dot(r, v)** (si los vectores están normalizados) y **f=1,…,∞** (espejo) (es un atributo del material)
- Ejemplo para algunos valores de **f**



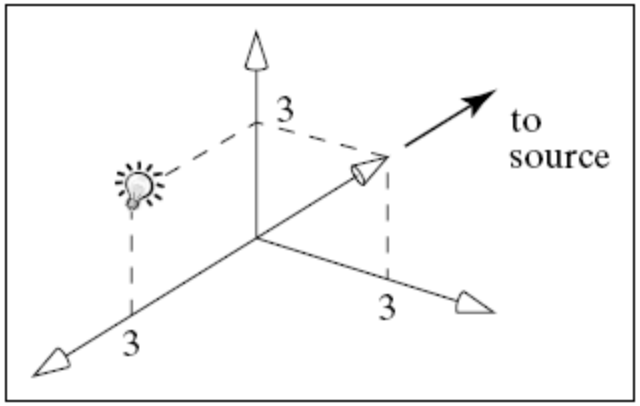
… y ambiente de las fuentes de luz

- Luz que alcanza una superficie aunque no esté expuesta a la fuente de luz.
- Es una luz que ilumina a todos los objetos por igual.
- Se implementa siguiendo un modelo simple:
  - `dvec3 ambient = ambLight * ambMaterial`

- **RESULTADO FINAL**
- El color resultante de **una** fuente de luz es la combinación de las tres componentes obtenidas:
  - `dvec3 finalColor = diffuse + specular + ambient`
- El color final de un punto de la superficie de un objeto es la suma de los colores finales por cada fuente de luz, teniendo en cuenta además otros factores como la atenuación de las fuentes, la emisividad del material, la pertenencia a conos de luz.

Fuentes de luz en OpenGL

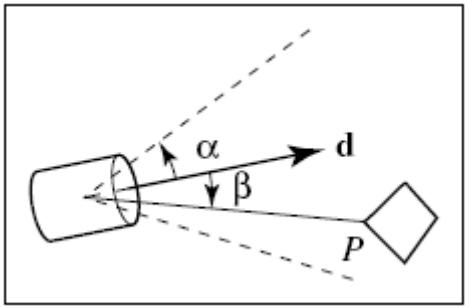
- Las fuentes de luz en OpenGL son de dos tipos:
  - Direccionales (o remotas): se supone que están infinitamente lejos y que, por tanto, sus rayos de luz llegan paralelos a la escena. Por su lejanía infinita se supone que no se atenúan. Se especifican mediante el vector que va **del origen a la fuente de luz.**
  - Posicionales (o locales): se supone que están localizadas en un punto y que, por tanto, los objetos de la escena son más o menos iluminados según su exposición. Se especifican mediante el punto en el que se encuentra la fuente de luz. En estos casos, el vector **`s`** en un punto del objeto se obtiene por **`s = (lightPosition-pointPosition).normalize()`** y se puede conocer también la distancia de la fuente al punto por lo que estas fuentes admiten atenuación.



Fuentes de luz en OpenGL

- En OpenGL se admiten dos tipos de fuentes de luz posicionales:
  - Focos: Están caracterizados 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 que determinan estos dos parámetros no reciben luz del foco.

    Los puntos que se encuentran dentro del cono de emisión reciben una cantidad de luz que varía según el ángulo β en un factor que es una potencia ε del **cos** β. Cuanto mayor es ε, más se concentra la luz del foco,



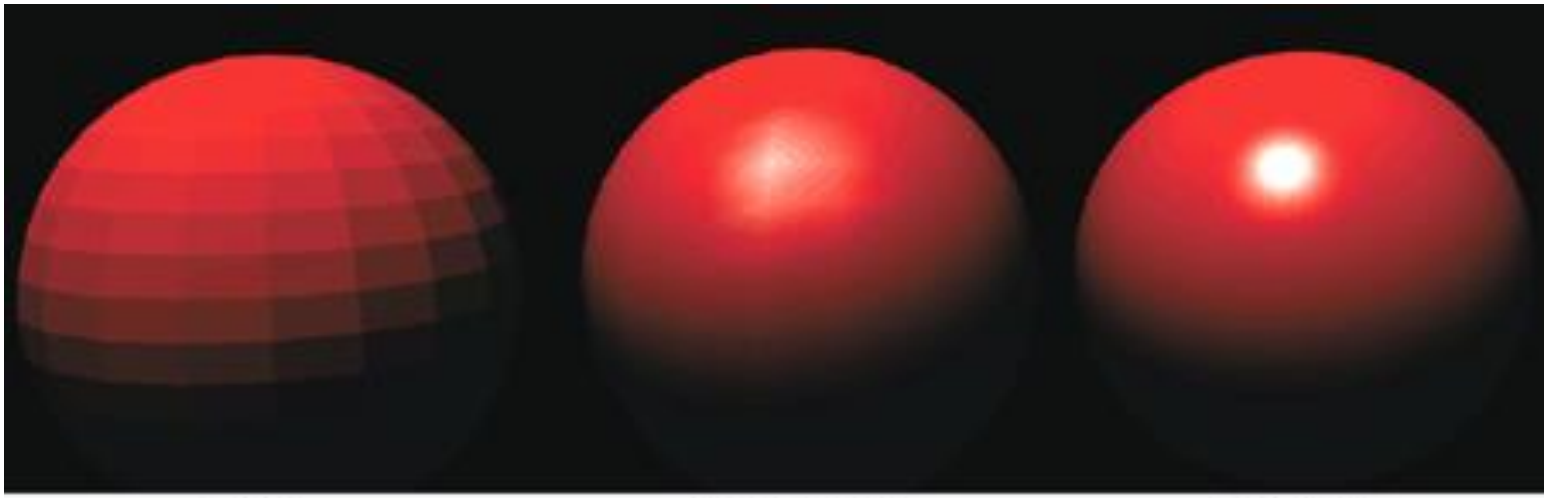
  - Omnidireccionales: El resto de las fuentes de luz locales, que se caracterizan por emitir luz por igual, en todas direcciones.

Shading models en OpenGL

- El modelo de matizado o tonalidad (shading model) en OpenGL se especifica con los comandos:

```
glShadeModel(GL_FLAT);    // Para el matizado plano
glShadeModel(GL_SMOOTH); // Para el matizado suave (o de Gouraud)
                          // Es el que se activa por defecto
```

Existe también el modelo de matizado de Phong, solo en las versiones de OpenGL 2.x y posteriores, que requiere ser definido por el usuario.



Las fuentes de luz en OpenGL

- La posición de la fuente de luz 0 se define con el comando:

```
glm::fvec4 v={3.0, 2.0, 1.0, 1.0};

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

- Las coordenadas de una fuente de luz posicional son mundiales.
- La forma en que OpenGL distingue si una fuente de luz es remota o local es por la cuarta componente de la posición con que se especifica. Si es distinto de 0, la fuente es local; si es 0, es remota.
- 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.

Las fuentes de luz en OpenGL

- Para definir en OpenGL las componentes difusa, especular y ambiente de la fuente de luz 0 se usan los comandos:

```
glm::fvec4 amb0={0.2, 0.4, 0.6, 1.0};
glm::fvec4 dif0=…;
glm::fvec4 esp0=…;
glLightfv(GL_LIGHT0, GL_AMBIENT, value_ptr(amb0));
glLightfv(GL_LIGHT0, GL_DIFFUSE, value_ptr(dif0));
glLightfv(GL_LIGHT0, GL_SPECULAR, value_ptr(esp0));
```

- Los valores por defecto son los siguientes:
  - (0, 0, 0, 1) para la componente ambiente de todas las luces (representa 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

Focos en OpenGL

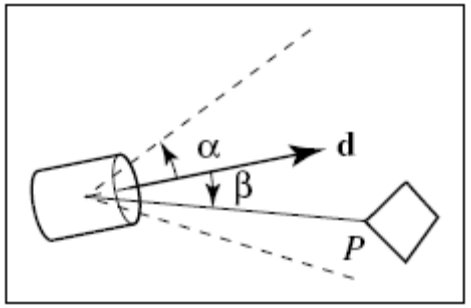
- Los comandos de OpenGL para especificar un foco en la luz 0 son los siguientes (los valores que aparecen son ejemplos):

```
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 4.0);
glm::fve3 dir={2.0, 1.0, -4.0};
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, value_ptr(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.

- Los focos, como luces posicionales que son, admiten ser atenuados.



Atenuación de las fuentes de luz locales en OpenGL

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

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

$$k_c = GL\_CONSTANT\_ATTENUATION$$

$$k_l = GL\_LINEAR\_ATTENUATION$$

$$k_d = GL\_QUADRATIC\_ATTENUATION$$

  donde *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, **kc=1**, **kl=0**, **kq=0**, luego, para estos valores, el factor de atenuación es 1 y, por tanto, no habrá atenuación. En realidad, la atenuación de luces remotas es posible, pero no lo es cambiar los valores por defecto de las constantes.

Comandos para tratar las luces en OpenGL

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

  `glEnable(GL_LIGHTING)/glDisable(GL_LIGHTING)`

- OpenGL permite definir hasta **`GL_MAX_LIGHTS`** fuentes de luz

  `GL_LIGHT0, GL_LIGHT1, …, GL_MAX_LIGHTS-1`

  cumpliéndose que **`GL_LIGHTi = GL_LIGHT0 + i`**.

- Cada fuente de luz de OpenGL tiene un identificador (**`id`**) interno (que se guarda en la tarjeta gráfica GPU)

- El identificador es un entero (**`GLuint`**) que varía sobre el rango **`GL_LIGHT0, …, GL_LIGHT0 + GL_MAX_LIGHTS - 1`**, donde **`GL_LIGHT0`** y **`GL_MAX_LIGHTS`** son dos constantes **`GLuint`**.

- Una fuente de luz particular (con identificador **`id`**) se enciende/apaga con los comandos:

  `glEnable(id)/glDisable(id)`

Comandos para tratar las luces en OpenGL

- El calculo del coseno del ángulo formado por dos vectores, necesario en el manejo de las componentes difusa y especular de las luces, requiere que los vectores estén normalizados. El comando:

```
glEnable(GL_NORMALIZE);
```

  normaliza todos los vectores normales antes de usarlos

- El método **`setGL()`** de la clase **`Scene`** define el entorno para manejar las luces en la escena:

```
// Se pone el fondo azul
glClearColor(0.7, 0.8, 0.9, 1.0);
// Se activa el test de profundidad
glEnable(GL_DEPTH_TEST);
// Se activan las texturas, por si las hay
glEnable(GL_TEXTURE_2D);
// Se activa la iluminación
glEnable(GL_LIGHTING);
// Se activa la normalización de los vectores normales
glEnable(GL_NORMALIZE);
```

Las fuentes de luz en OpenGL

- La única luz hasta ahora en la práctica es la que aparece en el método de **`Scene sceneDirLight(Camera const&cam)`**. Por supuesto los dos primeros comandos (activación de la iluminación y de la fuente de luz) no es preciso hacerlo (solo) aquí

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glm::fvec4 posDir = { 1, 1, 1, 0 };
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd(value_ptr(cam.viewMat()));
glLightfv(GL_LIGHT0, GL_POSITION, value_ptr(posDir));
glm::fvec4 ambient = { 0, 0, 0, 1 };
glm::fvec4 diffuse = { 1, 1, 1, 1 };
glm::fvec4 specular = { 0.5, 0.5, 0.5, 1 };
glLightfv(GL_LIGHT0, GL_AMBIENT, value_ptr(ambient));
glLightfv(GL_LIGHT0, GL_DIFFUSE, value_ptr(diffuse));
glLightfv(GL_LIGHT0, GL_SPECULAR, value_ptr(specular));
```

Es una luz llamada **`GL_LIGHT0`** que es direccional (la cuarta componente de la posición es 0) y cuyos rayos llegan a la escena en una dirección **`(-1, -1, -1)`** (el vector del origen a la fuente de luz es **`(1, 1, 1)`**), de color blanco (componente difusa), de color negro en los objetos no enfrentados a ella (componente ambiente), que proporciona poca reflexión (componente especular). Además la fuente de luz es fija pues su posición se determina una vez con respecto a la posición de la cámara).

El modo ColorMaterial

- ¿Cómo reaccionan las entidades a las fuentes de luz?
- Hasta ahora se hacía estableciendo lo que se llama el *registro de color*
- El código para definir los materiales usando el registro de color es lo que aparece en el método **`render()`** de las entidades

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(cara, componenteDeLaLuz); *
glColor3f(…, …, …);
mesh->render();
GlColor3f(1.0, 1.0, 1.0);
glDisable(GL_COLOR_MATERIAL);
```

* **`cara`** es **`GL_FRONT`**, **`GL_BACK`** o ambas, y **`componenteDeLaLuz`** es **`GL_AMBIENT`**, **`GL_DIFFUSE`**, **`GL_SPECULAR`**, …

Materiales

- OpenGL simula materiales por la forma en que reaccionan ante la luz roja, verde y azul en cada componente (difusa, especular, ambiente) de la fuente de luz. Por ejemplo, en cuanto al color, una superficie con componentes difusa y ambiente verde (0, 1, 0) bajo una luz con componentes difusa y ambiente roja (1, 0, 0) resulta ser … (0*1, 1*0, 0*0) = (0, 0, 0)
- En consecuencia, en OpenGL, los materiales se especifican con las mismas tres componentes (ambiente, difusa y especular) que las fuentes de luz
- En el caso de los materiales, las componentes se llaman coeficientes de reflexión o reflectancias del material
- Cada coeficiente del material indica cómo reacciona la superficie ante la correspondiente componente de cada fuente de luz. Por ejemplo, una superficie con coeficiente de reflexión especular alto parecerá brillante bajo una fuente de luz con componente especular alta
- Como los coeficientes de reflexión ambiente y difuso son los responsables del color, y el color de una entidad no cambia se enfrente a una luz o no, en los materiales estos coeficientes se suelen tomar iguales
- Como el coeficiente especular solo es responsable de los reflejos que tiene la superficie, en los materiales este coeficiente se suele tomar gris de forma que el color de la superficie no se vea alterado por el color de la luz incidente

- Coeficientes de reflexión para simular materiales comunes (junto con el valor del exponente para usarse en la componente especular):

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(cara, componenteDeLaLuz, value_ptr(a));
```

donde:

cara 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

a es un vector con tres componentes RGB y una componente α, para especificar los coeficientes de reflexión:

```
glm::fvec4 a = {…, …, …, 1.0};
```

- **`componenteDeLaLuz`** 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. Es el exponente que se usa como potencia en la reflexión especular

Componentes globales del modelo de luz en OpenGL

- Color de la 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 está pues siempre algo iluminada, a menos que no haya fuentes de luz y esta componente ambiente global se haya definido negra, en cuyo caso todo se verá negro salvo el color de fondo.

- Ojo de la cámara para la reflexión especular

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

El segundo parámetro es un booleano que especifica si el ojo es remoto o no. Por defecto es false. Si es true, el vector **v** al ojo que se tiene en cuenta en la reflexión especular pasa a ser (0, 0, 1) siempre.

Componentes globales del modelo de luz en OpenGL

- Coloreado de ambos lados de las caras de una malla

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

El valor por defecto es **GL_FALSE**. Es útil cuando se quiere que se muestren las caras traseras. OpenGL utiliza entonces como vector normal de una cara, el opuesto del de la cara frontal.

- Aplicación de la luz especular antes de las texturas o no

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

que hace que OpenGL:

- mantenga dos colores para cada vértice: primario (obtenido como se ha explicado, sin tener en cuenta componentes especulares) y secundario (obtenido teniendo en cuenta componentes especulares)
- combine la textura solo con el color primario, si hace el caso
- añada después el color secundario.

En OpenGL 1.4 y superiores el color **secundario** se puede fijar con

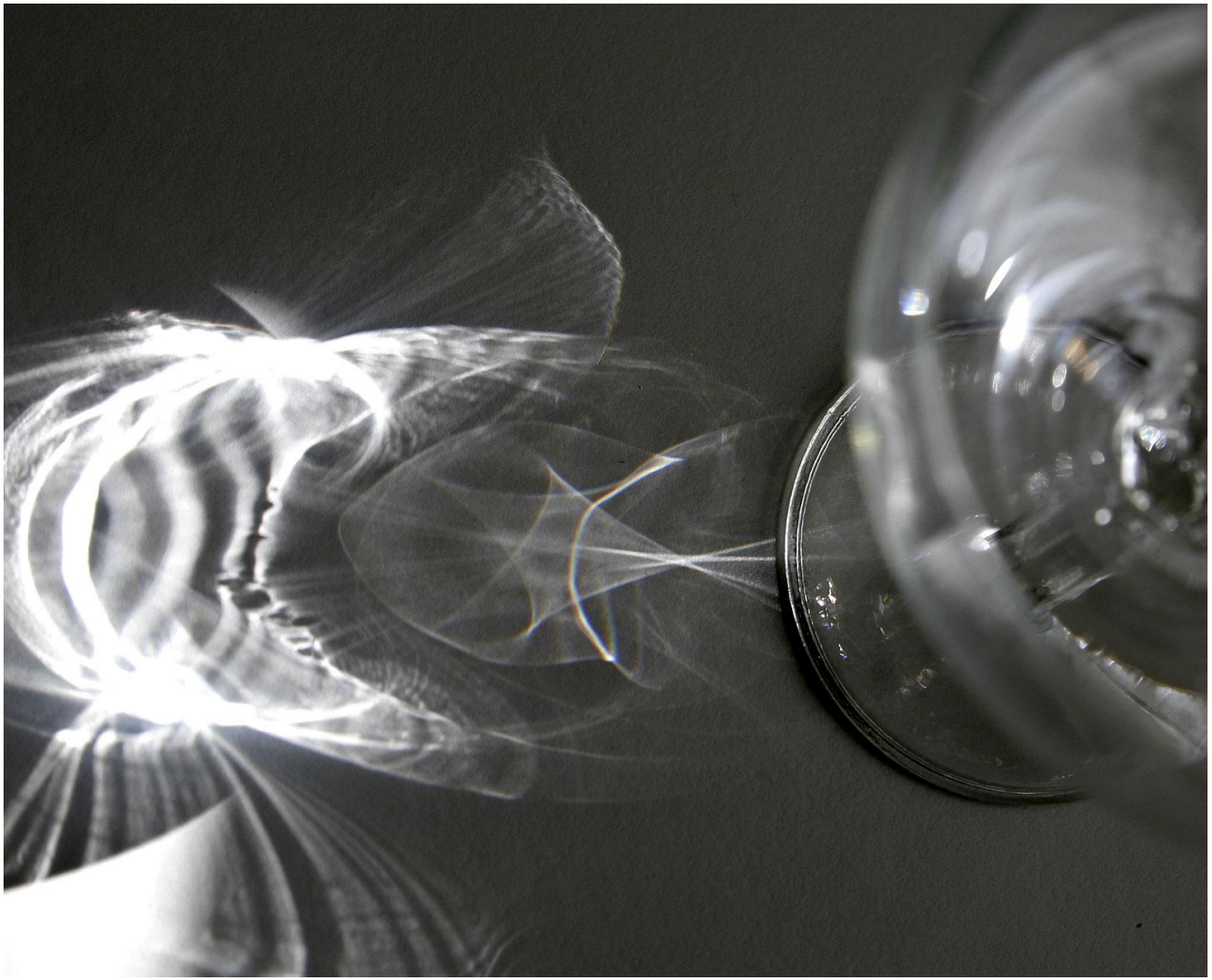
```
glSecondaryColor3f(…, …, …);
```

No tiene componente alfa. Se activa con **glEnable(GL_COLOR_SUM);**

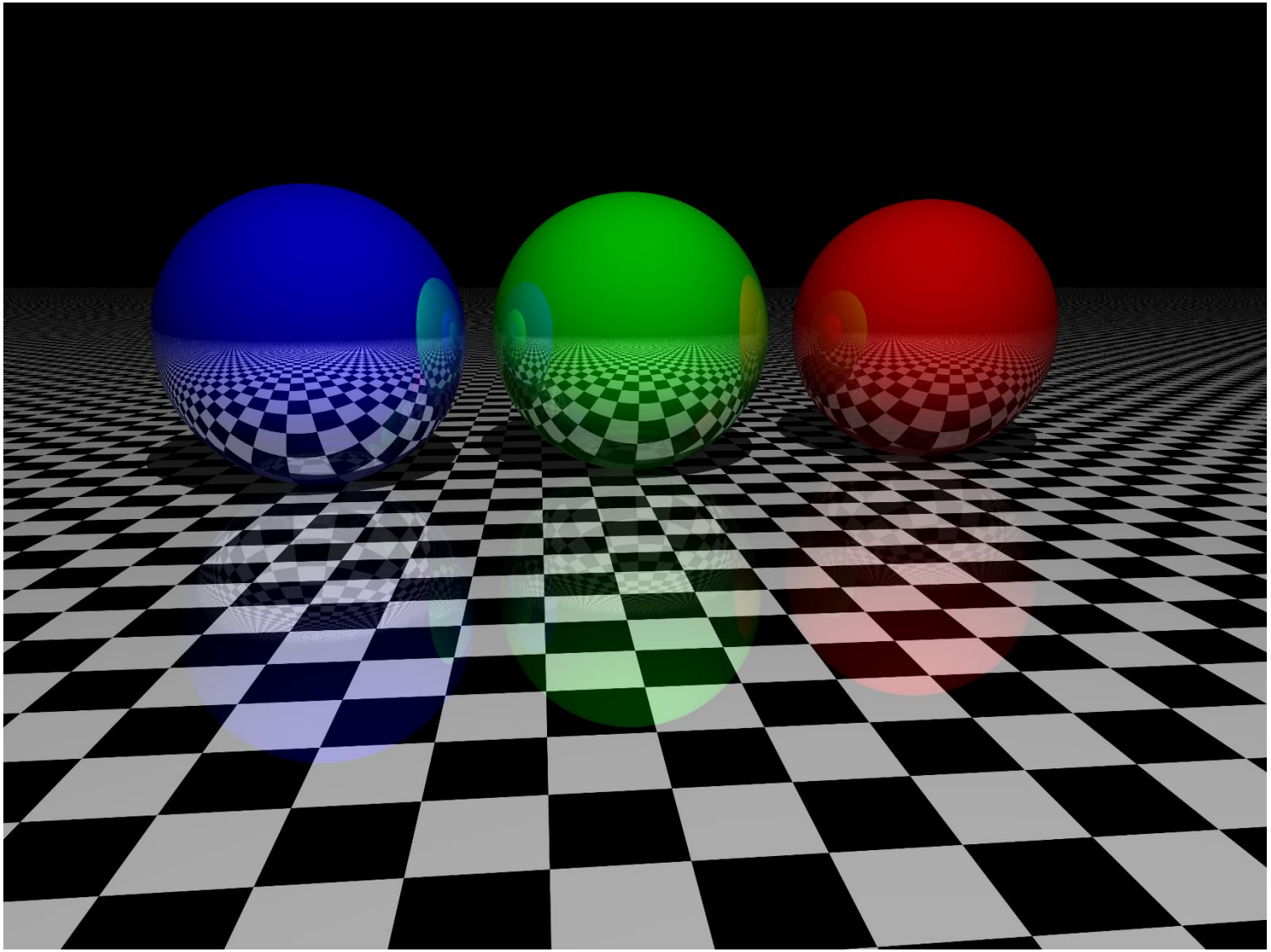
Modelos de iluminación locales vs globales

- 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, cáusticas) 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

Cáusticas



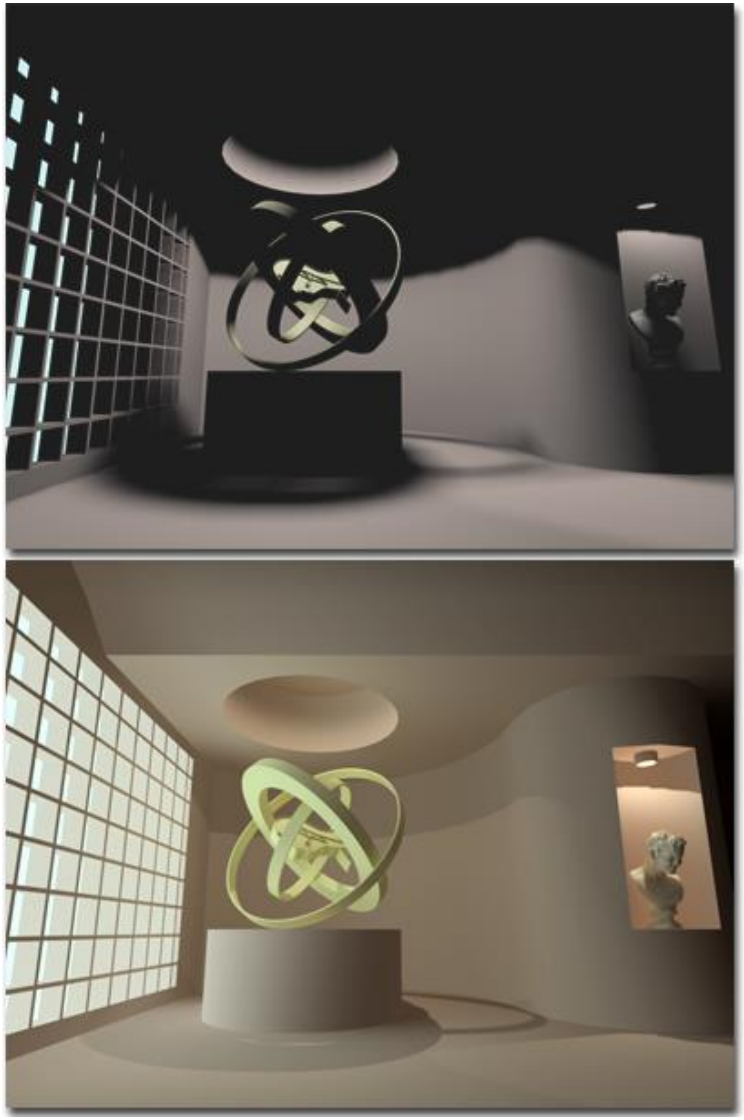
Reflejos



Sombras, reflejos y cáusticas

Modelo de iluminación por radiosidad

Modelo de iluminación por radiosidad



Modelo de iluminación por ray tracing

Modelo de iluminación por ray tracing