PRACTICA 2

Iluminación en GLSL

Parte opcional

Adrián David Morillas Marco y Guillermo Meléndez Morales
Diseño y Desarrollo de Videojuegos + Ingeniería de Computadoras

2ª Fuente de luz

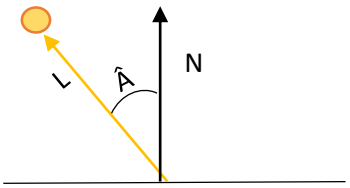
A la hora de crear una nueva luz nos basaremos en el Modelo de Phong, como se nos ha instruido. El modelo de Phong es una iluminación local, es decir, se basa en la interacción luz-objeto, por lo tanto los objetos no emiten luz. Se basa en obtener una iluminación igual a la suma de la iluminación ambiental, iluminación difusa e iluminación especular:

$$I_{Phong} = I_{ambiental} + I_{difusa} + I_{especular}$$

En este caso emplearemos una luz difusa. Una luz difusa es aquella cuyo ángulo de reflexión no es igual al ángulo de incidencia como en el caso de la luz especular. La iluminación difusa se divide en tres variables:

- Il: es la intensidad de la luz, esta se encuentra entre los valores 0 y 1.
- Kd: es el coeficiente de reflexión de la luz
- Angulo: es el ángulo que forma la luz con la normal, para ello emplearemos la siguiente normalización:
  - L: vector de incidencia de la luz
  - N: vector normal de la superficie

  El coseno de ambos vectores nos dará el ángulo



Para hallar la luz difusa emplearemos la siguiente operación:

$$I_{difusa} = Il * Kd * \cos(angulo) = Il * Kd * (N * L)$$

Lo primero que se hará es crear la luz para ello, crearemos una intensidad Il, y un punto donde ubicar la luz:

```
//intensidad difusa 2
vec3 Il2 = vec3 (1.0, 0.0, 0.0);    // la intensidad puede ser modificada para obtener un color en el foco
vec3 Pl2 = vec3 (-8.0, 0.0, 0.0); //posición de la luz, para ubicarla necesitamos un sistema de referencia
                                  //(supondremos que el sistema de referencia es el  sistema de la cámara)
```

Previamente en la parte guiada creamos un foco difuso y con él, creamos el coeficiente de reflexión Kd:

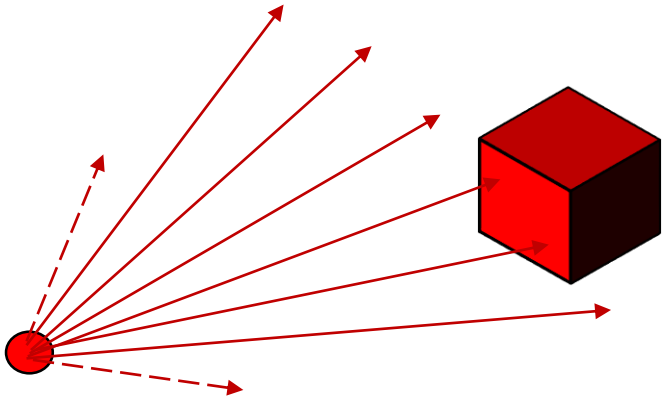
```
vec3 Kd;
Kd = texture (colorTex, texcoord). rgb;          //coeficiente difusa
```

A continuación nos dirigimos al método shade () donde "implementaremos los cálculos" y se los sumaremos a la variable color, la cual se encarga de ir sumando las iluminaciones para dar la iluminación de Phong:

- Calculamos el vector L de incidencia:
  ```
  vec3 L = normalize(vec3(view*vec4(Pl, 1.0)-pos));
  ```

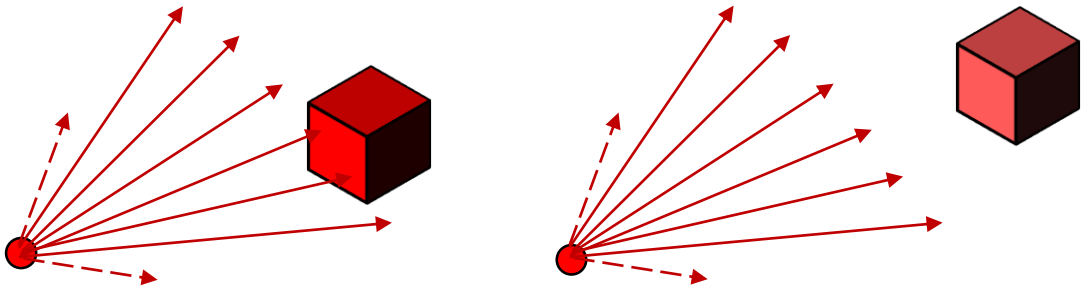
- A continuación sumamos la iluminación a la variable color. Para hallar el producto escalar de N y L emplearemos la función dot(x, y):
  ```
  color+= fatt*Il*Kd*dot(N,L);
  ```

El resultado será el siguiente, teniendo en cuenta que hemos escogido una iluminación de color roja



Atenuación de la luz

La atenuación de una luz es el desgaste de su iluminación a medida que su punto de luz se aleja.

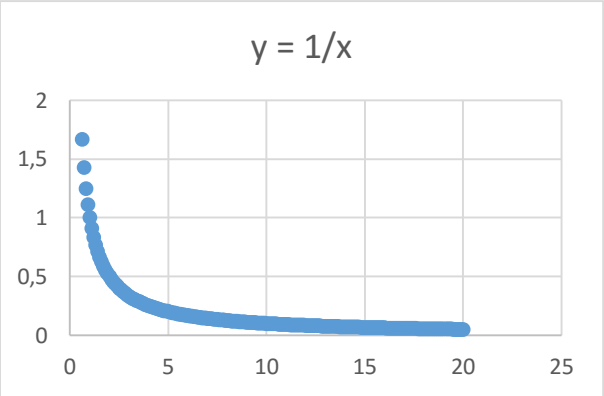


Pl (0.0, 0.0, -6.0) Pl (0.0, 0.0, -12.0)

Este efecto se obtiene complementando la constante de atenuación al cálculo de la luz, en nuestro caso en la luz difusa. La constante de atenuación tiene la siguiente formula:

$$fatt = \frac{1}{c_x + c_y * d + c_z * d^2}$$

Empezaremos por crear el vector C, el cual contiene las constantes de atenuación. Para su mayor comprensión observaremos el siguiente gráfico:

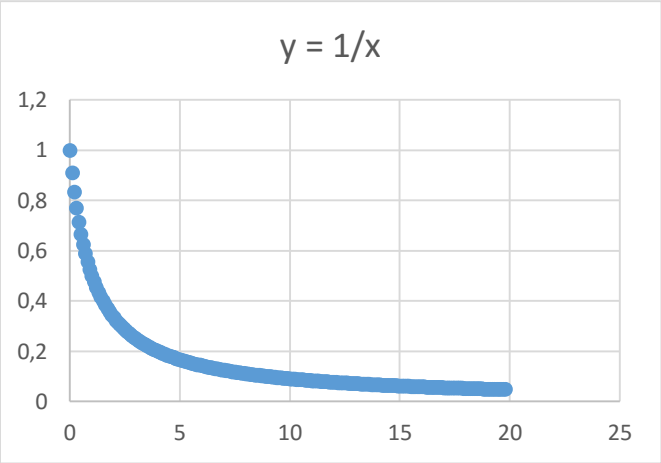


Cuanto mayor sea x menor es y, traduzcamos esto a nuestro caso, cuanto mayor sea la distancia menor será la iluminación. Nuestro problema son las indeterminaciones de este gráfico. Si nos ponemos a una distancia cero, resultará una indeterminación, al igual que si nos vamos a distancia inmensamente grande.

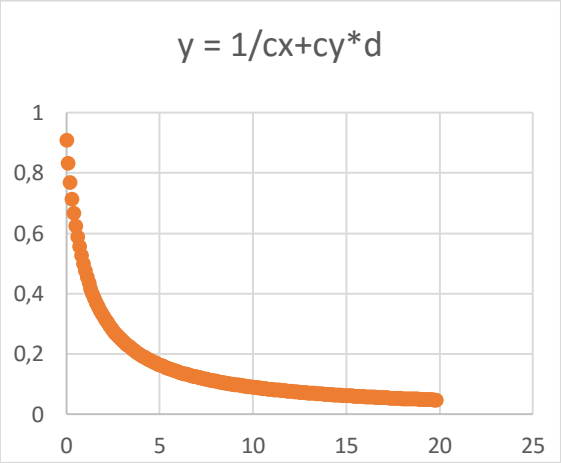
Para ello emplearemos las constantes de atenuación del vector C:

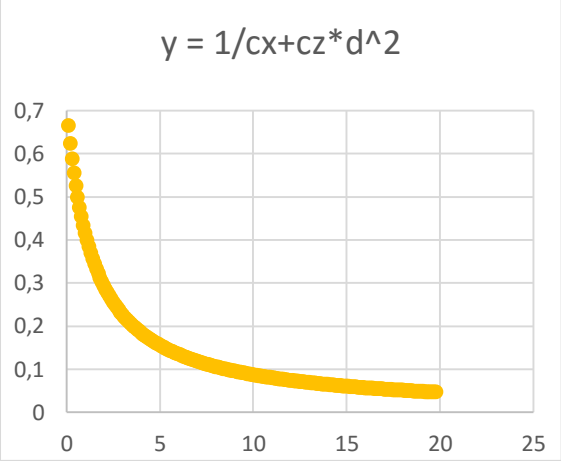
$$C = \begin{pmatrix} constante\ de\ atenuación \\ constante\ de\ atenuación\ lineal \\ constante\ de\ atenuación\ cuadrática \end{pmatrix}$$

Para evitar la indeterminación pondremos a la constante de atenuación (C.x) el valor 1, dando al siguiente gráfico:



Ahora ya no hay indeterminación pero se debería de poner algún valor a la constante lineal de atenuación o a la constante cuadrática de atenuación o ambas, para que no se quede constante a medida que cambie la distancia, valiendo siempre el coeficiente de atenuación 1. Para ello veremos qué pasa si añadimos una unidad a la constante lineal y qué pasa si añadimos una unidad a la constante cuadrática





Si añadimos una unidad a la constante lineal, se reduce mínimamente la gráfica, mientras que si aumento una unidad la constante cuadrática aumenta excesivamente la gráfica. Para nuestro beneficio hemos colocado el valor 0,3 a la constante lineal y un valor nulo a la constante cuadrática.
Esto en código glsl se traduce a:

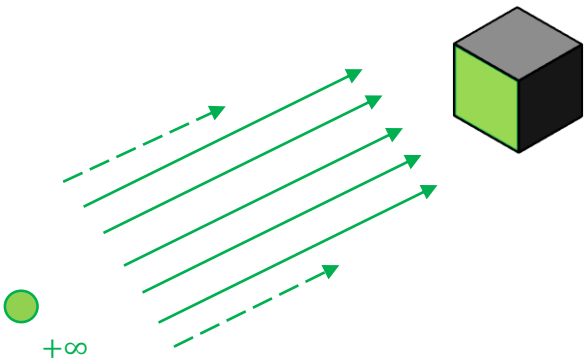
```
vec3 C = vec3 (1.0, 0.3, 0.0);          // k1 valdrá 1 porque sí, k2 es la atenuación lineal y k3 es la
                                        //atenuación cuadrática
float d = length (Pl);                  // distancia desde el vértice hasta su incidencia
float fatt = 1/ (C.x + C.y*d + C.z*pow(d, 2));

// las últimas dos líneas de código las emplearemos en cada luz difusa que empleemos, modificando el
// nombre de la variable d y fatt, para evitar errores
```

Finalmente añadimos la atenuación a nuestra luz difusa

```
color+= fatt*Il*Kd*dot(N,L);
```

Luz direccional y luz focal

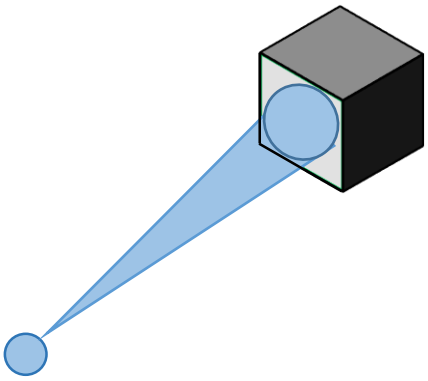
La luz direccional es una luz difusa la cual en vez de tener haces de luz a todas direcciones, tiene haces de luz en un sentido. Esto sería el caso de una luz difusa la cual se encuentra a gran distancia como el Sol. Según el principio de dualidad, dos rectas paralelas se cortan en el infinito. Si tomamos este postulado desde el otro sentido, dos rectas que se cortan, en el infinito acaban siendo paralelas, ese sería el caso de los haces de luz de nuestra luz direccional.



Para implementar la luz direccional simplemente habrá que crear una nueva iluminación difusa y una posición, y a continuación habrá que cambiar la luz de un punto a un vector, para ello tenemos la coordenada homogénea, la cual cambiaremos de un 1.0 a un 0.0:

```
//intensidad direccional
vec3 Il3 = vec3 (0.0, 1.0, 0.0);   // la intensidad puede ser modificada para obtener un color en el foco
vec3 Pl3 = vec3 (0.0, 0.0, 6.0);  // posición de la luz, para ubicarla necesitamos un sistema de
                                  // referencia (supondremos que el sistema de referencia es el sistema de
                                  // la cámara)
vec3 L3 = normalize(vec3(view*vec4(Pl3, 0.0)));
color+= fatt3*Il3*Kd*dot(N,L3);
```

La luz focal es una luz difusa que en vez de tener haces a todas direcciones, tiene haces a un numero de direcciones, es decir dentro de un rango. Esta iluminación forma un cono de luz:



Para implementarla emplearemos la siguiente formula:

$$si\ \vec{D} * (\overrightarrow{-L}) > cos\varepsilon \rightarrow I_{focal} = \left(\frac{(\vec{D} * (\overrightarrow{-L})) - cos\varepsilon}{1 - cos\varepsilon} \right)^{n} * I_{diff} \quad siendo\ I_{diff} = I_l * Kd$$

$$si\ \vec{D} * (\overrightarrow{-L}) < cos\varepsilon \rightarrow I_{focal} = 0$$

Siendo D la dirección del foco, el ángulo, el ángulo de apertura y n el coeficiente de atenuación del foco. Comenzaremos por crear una nueva luz y una nueva posición, y a continuación crearemos L, D, n y el ángulo:

```
vec3 Il4 = vec3 (0.3, 0.5, 0.0);   // la intensidad puede ser modificada para obtener un color en el foco
vec3 Pl4 = vec3 (0.0, 0.0, 5.0);  //posición de la luz, para ubicarla necesitamos un sistema de referencia
                                  //(supondremos que el sistema de referencia es el sistema de la cámara)
vec3 L4 = normalize(vec3(view*vec4(Pl4, 1.0)-pos));

vec3 D = normalize(vec3(view*vec4(Pl4, 1.0)));

float Ip;

float beta = 3.14159/10;        // angulo Epsilon
float no = 0.5;
```

A continuación efectuamos el primer calculo:

```
float calculo1 = dot ( D, -L4 );

float calculo2 = cos(beta);

float calculo = (calculo1 - calculo2) / (1 - calculo2)
```

Ahora se efectuará la condición $\vec{D} * (\overrightarrow{-L}) > cos\varepsilon$:

```
if ( calculo1 > calculo2 ){                    // si D*(-L) > cos (Epilon)
        Ip = pow (calculo , no);               // calculo ^ n
}else{
        Ip = 0;
}
```

Y finalmente aplicamos la iluminación a la variable color:

```
vec3 idiff = vec3 (Il4*Kd);

color+= Ip*idiff;
```