PRÁCTICA 2:
ILUMINACIÓN EN GLSL

GRÁFICOS POR COMPUTADOR
4º INGENIERÍA DE COMPUTADORES
CURSO 2018/2019

22 DE MAYO DE 2019

Ervi Saul Loachamin Llumiquinga
José Manuel Pérez Ocampos
Elena García-Morato Piñán

Parte Obligatoria

Para la realización de esta práctica, que implica la implementación de diferentes tipos de luz, hemos decidido emplear un struct para definir las variables que conforman cada una de ellas.

Así mismo, y ya que en el primer apartado necesitamos implementar varias luces, hemos optado por definir tres arrays de luces agrupen las de la misma naturaleza (puntuales, direccionales y focales) y que, al adaptar los algoritmos al manejo de éstos, nos permite aumentar su número fácilmente. El número de luces de cada tipo se define mediante constantes al inicio del shader de fragmentos:

```
#define NUM_PLIGHTS 2  //Luz Puntual
#define NUM_DLIGHTS 1  //Luz Direccional
#define NUM_SLIGHTS 1  //Luz Focal
```

Primer apartado

Tal y como acabamos de explicar, hemos creado un array de luces puntuales, que procesa y añade a la escena cada una de ellas mediante un bucle for, de forma que se generaliza el algoritmo que habíamos desarrollado en la parte guiada de la práctica para poder utilizar tantas luces como queramos (`shadePLight()`).

En nuestro caso hemos situado una delante del cubo, en el punto (0,0,1) y otra a su derecha, en el punto (3,0,-6).

Segundo apartado

Para añadir a nuestra implementación una nueva funcionalidad que atenúe la intensidad lumínica en función de la distancia del objeto a la fuente lumínica utilizamos el factor de atenuación, que se calcula en base a unas constantes definidas por el usuario (en nuestro caso, 0 ,1 y 0,1) y la distancia de la fuente al objeto:

```
float fatt = 1/(1 + 0 * d + 0.2*d*d);
```

donde: `float d = length (plights[i].Pl - Pp)`

Para ser añadida en la expresión final del modelo de iluminación:

```
c+=clamp (plights[i].Il*Kd*factor_difuso*fatt,0,1);
c+=clamp(plights[i].Il*Ks*factor_especular*fatt,0,1);
```

Tercer apartado

Para implementar luz direccional y luz focal, siguiendo la misma estructura empleada para la luz puntual, hemos empleado dos arrays, aunque en este caso solo hayamos implementado una luz de cada tipo.

En el caso de la luz direccional, cuya implementación se define en `shadeSLight()`, se utiliza la misma expresión que en el caso d la luz puntual, salvo porque en lugar de indicar la posición de la luz (Pl) se indica la dirección de ésta mediante un vector (L) y no hay factor de atenuación.

Para la luz focal, cuya implementación se define en `shadeSLight()`, también se utiliza la misma ecuación que en el caso de la luz puntual, pero con la diferencia de que solo se ilumina la parte que se encuentra dentro del ángulo de apertura de la luz (alpha), que se comprueba de la siguiente manera:

```
if(spot>cos(slights[i].alpha)
```

donde: `float spot = dot(-L, D);`

Parte Opcional

Niebla

Para implementar niebla que hace que los objetos se difuminen con el fondo a medida que se alejan de la cámara hemos utilizado el proceso descrito en el libro Real-Time Rendering, 3rd Edition (Capítulo 10, apartado 15 – Ecuaciones 10.13 y 10.16).

Para empezar, hemos declarado la niebla como un struct con dos variables:

```
struct Fog
{
	float Dens; //Densidad de la niebla
	vec3 Color; //Color de la niebla

};
```

Mientras que el valor de la densidad por motivos estéticos será de 0.1, el color de la niebla será negro, ya que lo que queremos es que el objeto se fusione con el fondo de la escena.

```
//Valor de las variables de la niebla
Fog fog = Fog(0.1, vec3(0,0,0));
```

Así, implementaremos la niebla acompañada de una luz puntual en shadeFog() utilizando la ecuación empleada para generar una luz puntual, pero añadiendo al final de ésta las siguientes declaraciones:

```
float f = clamp(exp(-(pow(fog.df*d,2.0))),0,1);
c= f*c + (1-f)* fog.cf;
```

- f (fogFactor) es un escalar definido entre 0 y 1 que se calcula utilizando la siguiente ecuación:

$$f = e^{-(d_f z_p)^2}$$

  Esta ecuación utiliza la densidad de niebla antes definida (fog.df) y Zp, una variable que toma el valor de la distancia que hay en el eje Z del fragmento a iluminar a la cámara (d).

- c es el color final del pixel, que obtenemos utilizando la siguiente ecuación:

$$\mathbf{c}_p = f\mathbf{c}_s + (1-f)\mathbf{c}_f$$

  En la que intervienen el color del pixel teniendo en cuenta solo la luz puntal (c), el color de la niebla (fog.cf) y el fogFactor (f) calculado anteriormente.