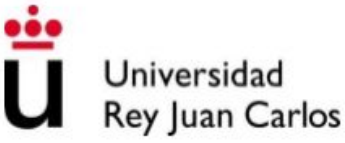

GRADO EN DISEÑO

Y

DESARROLLO DE VIDEOJUEGOS (MÓSTOLES) (2020-21)

Informática Gráfica

Práctica 2: Iluminación en GLSL

Participantes del grupo:
Pablo Burgaleta de la Peña
Raquel Mijarra Benítez
Adrián Mira García
Iván Pérez Ciruelos

Parte obligatoria

1. Ilumina el objeto con al menos 2 fuentes de luz.

Para iluminar el objeto con dos fuentes de luz hemos creado un shader de vértices y un shader de fragmentos (*shader.A1.vert y shader.A1.frag*).
En el shader de fragmentos hemos duplicado la luz puntual aprendida en la parte guiada cambiando la posición de la luz para apreciar ambas luces sobre el objeto.

2. Añade una nueva funcionalidad que atenúe la intensidad lumínica en función de la distancia del objeto a la fuente lumínica.

Para añadir la funcionalidad de atenuar la intensidad lumínica en función de la distancia hemos creado un shader de vértices y un shader de fragmentos (*shader.A2.vert y shader.A2.frag*).
Para este apartado hemos implementado diferentes fórmulas de las propuestas en clase:

- **Fórmula de OpenGl:**

Para implementar esta fórmula primero hemos definido los parámetros de *C0, C1* y *C2;* y el parámetro de d.

$$f_{dist}(d) = min\left(\frac{1}{c_1 + c_d d + c_d d^2}, 1\right)$$

- **Fórmula del modelo físico:**

Para implementar esta fórmula hemos definido la constante de d0 = 2 y reutilizado el parámetro de d.

$$f_{dist} = \left(\frac{d_0}{d}\right)^2$$

- **Fórmula de Unreal:**

Para implementar esta fórmula hemos definido la constante de épsilon = d0 = 2 y reutilizado los parámetros de d0 y d.

$$f_{dist} = \frac{{d_0}^2}{d^2+\epsilon}$$

- **Fórmula de CryEngine y Frostbite:**

Para implementar esta fórmula no hemos definido ninguna constante nueva, sólo hemos reutilizado el parámetro de d

$$f_{dist} = \left(\frac{d_0}{\max(d, d_{min})}\right)^2$$

Tras implementar las diferentes fórmulas de atenuación con la distancia hemos implementado la fórmula de la función ventana, declarando la constante d_{max}.

$$f_{win} = \left(\left(1 - \left(\frac{d}{d_{max}}\right)^4\right)^+\right)^2$$

Para la aplicación de ambas funciones hemos creado una función Fdw que multiplica Fdiff y Fwin.

Esta función Fdw la hemos aplicado de dos formas en nuestro programa:

- Afectando a todas las componentes de las luces:

Para que el cubo desaparezca completamente hemos multiplicado la función Fdw por cada componente (ambiental, difusa, especular y emisiva).

- Afectando sólo a las componentes difusa y especular:

En este caso hemos implementado la fórmula de las transparencias

$$I = I_a K_a + f_{dist}[I_{l,d} K_d (N * L) + I_{l,s} K_s (R * V)^n] + K_{emi}$$

Aplicando la función distancia solo a la componente difusa y especular.

3. Implementa los siguientes tipos de luz: luz direccional, luz focal.

Para añadir la implementación de una luz direccional y una luz focal hemos creado un shader de vértices y un shader de fragmentos (*shader.A3.vert y shader.A3.frag*)

Para implementar una luz direccional nos hemos basado en la implementación de una luz puntual pero modificando el valor de la dirección, convirtiendo la luz puntual aprendida en clase en una luz direccional.

Para implementar la luz focal hemos tenido que definir diferentes variables tales como: el ángulo de apertura del foco (angleap), la intensidad del foco (intensity), la dirección de la luz del foco (D) y el ángulo entre la dirección de la luz focal y la dirección de la luz (spotEffect). Para saber si el cubo está dentro del foco empleamos un bucle if → if(spotEffect > angleap), en el que calculamos las componentes ambiental, difusa, especular y emisiva si el cubo está dento, y las atenuamos en función de la distancia empleando la fórmula de Unreal del apartado anterior, además de la función ventana, también del apartado anterior.

Para este apartado hemos obtenido parte del código y la forma de solucionarlo en la siguiente página: [Iluminación - Tipos de Luces (acodigo.blogspot.com)]