OpenGL Shading Language

Tarea 07

Computación Gráfica

UNAM 2022-2

Gibran Zazueta Cruz

19/mayo/2022

1. Introducción

En el siguiente trabajo se realiza el renderizado de un objeto sobre una ventana utilizando OpenGL y OpenGL Shader language (GLSL).

Para crear la ventana se utiliza la biblioteca de QT y para importar objetos wavefront OBJ la biblioteca de assimp. El objeto elegido para renderizar es el conejo de stanford

La versión de OpenGL utilizada es la 4.5

1.1. GLSL

GLSL e un lenguaje de alto nivel, con syntax basado en *C*, que permite 'interferir' en el pipeline gráfico de OpenGL. Algunos tipos de shaders que permite implementar OpenGL son Vertex, Fragment, Tesselation y Geometry. EStos influyen en diferentes partes del pipeline. En este trabajo se implementa un vertex y fragment shader.

1.2. Escena y materiales

El objeto cuentan con 2 posibles materiales a seleccionar, estos se definen dentro del código como:

Material 1

- Ambiental = 0.0, 0.0, 0.0, 1.0,
- Difusa = 0.50, 0.50, 0.50, 1.0,
- Especular 0.70, 0.70, 0.70, 1.0
- $\rho = 32.0$.

Material 2

- Ambiental = 0.23125, 0.23125, 0.23125, 1.0,
- Difusa = 0.2775, 0.2775, 0.2775, 1.0,
- Especular 0.773911, 0.773911, 0.773911, 1.0

- $\rho = 89.6$.

La escena a renderizar cuenta con 2 luces (blanca y azul) y 3 cámaras. Las cámaras se definen con un FOV de 45

2. Estructura del código

Para almacenar la información del objeto se define la clase *CubeObject* y la función *importFile()*, definida en *functions.cpp*.

Esta función se llama desde *mainwindow.cpp*. La función recibe el path del archivo (como una cadena) y apuntadores al contenedor de vertices, indices de caras, normales y coordenadas de textura del objeto cubeObject, que es el objeto a renderizar en la escena. Dentro de esta función se utiliza la bilbioteca de assimp para importar el objeto.

En la función *initializeGL()* de *mainwindow* se declara la variable *program* de la clase *QOpenGLShaderProgram()*. Esta clase de QT maneja el compilado y linking de shaders escritos es GLSL. El código es agregado desde los archivos fuente $'shader.vert'$ y $'shader.frag'$.

En la función *paintGL* se utilizan matrices del tipo *QMatrix4x4* para contener las matrices de proyección, view y model de la cámara y objeto. Estas matrices se pasan al Vertex shader (como uniform mat4) para realizar la proyección en perspectiva. Por otro lado al Fragment shader se asignan los valores relacionados con la posición y color de las luces y la información del material actual del objeto.

Fragment Shader
Se implementa sombreado de phong utilizando las ecuaciones de luz difusa y especular. El shader recibe del vertex shader los valores interpolados de las normales y la posición de cada fragmento. También recibe del programa la información de las luces y del material. Las ecuaciones se implementan para cada una de las 2 luces de la escena.

3. Ejecutar el programa

En la carpeta de build se puede ejecutar el programa con el archivo GLSLRendering-Run. Desde la consola de comandos de linux:

bash

```
./GLSLRendering-Run
```

En la carpeta principal está el código fuente. Para generar el ejecutable primero se genera el Makefile con

bash

```
qmake GLSLRendering.pro
```

Después se construye el proyecto con *make*

4. Instrucciones de uso

Para cambiar entre las camaras se utilizan las teclas de los numeros

- "1". Cambia a la cámara 1
- "2". Cambia a la cámara 2
- "3". Cambia a la cámara 3
- "4". Cambia a la cámara 4

Para encender y apagar la luz azul se presiona la tecla ***L***. La luz inicia encendida.

Para activar y desactivar el sombreado de Phong se presiona la tecla ***P***. El sombreado inicia activo

Para cambiar entre materiales se utiliza la tecla ***8*** para el material 1 y la tecla ***9*** para el material 2.

5. Programa en ejecución

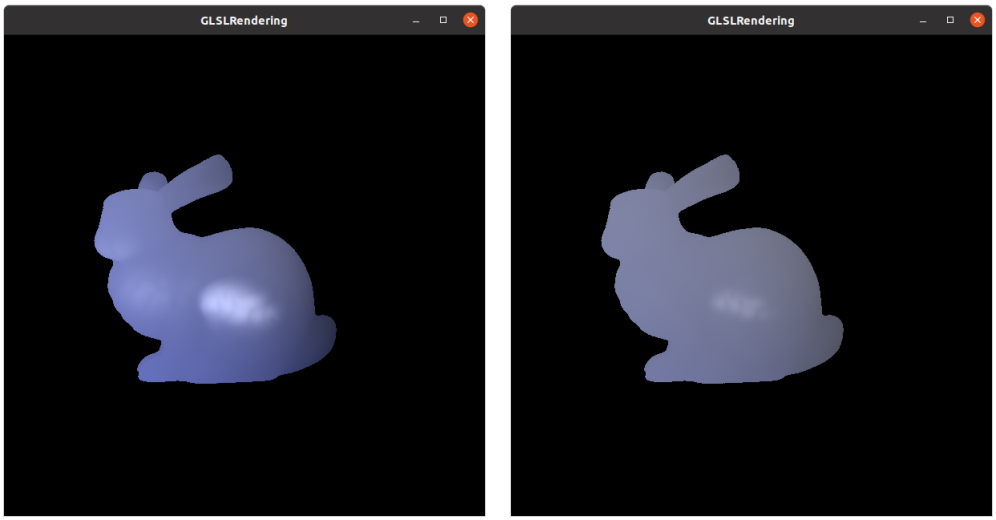

Figura 1: Renderizado cámara 1, materiales 1 y 2

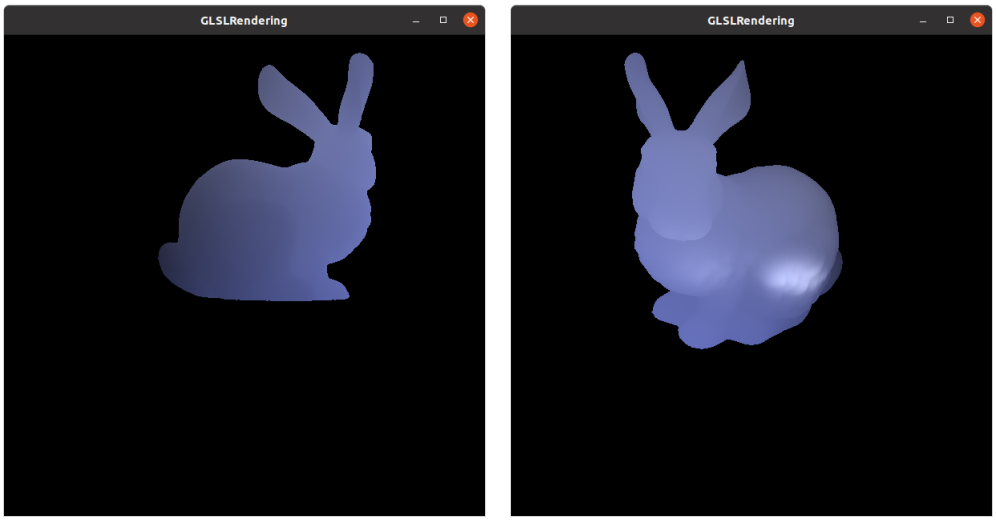

Figura 2: Renderizado cámara 2, 3 . Material 1