Ejercicio 4

Graficación por Computadora

May 15, 2019

Usando el programa generado en el ejercicio anterior (carga de modelos), el alumno le proporcionará la iluminación al objeto que se cargó.

1 Creación de luces

Antes de poder iluminar los objetos se necesita crear la luz que los iluminará, con la versión de OpenGl (con un pipeline fijo) que estamos utilizando sólo se pueden crear 8 luces. Por esto, cada luz tiene un identificador `GL_LIGHT`i donde la i va de 0 a 7.

Se debe especificar la posición donde estará nuestra luz con `glLightfv()` usando la macro `GL_POSITION` (también podemos especificar el componente ambiental `GL_AMBIENT` y difuso `GL_DIFFUSE`).

Recuerden encender la iluminación (`glEnable(GL_LIGHTING)`) y la luz que van a usar (ej. `glEnable(GL_LIGHT0)`). Juegen con estas funciones hasta que tengan un sombrado deseado

Para ver los parametros de cada una de estas funciones les recomiendo revisarlas en la documentación de OpenGl en *www.khronos.org*.

2 Calculo de normales

Uno de los modelos (el del oso) no tiene normales, por lo que tendremos que calcularlas nosotros, se tendrá una función llamada `calcNormals()` que recibe las caras y los vértices (o nada si es que son variables globales), y calculan sus normales de la siguiente manera:

Para cada cara f del modelo, obtenemos sus vértices v_1, v_2 y v_3, ahora para cada vértice (en este ejemplo, v_1) necesitamos calcular un vector ortogonal a los vectores $(v_2 - v_1)$ y $(v_3 - v_1)$, para esto usamos el producto cruz, al final normalizamos este último vector ortogonal, por lo que para encontrar la normal n del vértice v_1, sólo debemos calcular lo siguiente:

$$n = \frac{(v_2 - v_1) \times (v_3 - v_1)}{\|(v_2 - v_1) \times (v_3 - v_1)\|}$$

donde $\|(v_2 - v_1) \times (v_3 - v_1)\|$ es la norma del producto cruz.

Necesitamos guardar una normal por cada vértice, por lo que necesitamos un espacio (un arreglo) para guardarlo en orden respecto al orden de los vértices, si el vértice tiene más de una normal lo podemos sobreescribir sin problemas. Recuerden que puede que el vector normal les salga invertido (i.e. va hacia el interior del modelo), si eso sucede entonces sólo cambiamos el orden del producto cruz.

3 Tipos de Iluminación

Pedimos que se pueda cambiar el modelo de iluminación de Gouraud (`GL_SMOOTH`) a Flat (`GL_FLAT`) y de regreso usando la función `glShadeModel()`. Para esto, con presionar una tecla se debería poder

intercambiar entre los dos. También estaría bién poder rotar el modelo para ver como interactua con la luz.

4 Dibujado

Antes de dibujar a nuestro modelo necesitamos especificar las propiedades de su material, para esto usamos la función `glMaterialf()` en la cual damos los valoes para el componente especular (`GL_SPECULAR`), su brillo (`GL_SHININESS`), etc. En nuestro caso sólo queremos calcular la iluminación para la cara forntal `GL_FRONT`.

Por último por cada vértice de cada cara primero llamaremos su normal correspondiente y se la pasaremos a `glNormal3f()` y luego dibujaremos ese vértice como siempre.

Si pueden cargar el modelo de la estatua con sus normales, pueden aplicarle la iluminación con lo anterio, de otro modo tendrán que calcular sus normales primero.

5 Entrega

Se entregará un comprimido .zip con el Makefile y los archivos main.c y/o main.h comentados/documentados, además agregarán un README.txt donde especifiquen su nombre completo, número de cuenta y, si se desea, algún comentario sobre el ejercicio. Subir al Moodle el comprimido con el siguiente formato:

<Apellido Paterno><Apellido Materno>_04.zip