Informática Gráfica y Visualización
Prácticas - 2016/2017
Práctica Nº 4 – Iluminación y Texturas

Objetivos de la práctica:

- Iluminación de escenas utilizando luces y propiedades de material
- Mapeado básico de texturas

Sesiones de laboratorio: **14 de noviembre (corrección de apartados A) a D)) y 21 de noviembre de 2016 (corrección de apartados E) a G))**
Evaluación: **0.6 puntos sobre 10**
Material de teoría: **transparencias del tema 4: 21 a 26 y 44 a 50**

Material a entregar: fichero **pr4.zip** con los **ficheros .cpp y .h** proporcionados en la plantilla, modificados tal y como se pide en los diferentes apartados de la práctica

Recuerda llevar al aula lápiz y papel para dibujar geometría y hacer cálculos

PRÁCTICA Nº 4

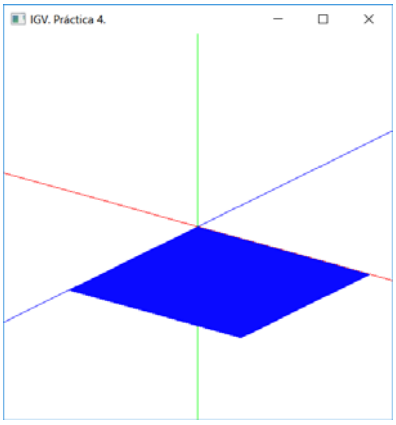
Para la realización de la práctica partimos del código:

- **pr4.cpp**: función main() del programa.
- **igvInterfaz.h** e **igvInterfaz.cpp**: especificación e implementación de la clase ***igvInterfaz***, que contiene la funcionalidad básica para crear una ventana de visualización, su configuración y la gestión de los eventos del sistema.
- **igvEscena3D.h e igvEscena3D.cpp**: especificación e implementación de la clase ***igvEscena3D***, que contiene la funcionalidad básica para visualizar una escena.
- **igvPunto3D.h e igvPunto3D.cpp**: especificación e implementación de la clase ***igvPunto3D***, que contiene la funcionalidad para declarar y utilizar objetos de tipo punto y vector.
- **igvCamara.h e igvCamara.cpp**: especificación e implementación de la clase ***igvCamara***, que contiene la funcionalidad básica para crear y manipular cámaras de visión en la aplicación.
- **igvMallaTriangulos.h** e **igvMallaTriangulos.cpp**: especificación e implementación inicial de la clase ***igvMallaTriangulos***, con la funcionalidad básica para crear y visualizar una malla de triángulos.

- **igvColor.h e igvColor.cpp**: especificación e implementación de la clase ***igvColor***, que contiene la funcionalidad básica para definir y utilizar objetos de tipo color.
- **igvFuenteLuz.h e igvFuenteLuz.cpp**: especificación e implementación de la clase ***igvFuenteLuz***, que contiene la funcionalidad básica para definir y utilizar luces puntuales y tipo foco.
- **igvMaterial.h e igvMaterial.cpp**: especificación e implementación de la clase ***igvMaterial***, que contiene la funcionalidad básica para definir y aplicar las características de un material.

- **igvTextura.h e igvTextura.cpp**: especificación e implementación de la clase ***igvTextura***, que contiene la funcionalidad básica para definir y aplicar texturas. Esta clase utiliza los ficheros **bmp.h** y **bmp.cpp** que contienen la funcionalidad para leer de disco y cargar en memoria una imagen BMP.

Si ejecutamos el programa sin realizar ninguna modificación se abre una ventana como la siguiente:



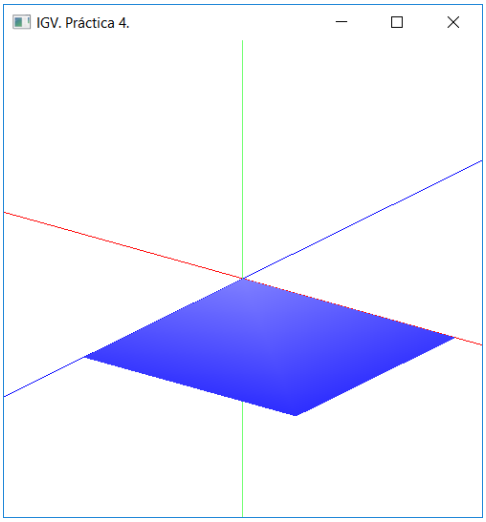
Lo que se muestra es la visualización de un quad de lado 5 con uno de sus vértices en el origen de coordenadas. Este quad se obtiene en la función `pintar_quad()` de igvEscena3D.cpp. La visualización es plana porque no hay definidas luces en la escena ni propiedades de material para el quad.

A) Implementa el método `igvFuenteLuz::aplicar()`, siguiendo las instrucciones entre comentarios, de tal manera que se defina la fuente de luz OpenGL en función de los atributos del objeto de la clase **`igvFuenteLuz`**.

B) Declara y aplica en el método `igvEscena3D::visualizar()` una luz de tipo puntual con los siguientes parámetros:

- identificador `GL_LIGHT0`
- posición: (1.0, 1.0, 1.0)
- color ambiental: (0.0, 0.0, 0.0, 1.0)
- color difuso: (1.0, 1.0, 1.0, 1.0)
- color especular: (1.0, 1.0, 1.0, 1.0)
- coeficientes de atenuación radial: 1.0, 0.0, 0.0

La visualización que se debe obtener es la siguiente:



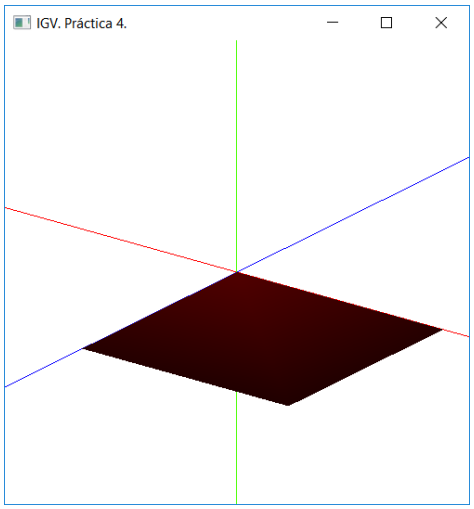
El color del objeto proviene del establecido para el eje Z y las propiedades del material son las definidas por OpenGL por defecto.

C) Implementa el método `igvMaterial::aplicar()`, siguiendo las instrucciones entre comentarios, de tal manera que se definan las características OpenGL del material en función de los atributos del objeto de la clase **`igvMaterial`**.

D) (**0.2 puntos**) Define y aplica al quad en el método `igvEscena3D::visualizar()` un material con los siguientes parámetros:

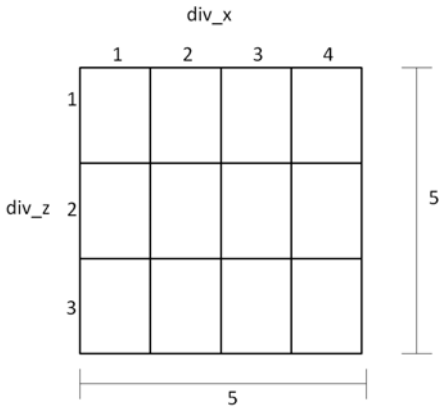
- coeficiente ambiental: (0.15, 0, 0)
- coeficiente difuso: (0.5, 0, 0)
- coeficiente especular: (0.5, 0.0, 0.0)
- exponente de Phong: 120

La visualización que se obtiene es la siguiente:

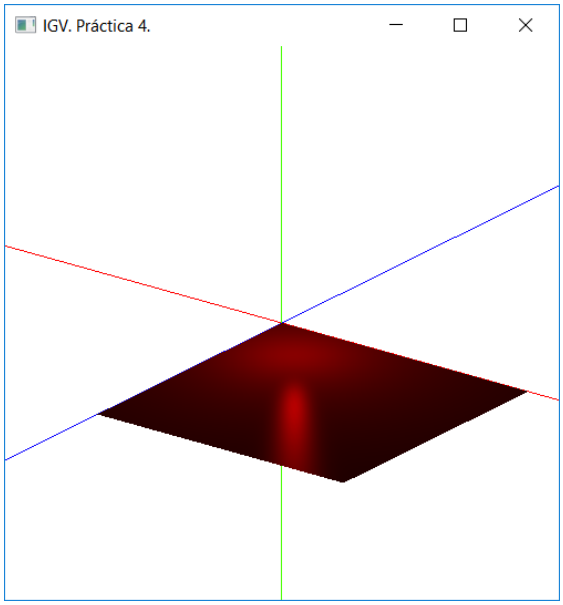


Esta visualización no es correcta porque al estar la luz situada en la posición (1, 1, 1) se debería ver tanto iluminación difusa como brillo especular sobre el quad. Esto ocurre porque la iluminación sólo se está calculando sobre los 4 vértices del quad y posteriormente la iluminación de los píxeles intermedios se obtiene por interpolación de esos 4 valores.

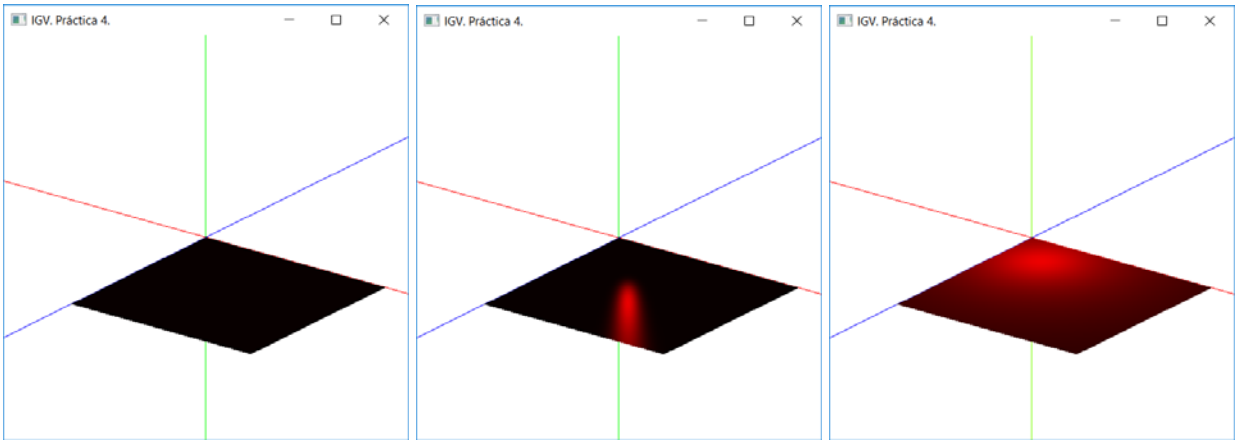
Para obtener una iluminación más realista es necesario convertir el quad original en una malla de quads, de manera que exista un mayor número de vértices donde se calcule la iluminación. Para ello, modifica adecuadamente la función `pintar_quad(div_x, div_z)` de manera que se genere una matriz de div_x x div_z quads cubriendo el original de lado 5 x 5. En la siguiente figura se muestra un ejemplo de malla de quads para div_x = 4 y div_z = 3:



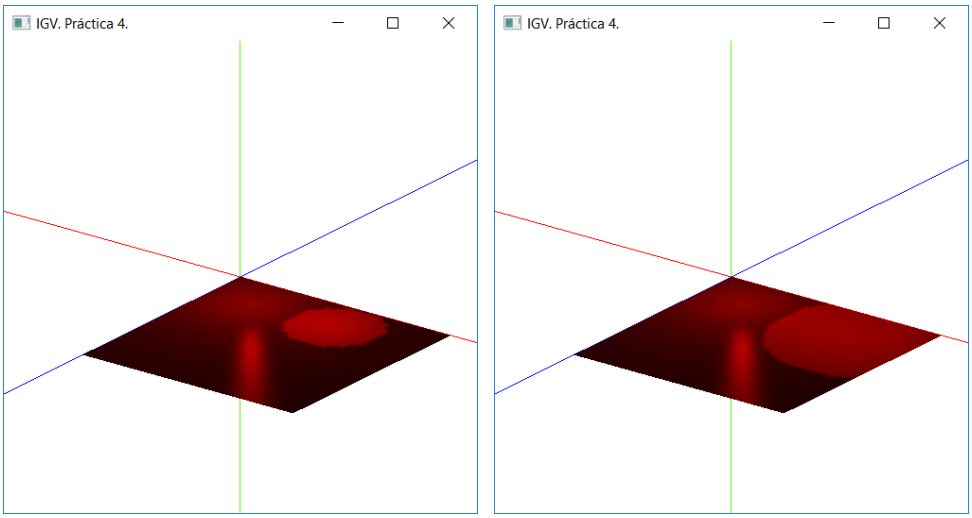
Sustituye la llamada `pintar_quad(1, 1);` que hay en `igvEscena3D::visualizar()` por `pintar_quads(50, 50);` debiendo obtener la siguiente visualización, donde ya sí se aprecia la iluminación difusa y el brillo especular sobre la malla de quads:



E) (**0.1 puntos**) Añade tres atributos a la clase `igvEscena` para almacenar la componente R del coeficiente difuso del material, la componente R del coeficiente especular del material y el exponente de Phong del material, respectivamente. Los valores iniciales de estos atributos deben corresponderse con los establecidos en el apartado anterior. Añade a la clase `igvInterfaz` el control por teclado de estos atributos, de manera que cuando se pulse 'd'/'D' se aumente/decremente en 0.1 el valor de la componente R del coeficiente difuso del material, cuando se pulse 's'/'S' se aumente/decremente en 0.1 el valor de la componente R del coeficiente especular del material y cuando se pulse 'p'/'P' se aumente/decremente en 10 el valor del exponente de Phong del material. Modifica convenientemente la aplicación del material implementado en el apartado anterior en el método `igvEscena::visualizar()` para que refleje las modificaciones de estos valores en las propiedades del material, de manera que utilizando el teclado se puedan obtener visualizaciones como las siguientes:

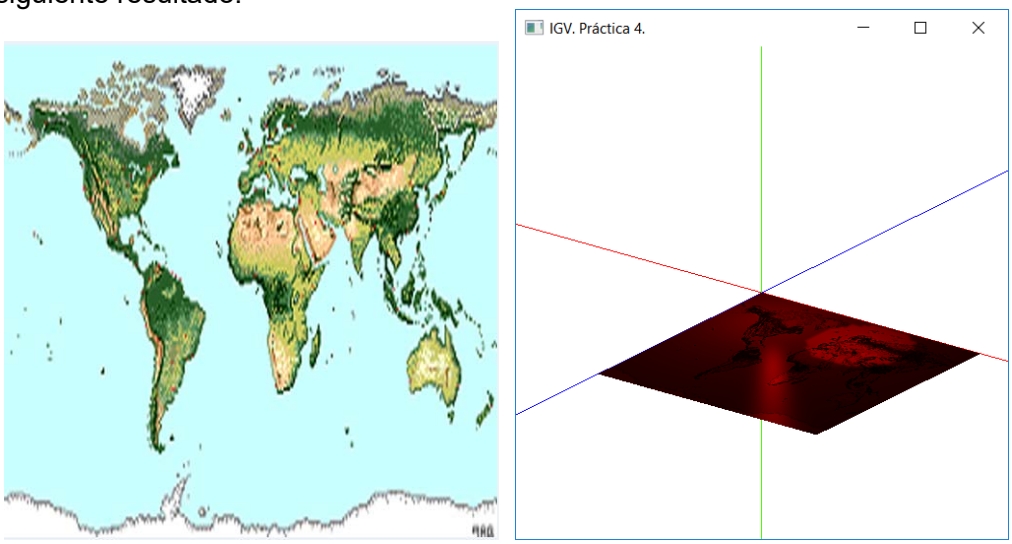


F) (**0.1 puntos**) Añade a la escena una fuente de luz tipo foco situada en la posición (3, 1, 1) de tal manera que se obtenga la figura mostrada a la izquierda:



Añade los atributos y el código necesario para mover el foco con las teclas del cursor de manera que el foco suba/baje en el eje Y en incrementos de 0.2 y se desplace a derecha/izquierda en el eje X en incrementos de 0.2. Lo que se debe mover es la posición del foco, pero la dirección del foco siempre apuntará hacia abajo. Por ejemplo, después de desplazar el foco 5 veces hacia arriba y 5 veces hacia la derecha, la visualización obtenida debe ser similar a la figura de la derecha.

G) (**0.2 puntos**) Añadir el código necesario para aplicar a la malla de quads la textura **mapa.bmp** y obtener el siguiente resultado:



Para ello:

- Modifica el código de la función `pintar_quads()` para calcular y asignar las coordenadas de textura correspondientes a cada vértice de la malla de quads.
- Añade el código necesario en el método constructor `igvTextura::igvTextura(char *fichero)` para cargar en OpenGL como textura la imagen almacenada en `fichero` en formato BMP. La imagen BMP está almacenada en **formato RGB** no RGBA.
- Añade el código necesario en el método `igvEscena3D::visualizar()`, justo antes de la visualización de la malla de quads, para crear y aplicar un objeto textura creado a partir del fichero MAPA.BMP.