1 (Practice 2) Lighting

Download the materials.zip associated to this practice and extract vector4.c and vector4.h to simplify the use of lights and materials data.

To do this practice you'll have to read the chapter about lighting models or at list know the Blinn-Phong model which is used in OpenGL.

In order to display lit objects, we have to define 3 components:

- ambient
- diffuse
- specular

for each light source and then for each object material.

1) Give the equation of the Blinn-Phong model and describe each parameters. Show which part of the equation is related to the ambient/diffuse/specular component.

1.1 Your first light

Create a function void init_light() and call it after the setup of GLUT's callbacks. You will add everything that's related to lighting initialization here.

To compute lighting OpenGL needs the normal at each vertex of the polygon mesh, these normals are not necessarily normalized (length one). Add glEnable(GL_NORMALIZE) in 'init_light()' to automatically normalize the normal vectors, to obtain correct lighting values. Enabling automatic normalization is not very efficient as OpenGL will normalize each time a frame is drawn. The ideal case is to have it disabled and ensure to give OpenGL normals of correct length.

Add glEnable(GL_LIGHTING) in the key function (use the 'e' key to activate/deactivate lighting).
1) What do you see when enabling lighting? Why?

Several lights are available in OpenGL: LIGHT0, LIGHT1, LIGHT2. Find out the maximum number of lights available and print that number into the command line with 'printf()'. See the documentation of 'glGet()' to do so.

With the function glEnable(GL_LIGHT0) and glDisable(GL_LIGHT0), switch on and off the LIGHT_0 (use the '0' key)
2) What do you see? Why?

In 'init_light()' Modify the ambient, diffuse, specular color of LIGHT0. To do so use:

```
Vec4 light0Ambient, light0Diffuse, light0Specular;
/* initialize them with vec4_set(...) */
glLightfv(GL_LIGHT0, GL_AMBIENT, light0Ambient);
```

```
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light0Diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light0Specular);
```

Try to change. the light location with glLightfv(GL_LIGHT0, GL_POSITION, light0_position) Put the LIGHT0 in a fixed location (e.g. to the left of the teapot's handle). Caution:

Lights positions are defined in object space. This means the light coordinates 'light0_position' will be multiplied by the current modelview matrix, the result is the final position of the light. Therefore the call to 'glLightfv(GL_LIGHT0, GL_POSITION, light0_position)' must be done in the display loop after 'gluLookAt()'. This way the position will be transformed by the modelview matrix just after the camera transformation, as a result 'light0_position' will be in world coordinates.

3) For LIGHT0, what type of light is created if the homogeneous coordinate (fourth component) is equal to zero?
4) And equal to one?

Activate a point light with LIGHT1 and place it in front of the teapot. Bound the key one '1' to activate/deactivate the light1.

1.2 Materials

First specify a white lighting:

```
    Vec4 light0Ambient  = {0.2f , 0.2f , 0.2f , 0.0f};
    Vec4 light0Diffuse  = {1.f  , 1.f  , 1.f  , 0.0f};
    Vec4 light0Specular = {1.f  , 1.f  , 1.   , 0.0f};
```

1) Do the glColor3f(...) function calls have any effect on the results?
2) What are the default values for the material components (ambient, diffuse, specular, shininess)?

Create the files material.c and material.h and implement the following header:

```
#ifndef MATERIAL_HPP__
#define MATERIAL_HPP__

typedef struct {
    Vec4 ambient;
    Vec4 diffuse;
    Vec4 specular;
    float shininess;
} Materials;

void materials_set_opengl_state(Materials m);

#endif // MATERIAL_HPP__
```

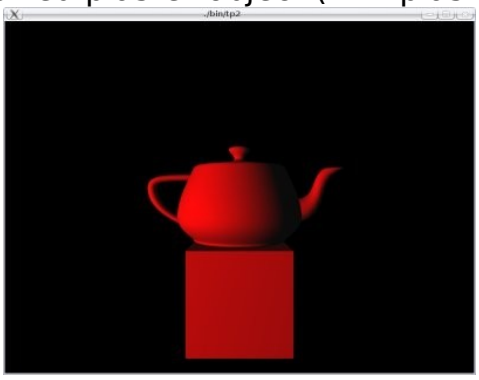
Don't forget to add the '.c' file into your makefile. To set the current OpenGl material state you can use the following function:

```
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT ,  object_ambient  );
```

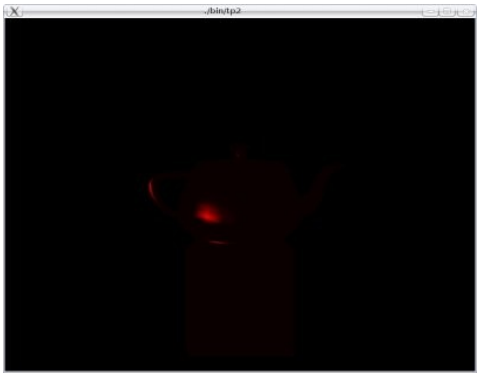
```
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE ,  object_diffuse  );
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,  object_specular );
    glMaterialf (GL_FRONT_AND_BACK, GL_SHININESS, object_shininess);
```

3) Recall what the 'fv' suffix stands for in 'glMaterialfv()'
4) GL_FRONT_AND_BACK indicates materials are to be applied for the front and back of the polygon. How does OpenGL distinguish front from back faces?
5) This criteria can be reversed with glFrontFace(GLenum mode); what is the default value?

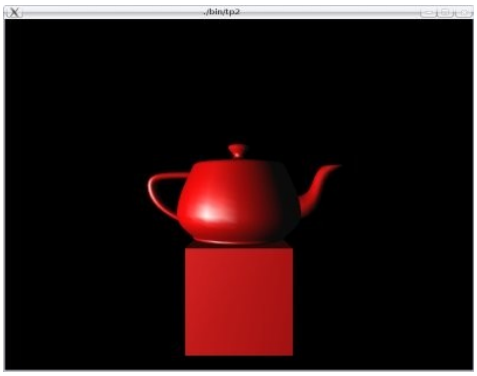
6) Give the correct values for the components (ambient, diffuse, specular and shininess) to obtain a red plaster object (hint plaster only diffuse light):



7) Give the values for a red mirror object (hint mirrors never diffuse light):

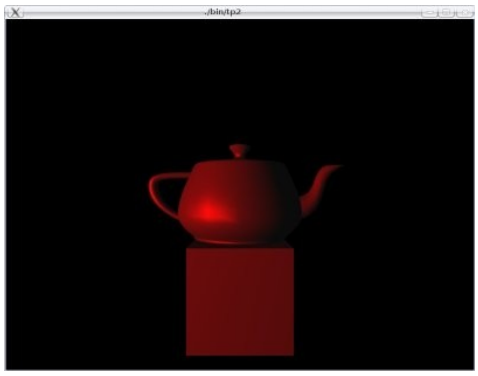


8) Give the values for a red "PVC" plastic object (hint: an insulating material has the same color than the light color for its specular component):



9) Give the values for a metallic conductor material (Hint: a conductor material has

the same values for its specular and diffuse components; same color but different intensities).

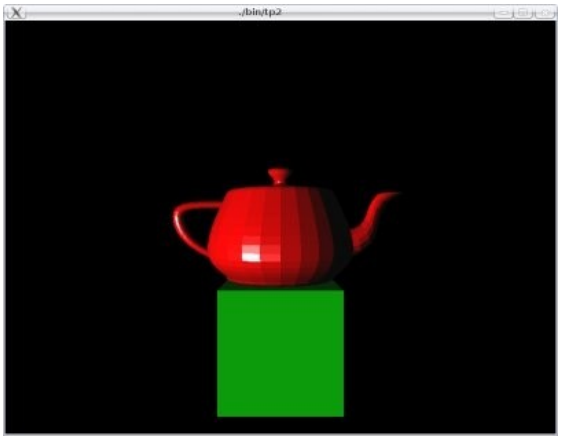


All these images was produced with the Gouraud shading model.

10) Recall the difference between a shading model and a lighting model.
11) Briefly describe how Gouraud shading is computed.

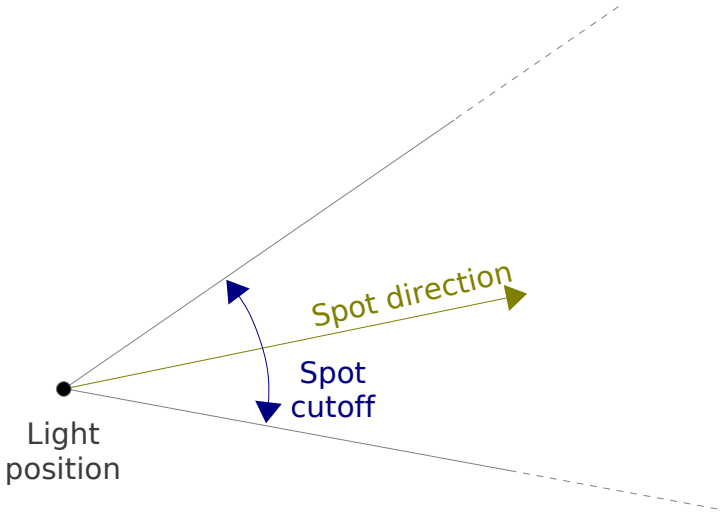
When modeling objects in software such as Blender/3D studio max artists often use the flat shading. This shading helps to see the topology of an object and its defects. Activate/deactivate the Gouraud shading, with the 'g' key. Use the OpenGL command:

glShadeModel(GL_SMOOTH/* or GL_FLAT */);



1.3 Add a frontal spot light

The goal here is to define a spot light bound to the camera origin and pointing ahead of us. The scheme below summarize the parameters needed to define a spot light:



First define another point light (GL_LIGHT2) and add a key shortcut to enable/disable it (use '2'). Try to put it above the objects.

In order to make GL_LIGHT2 a spot light we need to define the direction of the spot, it's aperture (the cutoff) and the light position.

Set the spot's parameters with:

```
glLightf(GL_LIGHT2, GL_SPOT_CUTOFF, 15.f);
glLightfv(GL_LIGHT2, GL_SPOT_DIRECTION, dir_light);
```

Note that the direction of the spot is also multiplied by the modelview matrix. Thus setting the direction must be done inside the display loop.

Because we want the spot to be attached to the camera position you must set the position and direction before the call to 'gluLookAt()'. This way the spot position/direction will be defined in camera space.

For the position choose (0., 0., 0., 1.) and the direction (0., 0., -1., 1.).

More parameters are available to tweak the spot characteristics use:

```
glLightf(GL_LIGHT2, GL_SPOT_EXPONENT, 100.f);
```

which set how much the intensity decrease as a point go away from the center of the spot. (look what happens for values 15. 50. and 100.)

There is more parameters if you're curious just look at the documentation of 'glLight()'.