Tutorial 14 :
Light & Material [Part I]

Tutorial | Download Section

Introduction

In the previous tutorials, we have the different kind of light in OpenGl and how to set up.

In the real world, when an object is illuminated, its reaction depends on its material. For example, a metal reacts differently than a piece of wood or a glass. This reaction can be described in OpenGL with material properties.
I've seen an example that display well the effects of material properties. It was created by **Silicon Graphics** and ported to Java by **Ron Cemer**.

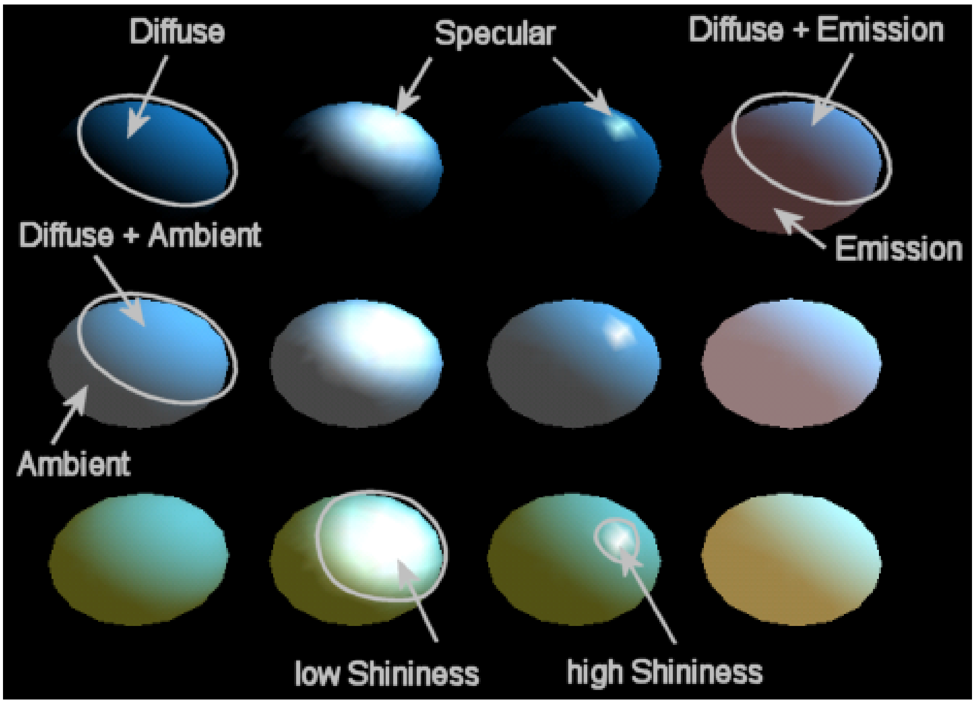
Light/Material models are described in Lesson 6 : Material.

Tutorial

Material

Light & Material are two notions that presents a lot of similarities. They have common properties like ambient, diffuse and specular colors.

The light used here have no ambient color, and a full white diffuse and specular color. Material have variables properties, the result is hown in this picture :



Material properties effects

Specular and shininess (specular exponenet) are two properties in relation with the highlight. Specular is color of the highlight, shininess is the specular exponent. A low shininess values will result to a spread highlight, a high shininess will result to a concentrated highlight.

Emission simulate material light emission, the material don't emit any light rays towards other objects. To create a material that emit light, like bulb, you need to add a light source at its location.

Here is the code that draw these spheres :

Object & Material properties

```
    float[] no_mat = {0.0f, 0.0f, 0.0f, 1.0f};
    float[] mat_ambient = {0.7f, 0.7f, 0.7f, 1.0f};
    float[] mat_ambient_color = {0.8f, 0.8f, 0.2f, 1.0f};
    float[] mat_diffuse = {0.1f, 0.5f, 0.8f, 1.0f};
    float[] mat_specular = {1.0f, 1.0f, 1.0f, 1.0f};
    float no_shininess = 0.0f;
    float low_shininess = 5.0f;
    float high_shininess = 100.0f;
    float[] mat_emission = {0.3f, 0.2f, 0.2f, 0.0f};

    /* draw sphere in first row, first column
     * diffuse reflection only; no ambient or specular
     */
    gl.glPushMatrix();
        gl.glTranslatef(-3.75f, 3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, no_mat);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, no_mat);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, no_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in first row, second column
     * diffuse and specular reflection; low shininess; no ambient
     */
    gl.glPushMatrix();
        gl.glTranslatef(-1.25f, 3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, no_mat);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mat_specular);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, low_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in first row, third column
     * diffuse and specular reflection; high shininess; no ambient
     */
    gl.glPushMatrix();
        gl.glTranslatef(1.25f, 3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, no_mat);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mat_specular);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, high_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in first row, fourth column
     * diffuse reflection; emission; no ambient or specular reflection
     */
    gl.glPushMatrix();
        gl.glTranslatef(3.75f, 3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, no_mat);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, no_mat);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, no_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, mat_emission);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in second row, first column
     * ambient and diffuse reflection; no specular
     */
    gl.glPushMatrix();
        gl.glTranslatef(-3.75f, 0.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, no_mat);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, no_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in second row, second column
     * ambient, diffuse and specular reflection; low shininess
     */
    gl.glPushMatrix();
        gl.glTranslatef(-1.25f, 0.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mat_specular);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, low_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in second row, third column
     * ambient, diffuse and specular reflection; high shininess
     */
    gl.glPushMatrix();
        gl.glTranslatef(1.25f, 0.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mat_specular);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, high_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in second row, fourth column
     * ambient and diffuse reflection; emission; no specular
     */
    gl.glPushMatrix();
        gl.glTranslatef(3.75f, 0.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, no_mat);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, no_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, mat_emission);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in third row, first column
     * colored ambient and diffuse reflection; no specular
     */
    gl.glPushMatrix();
        gl.glTranslatef(-3.75f, -3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient_color);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, no_mat);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, no_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();
    /* draw sphere in third row, second column
     * colored ambient, diffuse and specular reflection; low shininess
     */
    gl.glPushMatrix();
        gl.glTranslatef(-1.25f, -3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient_color);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mat_specular);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, low_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in third row, third column
     * colored ambient, diffuse and specular reflection; high shininess
     */
    gl.glPushMatrix();
        gl.glTranslatef(1.25f, -3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient_color);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mat_specular);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, high_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, no_mat);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();

    /* draw sphere in third row, fourth column
     * colored ambient and diffuse reflection; emission; no specular
     */
    gl.glPushMatrix();
        gl.glTranslatef(3.75f, -3.0f, 0.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, mat_ambient_color);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mat_diffuse);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, no_mat);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, no_shininess);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, mat_emission);
        glu.gluSphere(quadric, 1.0f, 16, 16);
    gl.glPopMatrix();
```

Light & Lighting Model

The light used is a classic light with no ambient component. The ambient component of the light is added with the Light Model.

Light Model

We can defines 3 properties for the lighting model :

- GL_LIGHT_MODEL_AMBIENT   set the global ambient light
- GL_LIGHT_MODEL_LOCAL_VIEWER use a local or infinite viewpoint, affect how specular highlight is calculated.
- GL_LIGHT_MODEL_TWO_SIDE   ont or two sided light. In case of two sided light, the material used for the back face is GL_BACK.

Look at Lesson 6 : Lighting Model to more details on this part.

The light

```
    //light properties
    float[] ambient = {0.0f, 0.0f, 0.0f, 1.0f};
    float[] diffuse = {1.0f, 1.0f, 1.0f, 1.0f};
    float[] specular = {1.0f, 1.0f, 1.0f, 1.0f};
    float[] position = {1.0f, 1.0f, 0.3f, 0.0f};
    gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, ambient);
    gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, diffuse);
    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position);

    //light model properties
    float[] model_ambient = {0.4f, 0.4f, 0.4f, 1.0f};
    int model_two_side = 1;                                //0=2sided, 1=1sided
    int viewpoint = 0;                                     //0=infiniteViewpoint,
1=localViewpoint

    /******************************
     * NEW * Global ambient light *
*********************************************************************************
     * We have seen in the previous tutorial that each lights are added between
them. *
*                                                                                *
     * We can add an ambient light for all the scene with GL_LIGHT_MODEL_AMBIENT.
The *
     * particularity of this ambient light is that it come from any
source.           *
     * In addition, this light is activated by GL_LIGHTING so you don't have
to       *
     * enable any GL_LIGHTi to use
it.                                                *
*********************************************************************************/
    gl.glLightModelfv(GL.GL_LIGHT_MODEL_AMBIENT, model_ambient);     //small white
ambient light

    /******************************************
     * NEW *   Local and infinite viewpoint   *
*********************************************************************************
     * The GL_LIGHT_MODEL_LOCAL_VIEWER propertie determine the calculation of
the    *
     * specular highlight (the
reflection).                                          *
     * The reflection depends on the direction of this two vectors
:                 *
     *   -> vector from a vertex to the
viewpoint                                    *
     *   -> vector from the vertex to the light
source                               *
     * Due to this reason, the reflection is dependant to the eye posiion. You
can   *
     * remarks that the reflection moves when the object is
deplaced.                *
*                                                                               *
     * Two different calculation
:                                                   *
     *   - with an infinite viewpoint : the direction between a vertex and
the       *
     * viewpoint is always the
same.                                                 *
     *   - with a local viewpoint : the directions don't remains the same.
Direction *
     * must be calculated so little slower. It is more realistic but
infinite        *
     * viewpoint is a good approximation in many cases (if object are
fixed)         *
*********************************************************************************/
    gl.glLightModeli(GL.GL_LIGHT_MODEL_LOCAL_VIEWER, viewpoint);

    /*********************************
     * NEW *   One/Two sided light   *
```

```
********************************************************************************
     * We can show if we would that light affects the two side of our shapes or
only *
     * one face (outside face defines by the normal
vector).                               *
     * We can define this with
GL_LIGHT_MODEL_TWO_SIDE.                              *

*********************************************************************************/
    //Only outside face because we don't see the inside of the spheres
    gl.glLightModeli(GL.GL_LIGHT_MODEL_TWO_SIDE, 1);

    gl.glEnable(GL.GL_LIGHT0);
    gl.glEnable(GL.GL_LIGHTING);
```

Download Section

Remember to download the GraphicEngine-1.1.2 to run this tutorial !

- Tutorial 14 src (7 ko) //Port to Jogl JSR-231 initially done by Magarrett Dias
- Tutorial 14 jar (8 ko)

If you've got any remarks on this tutorial, please let me know to improve it.
Thanks for your feedback.

Previous Tutorial Back Next turorial