OpenGL - Lab Session 3

Lights, materials, effects

During this lab session, we will work on lighting the scene, defining materials and implementing some effects with OpenGL.

Once again, this is based on your on-going program created during previous lab sessions.

1 Material components

In the first lab session, we used `glColor(...)` to define the material of objects. As we have seen in the lesson, the color defined by this command can be changed with `glColorMaterial(...)`.

For this lab session, let's define :

- the ambient and diffuse components with `glColor(...)`
- the specular component and shininess with `glMaterial(...)`.

Experiment the specular parameters by tuning the specular color and shininess. The shininess is a parameter that goes from 0 (large spot) to 128 (thin spot).

Move around the scene. What difference can you see between diffuse and specular components? Why?

2 Light components

Right now, the light being defined before the first transformation, it moves with the camera. To have a fixed light, we have to define the light position in the scene i.e. after the initial transformation `gluLookAt(...)`. To do this, let's create a function `setLight()` called at the end of `setCamera()` in which the light is defined instead of in `init(...)`. This way, the light is fixed.

We can go even further and manipulate the source light through keyboard. Inspiring yourselves from the camera placement, complete `specialKey(...)` and `setLight(...)` so that when F1/F2 are pressed, the light revolves around the scene.

To better understand the effects of the different light components, disable `GL_COLOR_MATERIAL`. Observe the result.

For now, the light only has a diffuse and ambient component. Add a specular component to the light.

Play with the different components, varying their intensities or the balance between red/green/blue. For instance, create a blue ambient component, a red diffuse component and a green specular component.

3 Spot lights

Enable `GL_COLOR_MATERIAL`.

A spot light is a light with a specified direction and half-angle. Add a `GL_SPOT_DIRECTION` and a `GL_SPOT_CUTOFF` to your light so that it acts as a spot light perpendicular to the plane with a half-angle of $30°$ and position $(0, 15, 0)$.

Depending on how your primitives are drawn, you may or may not see the influence of the spot light. Let's take the example of the plane.

The plane is defined by 4 vertices. OpenGL does a per-vertex lighting. It computes the local illumination on each vertex and interpolates these colors between vertices. In out case, the spot lights the center of the plane but none of the vertices of the plane. As a result, the illumination of the spot light is not detected. However, if we had vertices where the spot lights the plane, it could be detected and displayed. We see here that the 'resolution' of our primitives is important because of the per-vertex lighting. To better understand this phenomenon, let's draw our plane as a grid :

- define a parameter `gridSize` that gives how many quads are used for each row and each column of the grid with default value at 1;
- complete `commonKey(...)` so that `sizeGrid` is incremented (resp. decremented) when g (resp. f) is pressed ;
- Draw your plane as a grid of $sizeGrid * sizeGrid$ quads.

Now, as you increase the size of the grid, your plane is being refined and eventually the illuminated spot appears and gets more refined ... until it looks like a perfect circle.

Now that we can detect the illumination of the spot light, experiment the different parameters of the spotlight.

4 Effects

4.1 Transparency

Make some objects transparent or semi-transparent and play with the alpha value to see its influence.

4.2 Fog

Create a fog. Experiment the different parameters and don't forget to zoom in on your objects to better see the effect.