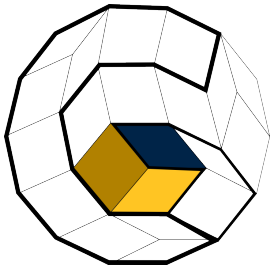
TU BRAUNSCHWEIG
Dipl. Inform. Martin Eisemann
Institut für Computergraphik
Lorenz Rogge (rogge@cg.tu-bs.de)



May 4, 2011

Echtzeit Computergraphik SS 2011
Assignment 3

This assigment is about the use of Lights and Materials as well as the feature of multiple viewports. Please present your solution to this exercise on Monday, May 9th, 2011.

3.1 Material and Light (20 Points)

Create a scene consisting of four spheres, each using a different material. Define four different materials used by the rendered objects, by setting up different parameter configurations for diffuse color, ambient color, specular color and shininess exponent. Try out different shininess exponents such as 10, 100, 1000 and so on. The larger the exponent get, the smaller the specular reflection spot will be. Use the `GLUT` method `glutSolidSphere(...)` to quickly render a sphere and use a different material for each sphere. Place the spheres at the following world space coordinates:

- $(-3.000, 0.000, 0.000)$
- $(3.000, 0.000, 0.000)$
- $(0.000, 0.000, -5.196)$
- $(0.000, 4.905, -1.732)$

Of course material alone is not sufficient. We need light sources! Setup two light sources - one emitting plain white light, the other emitting red light and define their ambient and specular parameters, too. Place the white light source as if it would be attached to the camera and the red light source fixed to the observed scene at $(10, 10, 10)$.
Now a rendering of the scene should look like this:

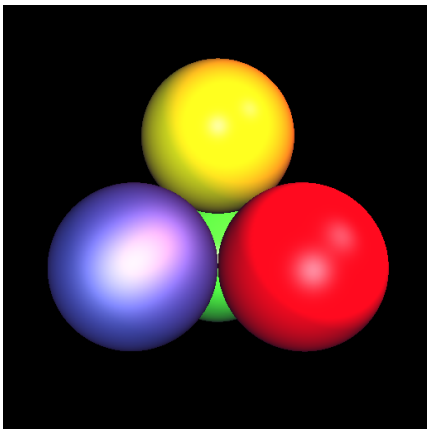


Figure 1: Example view of four spheres using different materials.

3.2 Using multiple Viewports (20 Points)

In this task you have to implement the rendering of two seperate Viewports. The OpenGL window should show two views of the same scene, where the first Viewport shows a rendering from a camera controlled by the user, called 1st-person camera. So this should work like the renderings used in previous exercises. Render the scene of task 3.1 and use a trackball to navigate around. Set the viewport parameters, so that only the left half of the window is rendered this time. The second Viewport then allows to observe the scene and the camera placement and orientation in a 3rd-person view. For the 3rd-person view implement the following:

- Set the viewport to the second half of the window.
- Use a second trackball to control the view within the second viewport. Be sure to forward user inputs to the correct trackball when handling events.
- Render the scene from task 3.1 at the world space origin.
- Place a camera model loaded from file *meshes/camera.obj* to represent the camera and place it like the camera is set up in the 1st person view. Do not extract the position and orientation of the camera from a trackball but make use of the possibility to save and reload matrices in OpenGL. You may use `glGetFloatv(...)` to extract the currently active matrix of a specific stack. Use `GL_PROJECTION_MATRIX` or `GL_MODELVIEV_MATRIX` to save the correct matrix. Save both matrix types when rendering the 1st-person view and use them in the 3rd-person rendering.
- To visualize the camera frustum of the 1st-person camera, transform the already given unit cube to the frustum shape using the saved projection matrix. Render the edges of this cube using `GL_LINES` or `GL_LINE_LOOP` after placing the frustum at the camera's position (use your saved modelview matrix again).

The program already allows to control the positions of the near and far clipping planes and the opening angle of the 1st-person camera. When changing these parameters using the keys `R/F`, `T/G` and `Z/H`, one is able to observe the result in the 3rd-person view directly. A second trackball allows to navigate in this 3rd-person view.
The result should look like Figure 2

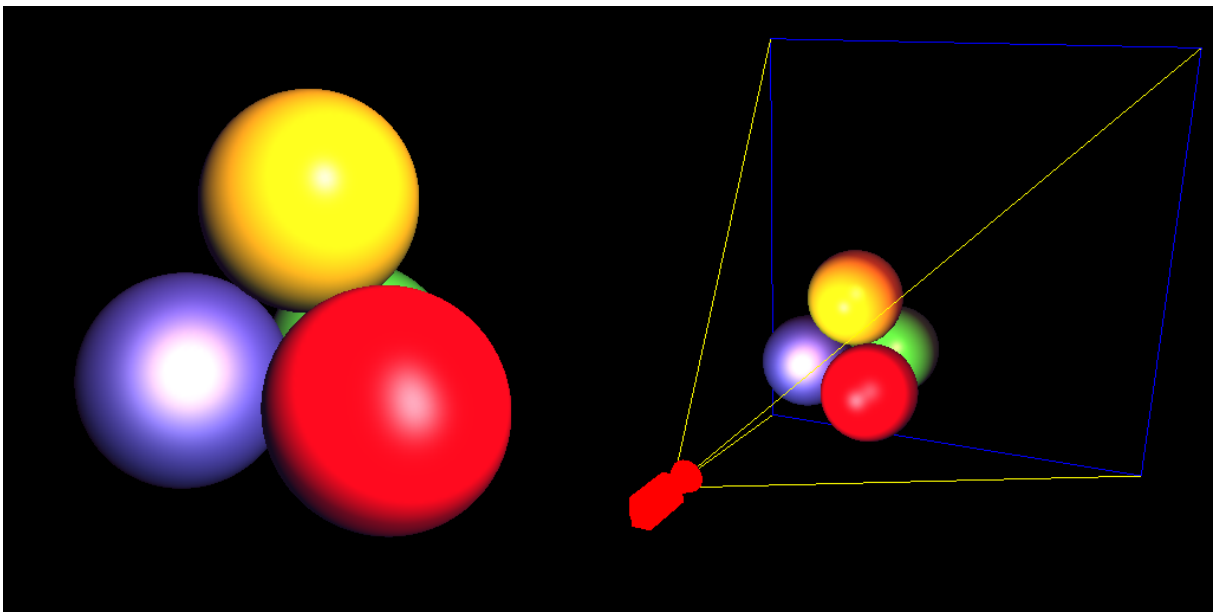


Figure 2: The left view is a 1st-person rendering of the scene, while the right view shows a 3rd-person visualization of scene and camera.