CG 2013-14
Assignment 1: Modeling

Out: Nov 12. Due: Nov 24

Introduction

In this first assignment, you will setup transformation hierarchies in order to model virtual objects composed by rigid primitives. Interactive rendering will be done through OpenGL directly into the provided template code.

You have to perform the assignment using the C language. To ease your development, we are providing a template code to perform basic vector calculations, interaction with the 3D scene through a trackball and the basic functions to initialize and update the rendering.

In order to compile the provide template code you will need `freeglut` (a free implementation of the glut library). Instructions to install `freeglut` on windows and MacOSX are available on the web page of the course.

To ease your debugging, we are providing the compiled version of the solution code, which can be used to compare your results with ours. Do not decompile the code.

For a specification of OpenGL, see
http://www.opengl.org/sdk/docs/man2/

For a specification of GLUT, see
http://www.opengl.org/documentation/specs/glut/spec3/spec3.html

Template code

The provided template code initializes an interactive window which draws the virtual scene 60 times per second, like already seen in the labs. Furthermore, the template code provides the following functionalities:

Trackball: by pressing the left button of the mouse and dragging, the scene rotates consistently with the motion of the mouse. By pressing the right mouse and dragging up and down, the scene zooms in and out, respectively.

Vector3: a simple data type providing the basic operations on a vector of three elements (sum, subtraction, multiplication by a scalar value, etc.), dot product, vector product, normalization and other helper functionalities.

glPrint: a helper function to print on the screen.

Requirements

1. **Hierarchical model of a lamp. [10 points]**
   Use the function DrawCylinder and gluSolidSphere(...) to model a simple lamp. The arms of the lamps can rotate using the keyboard. See executable "`assignment.1.exe`" for the expected outcome.
   - 'a' and 's' rotates the first arm.
   - 'd' and 'f' rotates the second arm.
   - 'g' and 'h' rotates the lamp.

2. **Implement the same lamp as the previous point with your custom implementation of transformation functions. [10 points]**
   By using the instruction `glMultMatrix(…)`, write your own implementation of the OpenGL translation, scaling and rotation. You will find an empty implementation of these methods, namely:
   - `mglScaling(scal)` must implement the uniform scaling
   - `mglRotateX(angle)` must implement the rotation around the x axis
   - `mglRotateY(angle)` must implement the rotation around the y axis
   - `mglRotateZ(angle)` must implement the rotation around the z axis
   - `mglRotate(angle, ux, uy, uz)` must implement the rotation around the axis (ux, uy, uz) *[this point is facultative]*

   Replace the standard OpenGL transformation functions with your own implementations, verifying their correctness.

3. **Light at the lamp position. [5 points]**
   Put a light near the lamp (near, not inside) and make it move together with the lamp. Place some random object into the scene to be lighted (e.g., a teapot, cubes, etc).

Submission

Please, send the code for each point ***within the 24rd of November*** to cg1314@gmail.com

Extra credits

Choose *some* of these items for extra credits.

- **Tuning rotation speed.**
  In point 1, multiply the rotation angle by a given speed (which is initially 1). By pressing the left and the right cursor, the speed shall respectively decrease and increase, reducing or increasing the rotation speed of the lamp arms.

- **Projection transformations.**
  In point 2, together with the listed transformations, implement the projection transformations corresponding to `gluPerspective` and `glOrtho` and use them in your code.