EECS 487 Winter 2010

Lab 4: Viewing and Lighting

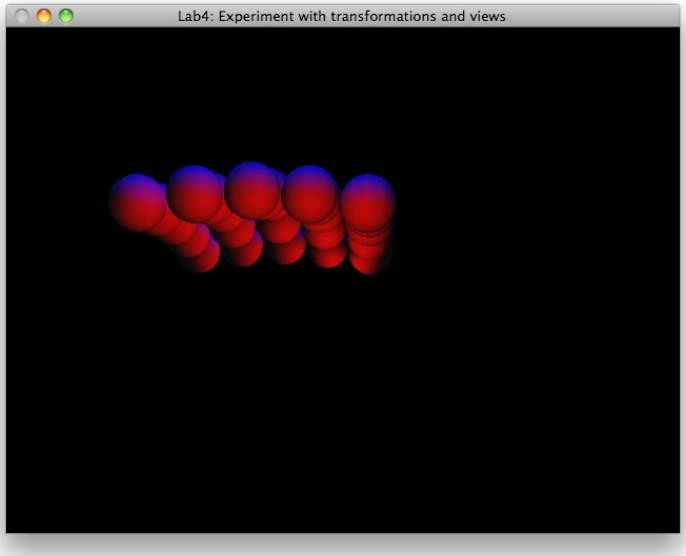

Fig 1: Screenshot showing one of the views of the objects in a 3D world

In this exercise you will implement the transformation hierarchy to simulate the effect of flying around a group of objects. You will be able to see how interactive motion can be implemented by proper manipulation of camera location and orientation and how the transformation pipeline, once implemented, is reused without modification (letting the computer do all the hard work). As with real airplanes, our imaginary airplane has the following degrees of freedom, shown in Fig 2:

1. *Translation* - forwards and backwards along the gaze direction (mapped to 'w' and to 's'),
2. *Translation* – left and right perpendicular to the gaze direction (mapped to 'a' and 'd'),
3. *Pitch* - turning up or down (mapped to mouse drag up/down), and
4. *Yaw* - turning left or right (mapped to mouse drag left/right).

Fig 2: Basic degrees of freedom for an airplane [http://en.wikipedia.org/wiki/Flight_dynamics]

Each of these types of motion can be simulated by manipulating the eye-position **e**, gaze-direction **g**, and the top-vector **t** (for instance, roll: rotate the top vector around the gaze vector). Each time you render the 3D world to screen you will have to transform and project the scene onto the 2D viewing plane (the *near*-plane in the viewing volume). This *near*-plane is your Open GL window. The origin for eye coordinates is the center of this plane. Use perspective projection;

otherwise translations will not be obvious. Starting in *fakeGL.h* implement the functions *_glVertex3f(...)* and *_glNormal3f(...)* by updating the passed-in coordinates by the ModelView Matrix and the Projection Matrix. Note that the behavior of each is not quite the same.

Then implement *_gluLookAt(...)* and *_glFrustum(...)* to post-multiply the necessary transformation on the *currentMatrix*. After this step the scene should come into view.

Then, *viewLights.cpp* implement the key bindings in *keyb(...)* to move the viewer. Next, in *motion*, update the top and gaze directions to implement pitch.

The above steps may seem quite extensive but are all straight-forward. Do not over think the tasks too much.

Finally, note a few lines at the top of the file:

```
GLfloat pos[4] = {0.0, 2.0, 0.0, 1.0};
glLightfv(GL_LIGHT0, GL_POSITION, pos);
```

Put these lines somewhere in *display()* so that the light has a fixed position with respect to the sphere's. That is, it should float above them at all times. You are welcome to draw a sphere at this location to ensure that it has been done correctly.

Then, note the other two lines:

```
 GLfloat pos2[4] = {0.0, 0.0, 0.0, 1.0};
glLightfv(GL_LIGHT1, GL_POSITION, pos2);
```

Put these lines somewhere in *display()* so that the light has a fixed position with respect to the viewer. You are welcome to draw a sphere at this location to ensure that it has been done correctly.

Note that the light “attached” to the viewer has attenuation, so its effect is not visible until the viewer gets close to the sphere's.

Note that for this lab it may be helpful to look at the man pages for the OpenGL functions being implemented.