Lab 4 - Lighting and Camera
Due (via Bitbucket and hard copy) Tuesday, 6 October 2015
50 points

Lab Goals

- Develop a three-dimensional scene with a moving camera and with a variety of lighting schemes
- Practice changing lighting schemes and controlling the camera

Assignment Details

For this lab, you have two options to chose from: summarized most easily as directed and undirected development. For the undirected development, you will create a three-dimensional scene under roughly the same guidelines as Lab 3. For the directed development, I provide some direction for a 3D scene you can create.

Scene Requirements

Regardless of your scene that you implement, you must implement at least the following:

1. A moving camera in the XZ-plane (should be a lot of copy/paste code from the example in Monday’s class), with updates based on arrow key and mouse input.
2. Three different lighting schemes, controlled by keypress inputs 'q', 'w', and 'e'.
3. Objects moving in the scene (a non-static scene).
4. One or more objects defined in a class.

Project Inspiration

You are welcome to let creativity be your guide, but if you want a project idea, here is one suggestion:

The Processing 3D library only includes the primitive shapes box and sphere, but you can design your own through use of the beginShape() and endShape() functions, much as we did in 2D. For example, you can redefine the box primitive by drawing six faces (draw them out for yourself if you need to prove to yourself that this works):
In a similar vein, you can define a tetrahedron shape by combining together four equilateral triangles, one for each face of the solid. A tetrahedron class could take as parameters the (x,y,z) center of the solid, plus a length parameter for the length of an edge on the faces. You can calculate
the positions of each of the vertices from there using a bit of geometry.

From there, you can reimplement our class physics system in 3D using these tetrahedra. Simply start with the Ball class, add a new sz variable for position and vz for velocity, and update the draw method to draw a tetrahedron instead of an ellipse. To make things more interesting, you can impart rotation in the X, Y, and Z directions by using your matrix multiply function from Lab 2 to compute the new positions of vertices following rotation. The tetrahedra collide with the walls and floor of the 3D space when one or more of the vertices intersect the planes that define the walls and floor.

Or you could be creative and implement your own scene. Your call.

Submission Details

For this assignment, please submit the following items which you have followed while completing this lab in paper form. Also, please upload these same items to your cs382f2015-<your user name> repository. Your submission should include the following:

1. Print: A representative segment of commented source code for the scene that you implemented
2. Upload: All of your commented source code for the scene that you implemented
3. Print and Upload: An Assignment Information Sheet

Additional Group Requirements

You are welcome to submit this assignment on your own, or in a group with one other person. In cases of groups, I would prefer one experienced programmer and one inexperienced programmer in the group, but this is not an absolute requirement. Additionally, each group need only submit one paper copy of their work, but each member of the group should push all items to their own repositories.

Finally, each group must create a short document (0.5-1 pages) detailing the work breakdown of the group members: who worked on which components, which lines of code belong to each group member, etc. Comments in the code will help here too.