Introduction to Computer Graphics

Assignment Five

“Let there be light”

U-grad due date: November 18, 11:59 PM
Grad Student due date: November 4, 11:59 PM
(U-grads doing project should shoot for Grad student due date)

Credit (Ugrad) 9 points (Relative, and roughly absolute weighting)
Credit (Ugrads doing projects): 6 points (Relative, and roughly absolute weighting)
Credit (Grad): 5 points (Relative, and roughly absolute weighting)

This assignment may be done in pairs if you prefer. In fact, this is recommended. However, to give those who are working alone a break, the part of the assignment which is optional for those doing projects is also optional for individual submitters.

**IMPORTANT**: All parts of all assignments should be understood for the exams, regardless of whether your partner did it, or whether that part was “optional” for you (“optional” only applies to implementation).

In this assignment we will improve “pp-world” even more. The same rules regarding input and output and program exit from the previous assignments apply.

### Non-project people working in pairs

Add a sphere command. Parameters are x,y,z, and radius. Implement an approximation of a sphere as a collection of polygons. You can use any collection of polygons that resembles a sphere. Try to use a simple strategy! The main requirement is that a sphere must be clearly distinguishable from a stretched/rotated box. You can give your spheres default colors.

Scaling of a sphere with x,X,y,Y,z,Z should scale the sphere uniformly (don’t create an ellipsoid, unless you want to do so for extra credit, in which case, go for it, but let us know that you are doing so in the README. Of course, a complete job would require an “ellipsoid” command as well, but this would be a secondary extra).

Rotating a sphere can be implemented as a NOP, but translation should work. The user should be able to add a default sphere with the menu. Picking should work on spheres.

**End part for non-project people working in pairs**
In the following, you can use the naïve version of color introduced in class (diagonal model). Also, R,G, or B values that are greater than 255 should be set to 255. OpenGL probably does this for you. (This is called clipping—yes, another use of the word).

Implement the following command to add ambient light:

```
ambient [ r ] [ g ] [ b ]
```

(r,g,b) is the color of a perfect white diffuse reflector. Thus even if there are no lights, your world is not black.

Implement a command to add a single point source. The command should be:

```
light [ x ] [ y ] [ z ] [ r ] [ g ] [ b ]
```

You can assume that it is a point source. Compute its direction relative to a polygon based on the center of polygon. It would be best to use the center of mass of the polygon (why?), but the average of the vertices is acceptable. If you want to allow more than one point source, that is fine (note this in the README).

Add a command:

```
specular [ v ] [ n ]
```

v is an integer from 0 to 100 which expresses a percentage of the light to be added via the Phong model, and n is the exponent in the Phong model. The command applies to every surface of a preceding box, or the preceding sphere. Apply the specularity to the entire surface, based on the center of the surface. Arguably it would be best to use the center of mass of the polygon, but the average of the vertices is acceptable.

Add a command:

```
mirror [ v ]
```

<v> is an integer from 0 to 100 which expresses a percentage of the light to be reflected. The command applies to every surface of a preceding box, or the preceding sphere. Add a bunch of white to the surface (proportional to <v>) to indicate to the user that this surface is a mirror. IMPORTANT: The mirror will not act like a mirror until assignment 6.

You should add the ability to make things specular and have mirror reflections using picking and user input. If you like, you can just have a few simple mirror and specular options such as not-a-mirror, poor-mirror, and good-mirror, but you should provide at least two levels of specularity strength, and two levels of specular sharpness (controlled through the exponent). If you choose to provide minimal capability, you need to make sure that the choices given provide noticeable and varied effects. Since I am purposely leaving this part of the interface up to your imagination, be sure to put the user incantations in the README file.

**Extra credit**

If you would like to improve on the program, be sure to explain what you did in the README file, and it will be considered for extra credit.
**Deliverables**

You must electronically submit a README containing any relevant information, but at a minimum, your name; an executable (called a5); and a src directory containing source files and a Makefile which can be used to build the executable.

The program must compile and run on one of the graphics machines (gr01, ..., gr10). Put in the README file the machine which you have verified this on.

Note that the graphics machines can be booted into Windows by people in the lab, so that it is possible that if you are working remotely that you will need to try more than one. We encourage students to use the higher numbered machines for Windows (7 through 10), but this cannot be enforced.

The turnin name is cs433_hw5.