**Option 2.** [200 points] You are to write a ray tracing program.  Your program should generate a fixed, raytraced view of a scene.  You are to write the raytracer yourself, from scratch.  That means that you are to write your own ray routines, not use external libraries.

Your program should have the following characteristics:

* You should allow and demonstrate the display of infinite planes and of spheres
* You should perform a local lighting calculation, including ambient, diffuse, and specular (Phong) terms
* You should include both shadows and reflections (refraction is not needed)
* You should have at least four objects and at least 2 lights (preferably more of each) in the scene.  The lights should vary in color and position.  The objects should have different coefficients of ambient, diffuse, and specular reflection, and should also have varying levels of reflectance.
* Your image should demonstrate all of the required properties.  That is, from looking at the image, a person should be able to tell that the objects are made of differing materials, that the lights are different, that the local lighting calculation is being performed, that shadows are working correctly (note that you want “complex” shadows, where one object causes a shadow on another object), etc. You may use multiple images to demonstrate all the characteristics, but should provide some way for users to switch between the different images.  Include a README file stating how each of the required characteristics is demonstrated in the scene, and give any commands the user needs to specify to change scenes (if that option is chosen)

Be sure that your code is well-written/documented enough that it can easily be verified that you are performing lighting, ray casting, etc. in the right manner.

Skeleton code will be provided on the web (similar to assignment 2) that will allow you to set individual pixel colors and display the resulting image (using OpenGL).  However, you are to set your own scene up however you would like it.  You will need to choose an eye position, world-space screen position, and positions for the objects themselves.

As a side note, if this program is well-written, it will be easy to extend it to handle more interesting scenes.  This can include things like: triangles/polygons, cylinders, and other shapes; refraction; texture mapping;

[10] Scene set up – eyepoint and screen in world, and rays generated

[20] Planes drawn

[20] Spheres drawn

[10] Ambient lighting correct

[20] Diffuse lighting correct

[30] Specular lighting correct

[35] Shadows included and correct

[35] Reflection included and correct

[20] Numbers of Objects, lights, etc. meet requirements