

## CS334: Assignment #2 – Animated Shading and Lighting

**Out:** Sept 18, 2008

**Back/Due:** Oct 2, 2008

### Objective:

This focus of this assignment is to obtain a good understanding of the shading and light component of the graphics pipeline. You can recycle as much as you like from previous assignments; however, OpenGL has all the necessary matrix operations you will need. Use of OpenGL matrix operations is not mandatory but if your matrix/point/vector library does not work, you will get points deducted. You will write a program that creates a basic scene model and for each frame issues OpenGL commands to control the lights and the light model, animate the objects, and move the viewpoint.

### Summary:

The assignment is to implement a program which renders and shades a 3D scene in a parameterized way. Once the scene is created, the user can control the viewpoint and navigate through and around the scene and can control the lighting/shading parameters. All control is via a GLUT-based user interface. Changes to any of the parameters should be effective in the next drawn frame.

### Specifics:

- (0) Scene Creation: To create objects for the scene, you may generate your own objects using polygons, triangles, and so forth but your scene must at least include the ability to render a sphere, cube, cone, and teapot using the GLUT functions `glutSolidSphere`, `glutSolidCube`, `glutSolidCone`, and `glutSolidTeapot`. These methods will generate the geometry and object for you. You must be able to individually control the 3D translation and 3D rotation of these objects as well so that the user can position them in the scene at will using the GUI. Thus to render each object you will make calls to `glTranslate` and `glRotate` and then call `glutSolidSphere`, for instance.
- (1) Shading and Lighting: The program should permit changing the global shading and lighting parameters. In particular, you must support two overall shading modes: (i) no shading and lighting – just use a solid color per object: sphere=red, cube=light blue, cone=yellow, teapot=light green, and (ii) enabling shading and lighting under the controls described next: (a) selection of using all point lights or all directional lights, (b) the number of OpenGL lights (up to three), (c) the location of each light, (d) the direction of each light, (e) the ambient color of each light, (f) the diffuse color of each light, and (g) the specular color of each light. The location/direction of the lights should be controlled in a manner similar to the viewpoint (e.g., using `glTranslate` and `glRotate`). Further, the program should allow specifying for the entire scene (h) a global ambient and diffuse reflection coefficient, (i) a global specular reflection coefficient, and (j) a global specular exponent (e.g., shininess). By altering these parameters, a model can be given a general satin look, glossy look, metal look, plastic look, etc.

- (2) Rendering: The scene should be rendered and shaded every frame using the aforementioned parameters and objects. The viewpoint should also be controllable using the trackball you implemented in the previous assignment.
- (3) User Interface: A GLUT-based user interface must be used to allow the user to specify all the parameters. Reasonable initial values should be given to all parameters at program initialization. The default viewing position should be (0,0,0) and default viewing direction should be (0,0,-1). A reasonable field of view is 60 degrees. A good near and far distance should be computed based on the world-space size of the model. The GUI should also include parameters to control all the previously described shading and lighting options. Below is a suggested list of GUI controls. You can develop your own GUI style but it should have at least the following functionality (if your GUI is not understandable or functional, you will appropriately be deducted points):
- a. 3D translation for any object being displayed in the scene (suggestion: use GLUT drop down box to select current object, then have a single 3D translation GUI), you only need to allow the user to translate the object along the x-axis and z-axis since the y position will be controlled by the “Animating Objects” section.
  - b. 3D rotation for any object being displayed in the scene (suggestion: use GLUT drop down box to select current object, then have a single 3D rotation GUI)
  - c. Check boxes to select which of the three lights are active
  - d. Select global point lights or global directional lights (hint: a point light has a position and a directional light only has a direction)
  - e. 3D translation for each light (suggestion: use GLUT radio button to select current light, e.g., 1 of 3, then have a single 3D translation GUI)
  - f. 3D rotation for each light (suggestion: use GLUT radio button to select current light, e.g., 1 of 3, then have a single 3D rotation GUI), you may assume the rotation for each light is about (0,0,0)
  - g. Ambient color of each light
  - h. Diffuse color of each light
  - i. Specular color of each light
  - j. Ambient and diffuse coefficient of all lights
  - k. Specular coefficient of all lights
  - l. Specular exponent of all lights
  - m. Virtual trackball/mouse controls that allow the rotation, translation, and zoom of the entire scene.
- (4) Animating Objects: The objects can be given a velocity component that will make them bounce up and down in the scene. For this assignment, the objects will only move along the y-axis. When the object reaches a user defined ceiling or wall, the object’s velocity will reverse which will make it seem like it has collided with the boundary and bounced the other direction. To update the position of an object use Euler’s equation  $y = y_0 + vt$ , where  $y$  is the next position,  $y_0$  is the current position,  $v$  is the object’s velocity along the y-axis, and  $t$  is the time step.

**Extra Credit (0 to 20%):** Implement additional shading and lighting, e.g. attenuation by distance, atmospheric effects, shadows, etc; implement additional objects; new (and intuitive to use) GUI parameters should be added for these effects; please also include a short description of what you added, how it is implemented, and how to use it.

**Grading:**

We will use the interface to alter the aforementioned parameters and expect to see the correct behavior. *If the interface does not allow a certain parameter to be changed, it will be considered not implemented.* If the program does not compile, zero points will be given.

**This program is a significant step from the previous assignments and thus please start early on the assignment. If you have questions, please see the instructor or TA.**