This assignment involves familiarization with illumination and shading. The assignment web-page is at www.cs.umd.edu/class/fall2004/cmsc427/assg4/.

(a) Set up a directional light source to represent the Sun at any point in the sky. The Sun should illuminate all geometry except for the sky box. Set up the material parameters for the Humvee so that it has both specular and diffuse colors. The trees and the terrain only need diffuse colors. Have OpenGL do specular and diffuse lighting on the Humvee using the per-vertex normals given with the dataset. Compute the per-vertex normals for the terrain by using the $64 \times 64$ terrain height map. For each pixel $(x, z)$, let the change in the height be $\Delta h$ over $\Delta x$ distance and $\Delta g$ over $\Delta z$ distance. Then, the normal can be approximated by the cross product of $(\Delta x, \Delta h, 0)$ and $(0, \Delta g, \Delta z)$. Have OpenGL normalize all the normals you give it. Have the lighting and the textures blend using OpenGL’s GL_MODULATE function. You can get better-looking specular highlights on textured Humvee by using GL_SEPARATE_SPECULAR_COLOR. **Point distribution:** 1 point for any kind of lighting, 1 point for per-vertex lighting, 2 points for normal-based per-vertex lighting, 1 point for blending textures with lighting. (5 points)

(b) Next have the Sun continuously rotate about the origin of the sky box. Set up the speed of rotation to be slow but perceptible (say, once every 60 seconds). Have the intensity of the Sun and the brightness of the sky box increase the closer it is to being noon. **Point distribution:** 2 points (2 points)

(c) Implement headlights for the Humvee using spotlights. Allow the user to toggle the headlights on and off using the <h> key. The headlights should illuminate the terrain and the trees, but not the sky box. Assume that the headlights do not cast any shadows. **Point distribution:** 3 points (3 points)

(d) Implement Humvee and tree shadows on the terrain as follows. Use a shadow-projection matrix to cast the shadows on the x-z plane. Remember that OpenGL stores its matrices in the column major order ([col][row]). Render the shadow triangles on a $3 \times 64 \times 3 \times 64$ region of the back buffer. Pressing the <s> key should toggle the display of the shadow back buffer. For every pixel of the $64 \times 64$ terrain compute the weighted average of the corresponding $3 \times 3$ region in the back buffer covered by a shadow triangle. Use this weighted average value to darken the corresponding vertex of the terrain accordingly. Simplify the casting of shadows by ignoring the effects of the terrain height. During shadow determination, simply assume that the terrain height and the geometry is at $y = 0$, determine the terrain vertices covered by the shadow, and then darken them at the appropriate height during final rendering. **Point distribution:** 3 points for displaying shadow back buffer; 2 points for displaying shadows. (5 points)

(e) Randomly distribute 20 or so spheres (you can use gluSphere here) over the terrain. Initially make the spheres red and large. Every time the Humvee hits a red sphere, the sphere should turn blue and shrink. Show the score at the top of the window. You can use a glutBitmapCharacter or a glutStrokeCharacter to display fonts in a GLUT window. Every time the Humvee hits a sphere the user gets 5 points and every time a tree is hit, the user loses 5 points. For the purposes of this assignment you can do the collisions by a simple distance check along the x-z plane between the origin of the Humvee and the origin of the tree or sphere. **Point distribution:** 1 point for placing and drawing the spheres, 2 points for collisions and having the trees fall and spheres change, 2 points for showing the score. (5 points)