Project 4 - Shadow Volumes

1 Goal

Draw the same kind of scene as in Project 1 (ray tracer) but this time taking advantage of the graphics hardware.

2 Input File

The input file format is exactly as in Project 1. As the output, produce an image of the scene where all of the primitives appear made of the same material as in Project 1 images, but the light source location is controlled by mouse movemetns (see next section).

3 Requirements

You should program a menu tied to the left mouse button. in two items The menu should contain the following items:

1. **Show shadows** - draw the primitives with shadows
2. **Show shadow volumes** - draw shadow polygons (all in one color) AND the primitives specified by the input file
3. **Primitives only** - just primitives, without shadows or shadow volumes

The image should be close to that obtained using your project 1 raytracer, with the exception of light location: in this project, the light location will be controlled by mouse movements. Use the idea similar to trackball.

First, compute the 'center point' of the whole scene. For example: compute average distance D from centers of spheres and centers of mass of triangles from the viewpoint and assume the center is along the ray from the viewpoint through the center of the screen, D away from the viewpoint. The light will always be D away from the center of the scene.

To compute the light location, use an imaginary sphere whose center projects to the center of the screen – exactly as in the trackball project. The coordinates of the point on the sphere computed using the mouse location will define the direction from the center of the scene to the light source. This will give the effect of light source that follows the mouse movements.

4 Remarks

Use smooth shading. The show shadow volumes mode may be useful for visual debugging your shadow polygon computation routine. One of the things you'll have to do is to ensure smooth transition between shadow and lit parts of the primitives. Take a look at the subsection 'mathematics of lighting' of the OpenGL programming guide, it might help to get this right.

5 New OpenGL commands needed

It's probably convenient to use quad strips for the shadow volumes. See `glBegin(GL_QUAD_STRIP);`

For setting the culling parameters: `glCullFace`, `glEnable/glDisable(GL_CULL_FACE)`.

For stencil buffer setup/use: `glClear` with `GL_STENCIL_BUFFER_BIT` argument, `glStencilOp`, `glStencilFunc`, `glClearStencil`, `glEnable/glDisable` with `GL_STENCIL_TEST` argument.

For depth buffer setup/use: `glDepthFunc`, `glEnable/glDisable (GL_DEPTH_TEST)`.

For enabling/disabling buffers for writing: `gl[Depth/Color]Mask`.

To set up lighting coefficients: `glLight*`.

To set up material properties: `glMaterial*`.

To add stencil buffer to your display: `glutInitDisplayMode`; include `GLUT_STENCIL` in the bitwise OR.

For drawing spheres: `gluSphere`, `gluNewQuadric`.

For setting global ambient coefficient: `glLightModel*(GL_LIGHT_MODEL_AMBIENT,...)`.

6 Grading

Send your code to the usual address, in the usual format. We'll run your code like this:

```
% proj4 < input_file
```

1. Primitives rendered correctly (not necessarily with shadows) - 25 points

2. Shadow volumes correct (will be tested by viewing in 'show shadow volumes' mode)- 25 points

4. Shadows correct - 25 points

5. Lighting equation coefficients set up correctly (the colors on the boundary between shadow and light on a sphere shouldn't appear discontinuous) - 25 points (available only if shadows work)