CSE 470 Introduction to Computer Graphics

**HW 3 In the Spotlight Due: Thursday, 4 April**

Instructor: Dianne Hansford ⋆ Spring Semester 2019

**Objectives**: gain experience with the following concepts

- Creating parameters for the Phong illumination model
- Gouraud shading in the vertex shader
- Drawing multiple objects with different ModelView matrices
- Setting input parameters to the lookAt and perspective functions
- Defining parametric surfaces and surface normals
- Building and rendering a triangulated surface
- Interaction and event handlers
- Graphics programming and debugging
- HTML5 programming

## Big Picture

You will develop an application that displays a shaded surface of revolution (SOR) and a spherical light source object. The surface may be rotated or zoomed via mouse controls. The user can select from two SORs, a cylinder or one of your own design. The surface and light object will be Gouraud shaded, using Phong illumination based on exact vertex normals. HTML user interface objects allows the user to select from three different material properties for the SOR, change the field of view, and choose between a fixed or rotating light position.

## Specifications

Build your program based on the code in “shadedsphere1” and “mousy” demos. The items below enumerate the specifications in a suggested progression of implementation.

1. Start with the shadedsphere1 program and integrate the mouse control functionality from the mousy program.

2. Add material properties
   (a) Create *three* distinct materials.
   (b) Add buttons (or other control) to switch between the materials in the HTML page.
   (c) Give these materials meaningful names to characterize them (not generic names like "material 1").
3. Add an emissive component to the Phong Lighting Model.
   (a) Each object sent to the vertex shader will need to set an emissive uniform (vector) variable.
   (b) In the final application, the emissive color vector should be black (off) for the SOR and white (on) for the light object.
   (c) Change the background color to a color other than white, so the object with emissive color is visible.
4. Viewing and Projection
   (a) The mousy code calls LookAt and sets the ModelView matrix.
   (b) Instead of ortho, use the perspective function from MV.js
   (c) Implement a slider to adjust the fov from 10 to 100 degrees. Initialize the fov to 60 degrees.
5. Surface of Revolution (SOR)
   (a) Create *two* surfaces of revolution: a cylinder and an interesting shape of your choosing (excluding a sphere). Both surfaces should be open at the top and bottom and be constructed in the following manner. Define a parametric generating curve – *generatrix* – in the $xy$-plane,

$$\mathbf{g}(t) = \begin{bmatrix} f(t) \\ t \\ 0 \end{bmatrix},$$

   where $f(t)$ is a function. The SOR is constructed by rotating $\mathbf{g}(t)$ around the $y$-axis, thus the surface is $\mathbf{s}(t, \theta)$ over the domain $\theta \in [0, 2\pi]$ and $t$ domain of your choosing (based on $\mathbf{g}$).

   (b) Create two integer variables defining the discretization of the generatrix and the rotational direction. Evaluate the SOR and store the vertices.
   (c) Calculate exact normals at the vertices.
   (d) Data management suggestion: Calculate the vertices and normals for both SORs at initialization. Keep track of the start and end indices in the vertices and normals data structure. Switching between the surfaces can be achieved by changing the start/end pointers for drawArrays.

6. Add a light object
   (a) Create the light as a sphere as done in shadedshphere1.
   (b) The light object vertices and normals can be placed in the same data structures that hold the SOR information. Record the start and stop indices.
   (c) Give the light object a white emissive vector, which in effect, overrides the rest of the light equation and just renders it white. Set the material properties.
   (d) Give the user two choices for the light object's position: a fixed position or rotating around the object. (For the rotating option, the light will *slowly* circle the SOR.) The light will need its own ModelView matrix.
   (e) The light object should be visible in the rotating mode, but not necessarily in the fixed mode.

**User interaction checklist**
All controls (buttons, sliders, etc) must be accompanied by descriptive text in the HTML page. The following user interaction is required.

1. SORs rotated and zoomed via mouse click and movements
2. Control (e.g., buttons) for selection of the SOR
3. Control for selection of the material properties
4. Control to change between a fixed and rotating light position
5. A slider to change the fov to change from 10 to 100 degrees

**HTML Page**

1. webGL canvas
2. Your name
3. Date
4. Program description
5. Resources you used

**Submission Guidelines**

1. Submit three files:
   lastName_hw3.html, lastName_hw3.js, lastName_geometry.js.
   (The geometry creation is separated from the graphics.)
2. Add your name to the top of each file.
3. Name your zip file: lastName_HW3.zip
4. Filename and Common folder references must be done as in the class examples.
5. Turn in your assignment on Canvas.

**General Tips**

1. Come to class – lots of tips and details will be explained.
2. Map out, step by step, how you want to tackle the HW.
3. Work on one piece of functionality at a time and test.
4. Print out intermediate and final values to test for correctness.
5. You can discuss the project with classmates, but be very careful about sharing code/variable names. Be sure you are aware of the Student Code of Conduct and Integrity policy. Links are provided in the class syllabus.
6. Start early and get help if needed.