CS1566 Assignment 2 (part 1)
Stitcher Algorithm

Out: Thu 09/13. Due: Wed 09/19 5:00pm

Problem 1
Your boss at Acme Graphics Corp. wants you to write a procedure to first compute and then render a sphere in OpenGL. His evil plan is to mutilate the sphere in a series of interactive simulations (e.g., make it grow a nose and a moustache), for which he will need direct access to the sphere geometry. Hence, the project specs say you are NOT allowed to use any GLU procedures; you need to generate and store the sphere geometry yourself, vertex by vertex.
The sphere is centered at the origin, and has a radius of r units. Because OpenGL can only display polygons, you are to split the sphere into v_s vertical stacks (along the Y-axis) and r_s radial slices (around the Y-axis; think latitude and longitude).

1a) Give a procedure (in pseudocode):

```
void make_sphere(float r, int rs, int vs);
```

to compute the vertices of such a sphere. For now, store the vertices in a big array `verts[MAX_RS][MAX_VS]`; when you code this up, you might want to use a generic data structure that can describe both a sphere and let's say a cone; you might want to also store the object faces. You may assume MAX_VS = MAX_RS = 50, i.e., your boss will never ask for a sphere with more than 50x50 stacks and slices (though he may ask for fewer). To get full credit, you need to spell out the math first and derive the parametric equations of the sphere (as in, write down the matrix multiplications). We expect you to use some matrix multiplication function (of your own, no need to spell it out) in your pseudocode.

1b) Give a procedure (in pseudocode):

```
void draw_sphere(int rs, int vs)
```

to render the sphere from the pre-computed array of vertices. Assume you have access to the verts array. You can assume OpenGL has already been initialized, camera, projections, callbacks are already set, and all you need to do in this procedure is draw each face as a GL_POLYGON (with either 3 vertices – triangular faces, or with 4 vertices – quad faces shown in the figure, as you wish). Again, you are not allowed to use any GLU functions. When drawing each face, make sure you're specifying the vertices in counter-clockwise order.

Problem 2

Your boss is particularly stubborn, and next, demands you compute and draw an upright cone. Again, you can not use any GLU procedures. The cone has its base centered at the origin and has height h and a base ray *r*.

Again, because OpenGL can only display polygons, you are to split the cone into patches, and stitch the patches together later. The split will be done in v_s vertical slices (think of them as meridians on the globe, along the *Y*-axis) and r_s radial slices (think of them as parallels on the globe, around the *Y*-axis).

2a) Give a procedure (in pseudocode):

```
void make_cone(float r, float h, int rs, int vs);
```

to compute the vertices of such a cone. For now, store the vertices in a big array `verts[MAX_RS][MAX_VS]`. You may assume MAX_VS = MAX_RS = 50.

As for Problem 1, to get any credit you need to first derive the parametric equations of the cone and spell out matrix multiplications etc.

2b) Give a procedure (in pseudocode):

void draw_cone(int rs, int vs)

to render the computed cone. Assume you have access to the verts array. You can assume OpenGL has already been initialized, and all you need to do in this procedure is draw each face as a GL_POLYGON (with either 3 vertices – triangular faces, or with 4 vertices – quad faces shown in the figure, as you wish). Again, you are not allowed to use any GLU functions. When drawing each face, make sure you are specifying the vertices in counter-clockwise order.

Problem 3
Give a procedure (in pseudocode):

void make_torus(float r1, float r2, int rs, int vs);

r1 = Distance from the center of the tube to the center of the torus.
r2 = radius of the tube.
rs = Number of circles along the tube.
vs = Number of circles across the tube.

For now, store the vertices in a big array verts[MAX_RS][MAX_VS]. You may assume MAX_VS = MAX_RS = 50. As for Problem 1, to get any credit you need to first derive the parametric equations of the torus and spell out matrix multiplications etc.

Problem 4.

4a) Given the points of a triangle are P1 = (1,1,1), P2 = (2,2,2), and P3 = (3,2,2), calculate the surface normal of this triangle. (See Linear Algebra recitation for cross-product definition and uses).

4b) Given a cube centered at the origin and extending from -1 to 1 along each axis, write the normals for each of the faces as (nx, ny, nz) triplets.

Next, still using the cube, please compute for each cube vertex the bogus quantity called a "vertex normal". A vertex normal is the average of all incident face normals at that vertext. Give the normals for each vertex.

Problem 5.

5. Write the pseudocode for calculating the normals of the vertices (!! You will need to also compute the face vertices along the way) of a sphere. Store them in a big normals[MAX_RS][MAX_VS] array.
Hint: use cross-product, then average incident face normals at each vertex.

Grading and Collaboration Policy
1a … 10pts
1b … 3 pts
2a … 8 pts
2b … 3 pts
3 … 3 pts
4 … 8 pts
5 … 5 pts
Total: 40 pts

You are not allowed to collaborate in any form on this assignment.

Extra Credit and Late Policy
For 10 points EC, wow your grader and write the pseudocode for a fractal structure of your choice, or some geodesic shape you fancy.

You cannot hand in late the algorithmic component of Assignment 2 (i.e., the solution to this handout).

Handing-In
You can type or handwrite your solutions, as you wish. If hand-writing, do your best to write legibly – you'll lose points if the grader can't read your writing. Hand in your solution to the graduate TA (either in person during TA hours, or to their mailbox) by the due date. Because we will post the solution on Tue at 5pm sharp so everyone can start coding the rest of the assignment, we will not accept late hand-ins. The grad TA will empty their mailbox at 5pm.