Procedural Meshes

CIS 460/560 Homework 3

Due Wednesday, November 6, 2013 at 11:59 am

1 Goal

You will get some experience in generating 3D objects procedurally given some basic information. You will have to create a mesh class to store the data, and render the objects with OpenGL.

2 Bigger Picture

You've made clouds and rendered them, then in the previous assignment you created furniture based on cubes, spheres, and cylinders. We will now expand on the potential contents of your room by creating mesh-based objects. You will need to write a mesh data structure and implement the creation of two kinds of procedural mesh objects: extrusions and surface revolutions. We are still using OpenGL at this point, but in assignment number four we will be also add a raytracer to our rendering box of tricks. The result will be prettier pictures and a potential loss of hair in some students.

3 Software Environment

Same as assignment 3. Use the code from assignment 3 as starter code for this assignment. You may have to fix bugs in the previous version, or change some features' implementations if necessary so even though you are using a codebase that has compiled in the Moore 100 lab previously you will need to check that this is still true.

4 Supplied Code

You will build this assignment off of the cod you wrote for assignment two.

5 Requirements

5.1 Mesh Data Structure (30 points)

This is the most crucial requirement in the assignment. *Failure to comply with this requirement will result in a zero.* You must make a mesh class. This class will store an arbitrary set of triangles that represent the geometry of an object. The face list, vertices, and corresponding normals should all be stored. This class exists for the purpose of perpetual storage from frame to frame. If you are creating the mesh or calculating normals as part of the display function (or its calls), you are not doing this part correctly, only the actual rendering of the mesh should occur in a display function.

The mesh will get its data from the two object types below. Both use file input. Your application should allow more than one mesh-based object to be used at the same time, and using a new file should not affect old meshes already in the scene graph. Your program should gracefully handle file name and file format errors (i.e. file does not exist, file is not in extrusion/surfrev format).

Using the two techniques below you will generate a set of vertices and a set of faces built from those vertices. For each face, you will need to calculate the outward facing normal and store it. There is an efficient way to store this information. When drawn, the mesh should put its vertex data into the proper VBO, normal data into the proper VBO, have some color data of your choice also stored, and use the face list to generate the IBO.

5.2 Shader Augmentation (15 points)

Add the following features to your shader:

- A specular component to the lighting in your fragment shader (i.e. change your shading from Lambertian to Phong).
- A uniform for light color

5.3 Extrusions (25 points)

An extrusion is essentially a prism. A polygonal base will be defined in a file, which your program will take as input. You will need to test whether or not the base is a convex polygon. If it is, the endcaps must be displayed, if it is not a convex polygon, only the sides should appear on the screen. You must then generate the sides of the extrusion automatically. Make sure to generate sensible normals for your new mesh. The normal of each face should be perpendicular to the face and point outward. The format of the file will be as follows:

- The first line has the word "extrusion."
- The second line has the length of the extrusion from base to base.
- The third line has the number of points in the polygonal base. The base is a polygon, so the last point should be the same as the first point.

- The rest of the file contains the (x, z) coordinates of the base polygon with one point per line and whitespace separating x- and z- coordinates.

A sample file is supplied on Canvas. Your application will be tested with a file that you have not seen, so make sure you do not tune to any particular example when you debug. You must use your mesh class for this.

5.4 Surface Revolutions (25 points)

A surfrev takes a polyline and rotates it about the Y-axis to create a 3D object. The rotations are made in discrete steps; for instance every 30 degrees construct sides. If the polyline ends at the Y-axis, the object will be closed at the end by its nature. If the polyline does not end at the Y-axis, you must construct the endcaps automatically. Make sure to generate sensible normals for your new mesh. The normal of each face should be perpendicular to the face and point outward. The format of the input file will be as follows:

- The first line has the word "surfrev."
- The second line has the number of radial slices in the revolved mesh. This number should be at least 3, or your code should report an error.
- The third line has the number of points in the polyline.
- The rest of the file contains the (x, y) coordinates of the points of the polyline, one point per line using whitespace as above. Your code should report an error if the x-coordinate is negative.

As with extrusions, a sample file is supplied on Canvas.

5.5 Integration with Homework 3 (5 points)

Expand you parser from Homework 3 with the following geometric object:

mesh
filename.dat
image.bmp
xIndex zIndex
rotation
xScale yScale zScale

The main change here is *filename.dat*. This tells you what file to open for this mesh. The second filename replaces color options from the past assignment ONLY for meshes. It is optional to implement texturing. You need to make the objects stack appropriately along with the furniture from the prior assignment and interact the same as any furniture object.

6 Extra Credit (20 points maximum)

The following features can be added for extra credit. The point values given are the absolute maximum, and actual points awarded are at the discretion of the TA/graders.

6.1 Texturing (20 points max)

Add texturing to meshes and all objects from homework 3, including the floor. You will have to add a vertex attribute to your shaders to handle texturing coordinates, as well as learn to use samplers and how to bind textures. A lot of code changes will be required to do this well, so make sure you only attempt this after completing the project requirements. Also, save a copy of the code that completes the project requirements so you don't back yourself into a corner on homework 4. 10 points will be awarded for successfully showing textures with correct colors, while20 points will be granted if the textures are never found to be warped or in other ways badly applied.

6.2 Object design (20 points max)

Create your own extrusion and surfrev files that create interesting meshes. The amount of extra credit earned depends on how impressive the designs are to the grading staff.

7 Example Usage

A grader reopens the classroom and adds some extra geometry using the new surfrevs and extrusions. He adds a cup stacked on the front desk of the classroom. The cup is a mesh object created through surface revolution. He uses extrusions to create "textbooks" and stacks them on the floor. Satisfied, he closes the program. Keep in mind the main amount of work was in homework 3; comparatively there's not a lot new here.

8 Deliverables

You will submit a zipped folder via Canvas containing your Visual Studio project and a readme file with anything you think the graders need to know. Please make note of the extra credit you implemented, or you may not get credit for it. As always, please test your code in Moore 100 before submitting!