Program 4: Procedural Geometry
100 points

Assigned: March 25, 2014
Due: April 2, 2014

Assignment: **Multiple Parts**

To make the following easier to implement, you should create a program separate from the prior assignments. You will pull this assignment into our ongoing framework later. The fastest way to make a new testing "framework" is to COPY your current room editor and strip out all the scene graph functionality and instead focus on rendering one mesh at a time.

Mesh Class

This class will store an arbitrary set of triangles that represent the geometry of an object. You will use the face list structure discussed in class. 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 your preparation to draw, you are not doing this part correctly.

The mesh will get its data from the two object types below. Both use file input. Your mesh class needs to be created such that a mesh is not required to be unique. That is to say that multiple meshes (probably with different geometry) can exist in the same program.

Using the two techniques below you will generate a set of vertices and the corresponding set of faces. For this assignment, calculate a normal for each face (requiring vertices with duplicate positions). Later we will use per-vertex lighting for smoother appearances.

Extrusions

An extrusion is essentially a prism. A polygonal base will be defined in a file, which your program will take as input. Using the resulting top and bottom bases as endcaps, you need to create faces for the sides of the prism. You will need to test whether or not the base is a convex polygon. If it is, the endcaps should be triangulated and rendered, if the base is nonconvex the ends should not be capped. Be sure to also triangulate the sides of the extruded shape.

The format of the file will be as follows:

- The first line is the word "extrusion".
- The second line has the height of the extrusion from base to base.
- The third line has the number of points in the polygonal base. The last point is 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.

Samples will be available on MyGCC.

Surfrevs

A surfrev takes a polyline and rotates it about the Y-axis to create a 3D object. The rotations are made in discrete steps called "slices." After rotating by the discrete amount, the transformed polyline's points are connected with the previous rotation's points to create the sides of the shape. If the polyline ends at the Y-axis, the object will be naturally closed at a point. If the polyline does not end at the Y-axis, a convex endcap will need to be created and rendered.

The format of the file will be as follows:

- The first line has the word "surfrev".
- The second line has the number of "slices" in the revolution. This number should be at least 3 or the mesh would be flat and not make much sense.
- 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. X-coordinates may not be negative!

Again, samples will be made available on MyGCC.

Rubric: To receive credit, submit a .zip file containing the following items by 11:54:59 PM on the due date listed above:

1. **Source code (90 points):** Your entire Visual Studio project, including the source code and any non-standard headers or source files on which your program depends.
   Your code must compile, link, and execute using Visual Studio 2010 or 2012: no compile equals no credit.

   In addition, you must include a comments section at the beginning of each of your source code files that provides a description of the code and its intended purpose. For example,

```
/*
  Author:  Cory D. Boatright
  Course:  COMP 361, Computer Graphics
  Date:    March 11, 2014
  Description:  Par time: 30 minutes
*/
```

   The following criteria will be used to grade your submission:

   - Does the code compile?
   - Does the code function according to the problem specification?
   - Is there an appropriate comments section at the beginning of each file (similar to the one shown above)?

- Is the code readable and well-formatted? Is it well-documented and clear?

The available points are distributed as follows:

Correctness:	80 points	*Supports course outcome 5*
Comments:	5 points	
Organization:	5 points	

A program that does not compile or link will not be graded.

2. **Design decisions document (5 points):** A text file describing the design decisions that you considered while implementing the required classes. You will not be graded on your choices, only on the completeness of your justification for each choice.

   Supports course outcome 2

3. **Supplemental information (5 points):** A text file with answers to the following questions:

   (a) How long did you spend on each part of this assignment?

   (b) Was the assignment difficult for you? Why or why not?

   Feel free to expound or to be brief. You will not be graded on the responses themselves, only on the presence or absence of the responses. Your answers will be used to help improve this assignment in future incarnations of the class.

Submission: Programs must be submitted electronically via LMS before 11:54:59 PM on the due date listed above. Be sure to upload your files correctly the first time. If you have any problems prior to the submission deadline, please contact the instructor.
Extensions will not be granted for technology-related issues. Leave yourself enough time to complete the assignment, submit the assignment via LMS, and contact the instructor if you run into problems.