

CS 175: Assignment 7

Meshes and Subdivision

Due:

Monday, November 9, 11:59pm

Assignment Objectives

The purpose of this assignment is to use a mesh data structure and implement Catmull-Clark subdivision.

We provide you with a mesh data structure, a mesh file, and some shaders. In addition, you will need an assignment 3 solution, which gives you an arcball to move the camera around the object and draws a floor. We provide a file “mesh.interface” with a simple interface description of the `Mesh` class, and a sample usage of the `Mesh::VertexIterator` class which you will need in order to walk around the 1-ring of a vertex.

Task 1

Read in the cube mesh from the provided file “cube.mesh”. The mesh will just be a cube with 6 quads as faces. Draw the mesh using the RBT provided in the code representing the object frame for the mesh. You should draw each face as two triangles, with “flat shading” (when drawing, use the same normal for all vertices in a face). Use the mesh data structure to obtain the proper normals for each face.

Task 2

Compute one “average normal” for each vertex. An average-normal at a vertex will be the average of the normals of all the faces incident to the vertex. Draw the mesh with “smooth shading”, using the average-normal of each vertex.

Task 3

Register an *idle* function using `glutIdleFunc` and use this idle function to “animate” the vertices of the cube. In particular just scale the object coordinates of each of the cube’s vertices by a periodic, time-varying scalar.

Task 4

Implement subdivision. Use the Catmull-Clark rules to calculate the coordinates for new face-vertices, new edge-vertices, and new vertex-vertices. Place these coordinates in the appropriate slots in the data structure. Then call the subdivide routine. This routine will use the coordinates you provided for face-vertices, edge-vertices, and vertex-vertices to compute the new, subdivided mesh. Iterate this process 3 or 4 times.

Once you are done with the subdivision, you will compute average-normals for the final vertices, and use these normals when drawing the mesh using “smooth-shading”.

Your final version will include the following hot keys:

- ‘s’ will toggle between smooth and flat shading.
- ‘+’ will increase the number of subdivision steps applied to the cube before being drawn.

- ‘-’ will decrease the number of subdivision steps applied to the cube before being drawn.
- ‘<’ will half the speed at which the cube deforms.
- ‘>’ will double the speed at which the cube deforms.

Note:

Remember that you will be starting from the simple cube and applying subdivision every time that your “idle” function is called. To do this you can use two meshes: a “reference” mesh that holds the original cube unmodified, and a “temporary” mesh. Every time that your idle function is called, you can create a temporary mesh to be a copy of the reference mesh (a copy constructor and copy assignment operator is provided in the `Mesh` class), you can then subdivide the temporary mesh and draw it. This way the reference mesh is unchanged and still represents the simple cube, so you can use it the next time you need to draw a frame.