CS 354 Project 2: Triangular Mesh and Deformation

This project asks you to implement a basic triangular mesh data structure and perform loop subdivision. The task is that you are asked to start from a base-mesh, and perform subdivision to generate a dense mesh and render it. The subdivision implementation should be fast so that when modifying the base-mesh, the dense mesh can be recalculated and rendered on-the-fly.

Loop subdivision. there are many good tutorials about loop subdivisions, e.g., Footnote1 [1], Footnote2 [2], and Footnote3 [3]. Please get familiar with the formulas on how to determine the locations of new vertices.

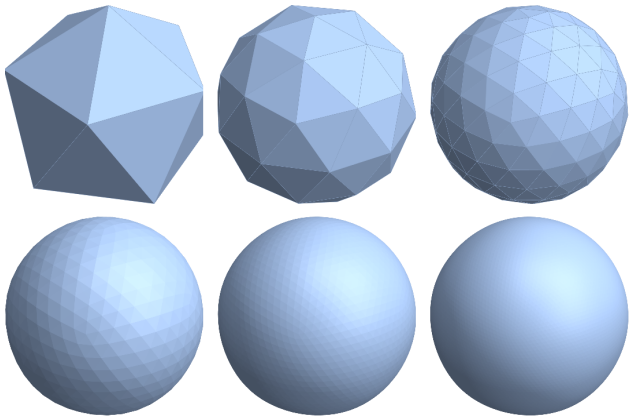

Figure 1: An example of loop subdivision.

Your tasks. Your tasks include a basic data structure for triangular meshes and basic operations for refining the mesh topology and computing vertex locations of refined meshes. These include

- A data structure that allows you to find two neighboring triangles along an edge. There are many online resources, such as [4] and [5].
- An operation that splits a triangle into four triangles. For simplicity, it is recommended to generate a new mesh at each iteration.
- An operation that determines the locations of mesh vertices.
- Provide a simple interface for specifying and modifying the base mesh, e.g., changing the location of the third vertex.

Base mesh. We will test your code on the following tetrahedron:

$$V_0 = (0, 0, 0); \quad V_1 = (1, 0, 0); \quad V_2 = (0, 1, 0); \quad V_3 = (0, 0, 1).$$

Programming language. It is recommended to use the programming framework of project I.

Hints.

[1] http://mrl.nyu.edu/ dzorin/cg05/lecture11.pdf
[2] http://www.cs.cmu.edu/afs/cs/academic/class/15462-s12/www/lec_slides/lec07.pdf
[3] http://www.cs.utexas.edu/ huangqx/CS354_Lecture_14.pdf
[4] http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml
[5] https://homes.cs.washington.edu/ edzhang/graphics/a2.html

- The most time-consuming part is the half-edge data structure and the splitting operation. One way to debug is to just split the triangles, to see if you are generating topologically correct triangular meshes.
- Pay attention to singular vertices, as we have a different rule for determining the vertex locations for singular vertices.
- Note that the vertex normals can be calculated from the refined mesh, or use a closed-form formula. At the same mesh resolution, it is expected that using the closed-form formula leads to better results.

Grading.

- (60 pts) Half edge data structure and triangle splitting.
- (20 pts) Vertex locations of new triangular meshes.
- (20 pts) An interface for modifying the base-mesh.
- (**20 bonus pts**) Change the subdivision rule to create sharp edges.