

# Project 4: Subdivision

## 1 What to do?

The goal is to implement the Loop and Butterfly subdivision schemes for triangular meshes. The input mesh will be given in the same format as in Project 2. The first two entries will be integers, the triangle count and the vertex count. Then, there will follow a triangle table: a number of lines, each listing three integers (ID's of vertices bounding a triangle). Finally, there will be the vertex table: coordinates of all the vertices. For example, here is an input file that represents a tetrahedron:

```
4 4
0 1 2
2 1 3
2 3 0
0 3 1
0.000000 0.000000 1.000000
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 0.000000
```

You can assume that the triangles are consistently oriented and the input mesh is manifold.

Your program *must* compile on the CoC Linux machines and run as follows:

```
% project4 number_of_subdivisions kind_of_subdivision input_file_name
```

`kind_of_subdivision` will be just one letter, 'L' (Loop) or 'B' (Butterfly). `number_of_subdivisions` is an integer, 0 (no subdivisions at all) or more. We don't give any bound on the number of subdivisions, your code should attempt to do any number of subdivisions (when testing your code, we'll generally try to keep the number of vertices within 500,000, but please don't hardcode this). You should send a tar file containing all the files (they should extract to '.' and compile with 'make'). Projects which do not behave as described above (and below) will not be graded.

Your programs should display the result (i.e. mesh subdivided the specified number of times) using **FLAT SHADING** (so that we can see the individual triangles if the mesh is coarse) and implement a menu with the following items:

1. zoom in: of course, this one should zoom in :)
2. zoom out: zoom out
3. toggle animation: should start/stop rotating the output (in exactly the same way as the skeleton code for project 2 does)

Use display lists to speed up rendering. Of course, your code should display things neatly: scale the output mesh so that it nicely fits into the window. For

the Loop subdivision scheme, use weights  $\frac{7}{16}$  (for central vertex) and  $\frac{3}{16}$  for all its neighbors (cf subdivision notes, page 7, just before Section 4) for degree-3 vertices. For all other degrees, use weights  $\frac{5}{8}$  (central vertex) and  $\frac{3}{8n}$  (all surrounding vertices) where  $n$  is the degree of that vertex.

## 2 Grading

Half of the grade for the Loop scheme and half for the Butterfly scheme.

Here are things we are going to look at:

1. Execution speed (should never be more than 15 seconds if number of vertices times  $4^{\text{number\_of\_subdivisions}}$  is around 500,000 or less): 50%
2. One subdivision step works and looks OK: 50%

## 3 Deadline

March 31 midnight. Usual  $3^{N-1}$  late penalty applies.