Kevin King
Professor Zhu
COSC 77: Computer Graphics
Assignment 1 - Technical Implementation

Step 1: Add mid-point vertices and triangles

I looped through old_tri, and for each edge of a triangle, I checked in edge_odd_vtx_map() if that edge was already stored and, if so, continued. I then added three new vertices (located in the middle of each edge) to new_vtx. Within the same initial for-loop for each old triangle, but outside of the second for-loop that looked at their edges, I added the four new triangles to new_tri (starting by taking each new vertex and making the four new triangles using the original old vertices), making sure that the orientation was consistent. From there, I built the edge_tri_map by adding edges to their incident triangles from old_tri.

Step 2: Update position for each new mid-point vertex

I looped through edge_odd_vtx_map, in which the first element was the Vector2i edge and the second was the odd vertex. From there, I found the two end-point vertices A and B in old_vtx and then the two opposite-side vertices C and D using the “find_opp_vtx” written on lines 7-16 of the program. I then update the new position using the weighted average formula given to us for odd vertices.

Step 3: Update vertices of each old vertex

I started by building the initial vtx_vtx_map using the code from the tutorial_mesh main.cpp. I then copied all the vertices from new_vtx into the new_even_vtx array to store the new positions without affecting the old ones. I looped through vtx_vtx_map, determined the beta value based on the number of neighboring vertices of a specific vertex, and initially set smoothed_pos using part of the weighted average formula given to us for even vertices. From there, I looped through the neighboring vertices and updated smoothed_pos using the rest of the weighted average formula with beta. I then updated the position of the vertex in new_even_vertex. Lastly, I copied new_even_vtx back into new_vtx to synchronize the information.