Sooyeon Yang (sy22975), Coco Zhou (ccz247)

Catmull-Clark and Loop Subdivision

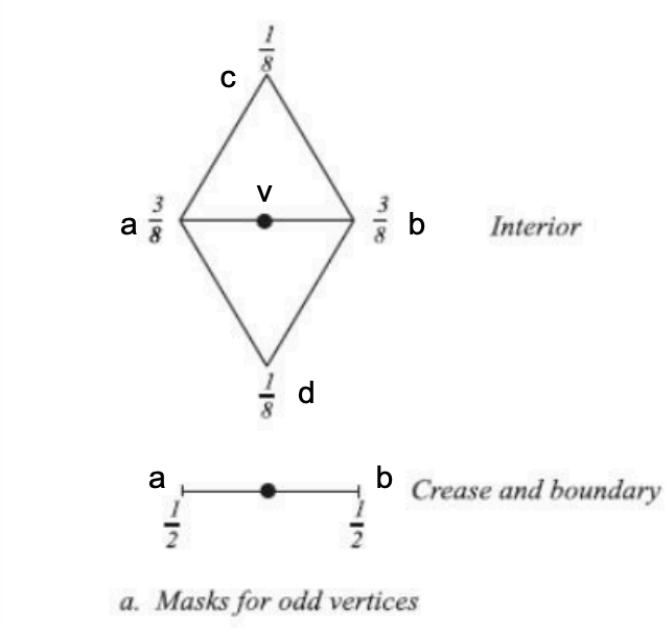
Section 1:

Catmull-Clark and Loop Subdivision are subdivision techniques used to divide polygonal meshes into smaller polygons by changing the connectivity and geometry of the mesh, creating an overall smoother look. However, Loop Subdivision operates on triangle meshes while Catmull-Clark operates on quad meshes, and they have different methods for updating the mesh.

Our goal for this project was to implement both subdivision algorithms in a way that handles extraordinary vertices, and displays the different subdivision levels alongside the limit surface.

Our Loop Subdivision algorithm approximates the shape of the original mesh by splitting faces, adding new vertices (odd vertices) and modifying pre-existing vertices (even vertices). Essentially, our algorithm splits every existing face in our mesh into four new faces using a set of rules to compute the odd and even vertices that make up these subdivisions.

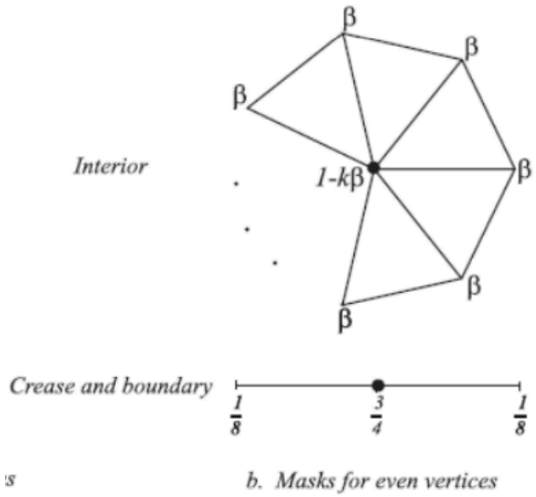
Interior odd vertices are calculated as: $v = \frac{3.0}{8.0} * (a + b) + \frac{1.0}{8.0}(c + d)$, where a and b are vertices on the given edge and c and d are shared neighbors of a and b. Boundary odd vertices are calculated as: $v = \frac{1.0}{2.0} * (a + b)$.



a. Masks for odd vertices

Interior even vertices are computed to be: $v = v * (1 - k\beta) +$ (sum of all k adjacent vertices) * β and boundary even vertices are computed to be: $v = \frac{1.0}{8.0} * (a + b) + \frac{3.0}{4.0}v$, where k is the number of adjacent vertices and $\beta = \frac{1.0}{n}(\frac{5.0}{8.0} - (\frac{3.0}{8.0} + \frac{1.0}{4.0}cos\frac{2\pi}{n})^2)$.

These formulas calculate the new vertex positions based on the number of adjacent vertices, and they already account for any potential extraordinary vertices.



Our Catmull-Clark algorithm is another approximation algorithm, which follows a similar general structure to our Loop Subdivision algorithm by subdividing each quad face into four new faces. Differing from Loop Subdivision, this algorithm creates each subdivided face from face points and edge points, where face points are the barycenter of vertices incident to the face and edge points are the barycenter of adjacent faces' vertices and vertices incident to the edge. From these face points and edge points, new vertices are calculated from the original vertices.

Face points can be calculated by averaging four vertices that make up that face. Edge points were computed using the following formula: $e = \frac{v1+v2+f1+f2}{4.0}$, where v1 and v2 are the vertices on the edge and f1 and f2 are the two adjacent face points of the faces sharing that edge. If the edge is a boundary edge, $e = \frac{v1+v2+f1}{3.0}$ since it will only have one face using that edge. The formula for calculating the new vertices from the original vertex, V, is: $v = \frac{(F+2E+(n-2)V)}{n}$, where n is the number of faces incident to V, F is the average of all face points of faces adjacent to V, and E is the average of all edge points of edges incident to V. The subdivided faces are created by connecting these newly computed vertices with the corresponding edge points and face points to form four new faces.

Section 2:

Both Loop Subdivision and Catmull-Clark require some sort of adjacency data structure to construct subdivisions, and typically half-edge data structures are used for this purpose. However, our implementation uses a custom-made adjacency data structure that utilizes maps and arrays because we are more confident with them. The adjacency data structure for both subdivision techniques keeps track of a vertex map, edge face map, face edge map, and list of faces, which differs slightly in construction due to the nature of triangle and quad meshes. Loop

Subdivision will also have a vertex-adjacency map while Catmull-Clark will have a vertex face map and vertex edge map (More details are in code comments).

For each iteration of Loop Subdivision, we loop through every edge of the mesh and compute the odd and even vertices based on the rules mentioned above in Section 1, storing each separately. Then, we rebuild the mesh by connecting the previously computed even and odd vertices to form new faces and edges. We also recompute each data structure in our adjacency data structure after each iteration, and the mesh's positions and normals are created and set after all iterations are done.

For each iteration of Catmull-Clark, we loop through each face to compute all the face points since these points are necessary for the computation of all other new points. Then, we loop through each edge in the mesh to compute the edge points. After that, we use the formula specified in Section 1 to find the new positions of the original vertices of the mesh. Next, we construct the new quad faces and recompute each data structure in the adjacency data structure after each iteration. After all iterations are done, we triangulate the new faces we've constructed and compute the normals for each face so that we can draw them correctly. Then, the mesh's positions and normals are updated.