Assignment 1

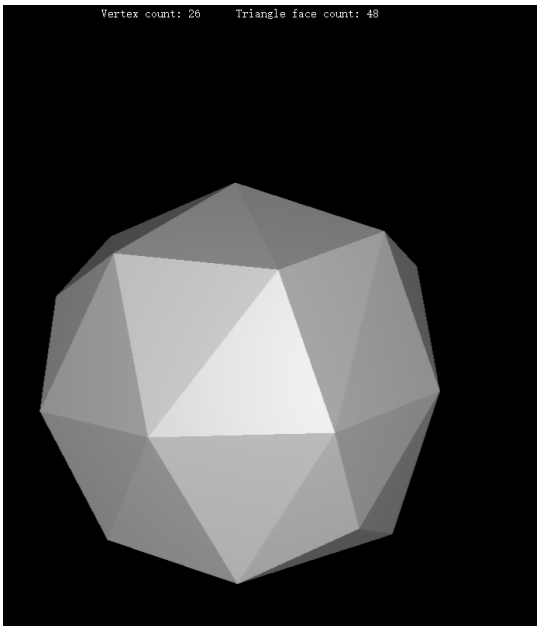
(Since my computer system doesn't support VS code and the browser can't visualize the obj file of the html file either, so I use pdf with multiple perspectives to show the results at this time.)

Loop Subdivision

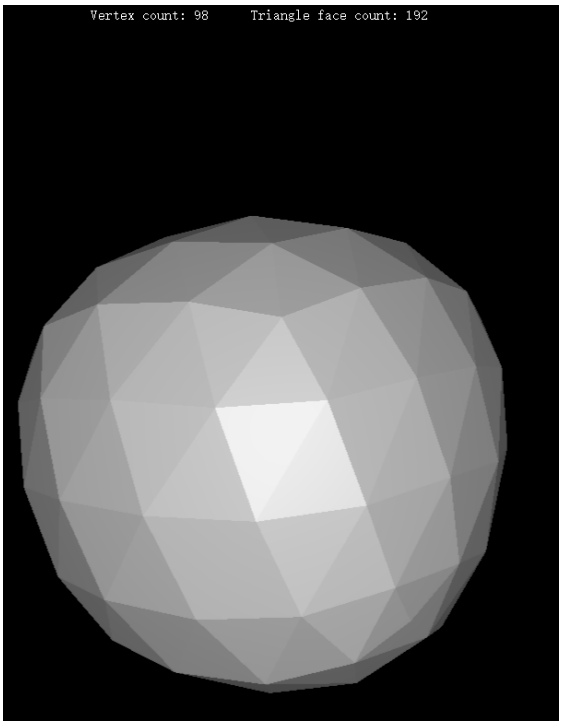
Data Structure:
We use the regular adjacency list to store the connection information between vertices and edges.

Algorithms:
As the hint code, we apply Loop subdivision to the input mesh for the specified number of iterations. We use "iterations" to denote the number of iterations we want. Then, we calculate the new positions of odd vertices and the even vertices respectively.
For the odd vertices, for the interior case, we make it as 0.75*mean(v0,v1) +0.25*mean(v2,v3). For the boundary case, we make it as mean(v0,v1).
For the even vertices, for the interior case, we make the new position of v as (1-kB)*v + B*(position of every adjacencies). For the boundary case, we make the new position of v as 0.75*v + 0.25*mean(position of the two adjacent vertices at the boundary).
And then we stack the odd and even vertices together.

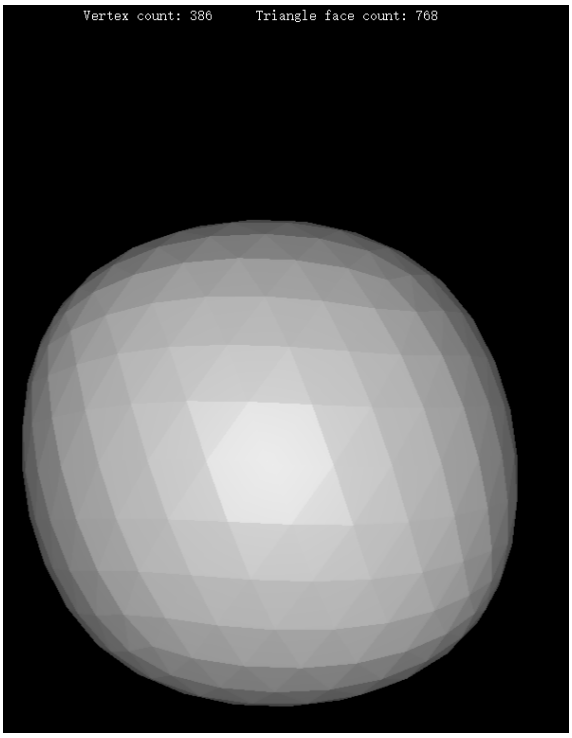
Visualizations:
Subdivision 1 time for the cube(0.006s):



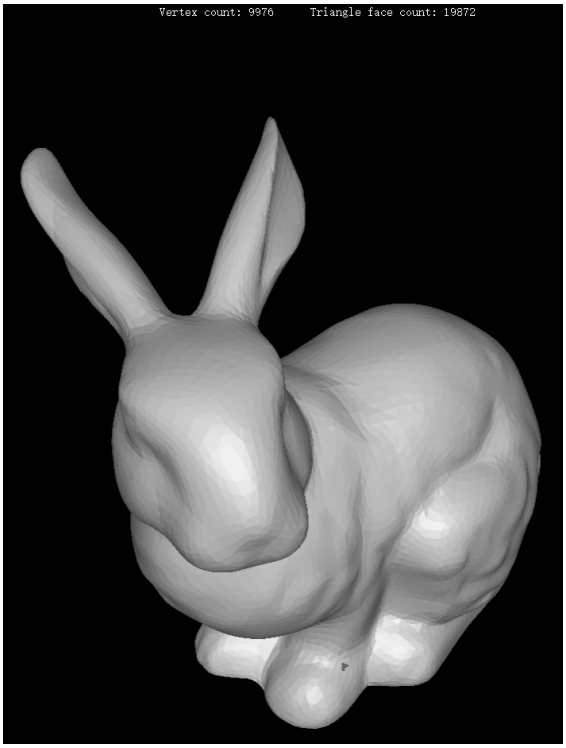
Subdivision 2 times for the cube(0.008s):



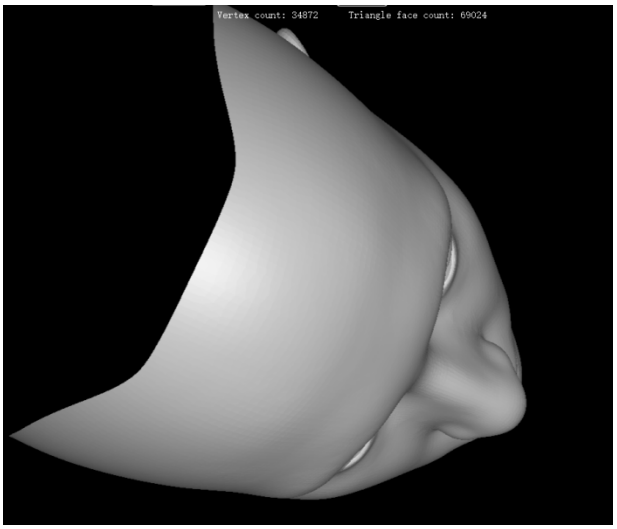
Subdivision 3 times for the cube (0.015s):



Subdivision 1 time for the bunny.obj(0.19s):



Subdivision 1 time for the face.obj (0.60s) (the boundary performs well):



Quadric Error Mesh Decimation

Data Structure:

Vertex Class: Represents a vertex in the mesh. Each vertex has an index (idx), a position (pos), a set of adjacent vertices (adj_vs), and a quadric error matrix (quadrics) associated with it.

Edge Class: Represents an edge in the mesh. Each edge is defined by two vertex indices (v1, v2), has a set of adjacent faces (adj_fs), a combined quadric error matrix from its vertices (quadrics), and properties to store the optimal vertex position (min_vert) and the minimum error (min_err) after collapsing the edge.

Graph Class: Represents the entire mesh. It contains a list of vertices (verts), a dictionary of edges (edges), and a list of faces (faces). The class provides methods to check if an edge is interior and to get opposite vertices for a given edge.

Algorithm:

For each edge in the graph, if it's an interior edge, its quadric error matrix is computed by summing the quadrics of its vertices.

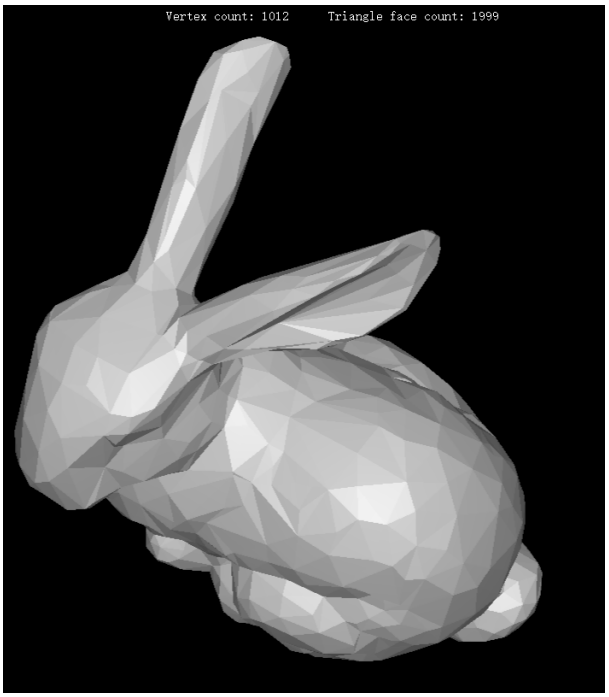
Then, the optimal position for collapsing this edge (to minimize the error) and the error value are calculated. A min-heap is created from all the edges based on their minimum error values. This heap is used to prioritize the edges for collapsing.

The algorithm repeatedly collapses the edge with the lowest error from the heap, aiming to reduce the face count of the mesh to the target face_count. Collapsing an edge involves merging its two vertices into one, updating adjacent vertices and faces, and recalculating the quadric error matrices for affected edges. After reaching the desired face count or exhausting the collapsible edges, the algorithm reconstructs the mesh from the remaining vertices and faces.

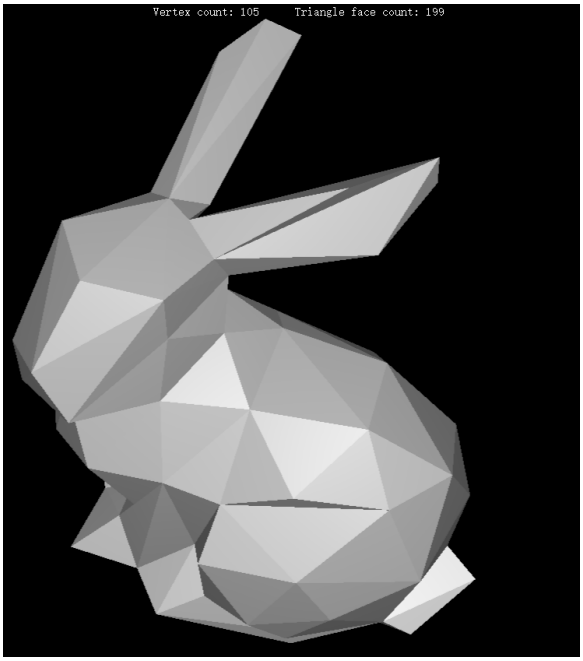
The vertices are re-indexed, and the faces are updated to reflect the collapsed vertices. The simplified mesh is returned, which should have fewer faces than the original while attempting to preserve the overall shape and features based on the quadric error metric.

Visualizations:

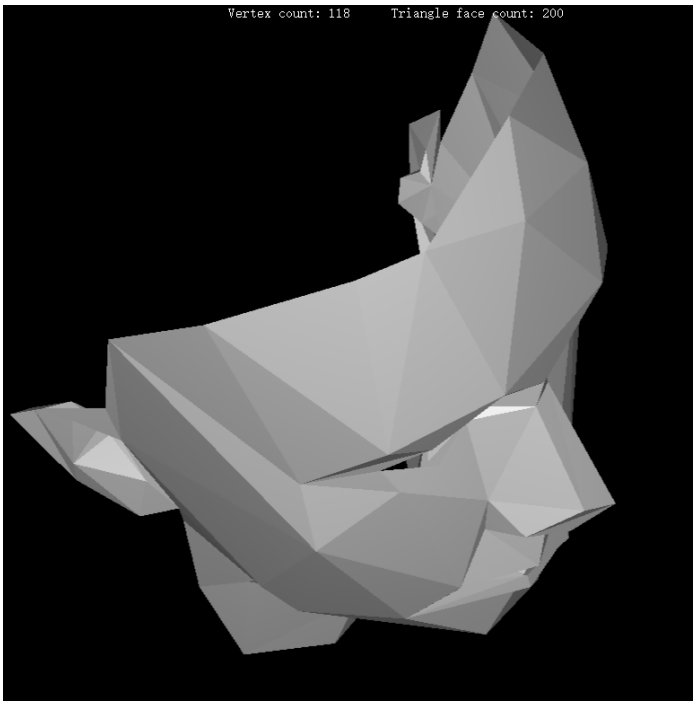
Decimation to 1999 faces for bunny.obj(2.61s):



Decimation to 199 faces for bunny.obj (3.77s):



Decimation to 200 faces for face.obj(12.77s):


Vertex count: 118
Triangle face count: 200