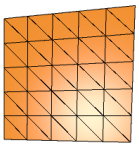
Computer Graphics (COL781) Assignment 2
Mesh Processing

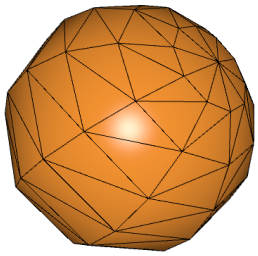
Parth Patel (2021CS10550)
Tanish Singh Tak (2020EE10560)

March 7, 2024

§1 Simple meshes (plane and sphere) created by our code



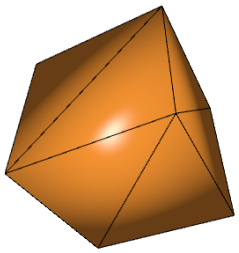
- 5x5 grid



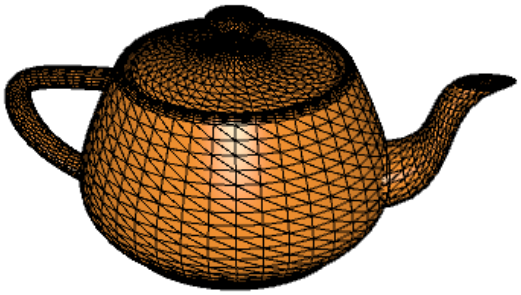
- sphere(longitude, latitude = 10,10)

§2 Cube, Teapot, and Bunny meshes

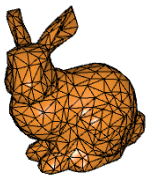
- Cube



- Teapot

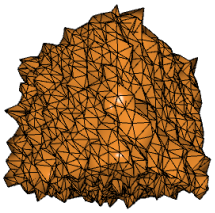
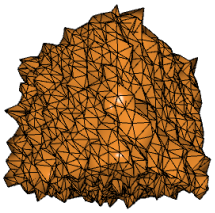


- Bunny

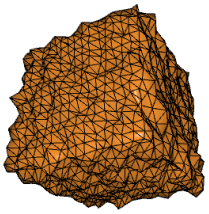
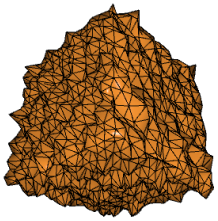


§3 Naïve and Taubin smoothing on the noisy cube mesh

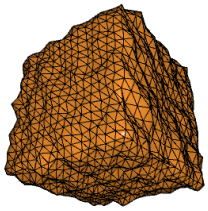
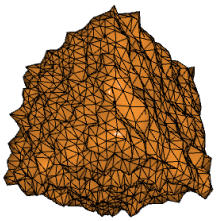
Iterations —— Naive smoothing result —— Taubin smoothing result

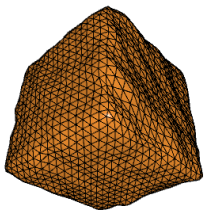
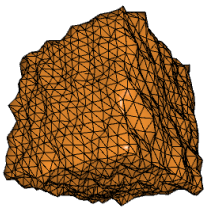
- 0

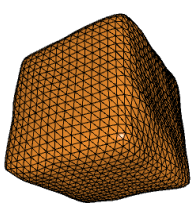
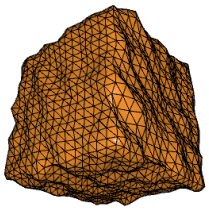
- 1

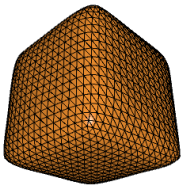
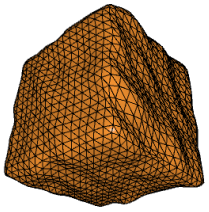
- 2

- 5

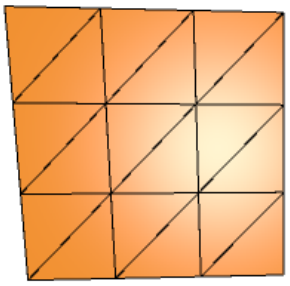



- 10

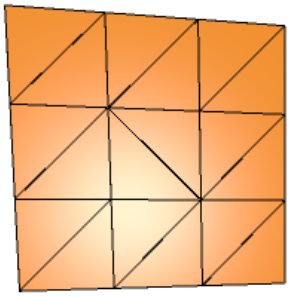



- 25

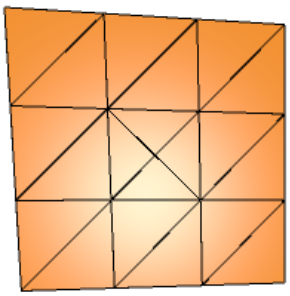
§4 Edge flip, edge split and edge collapse on the 3×3 grid



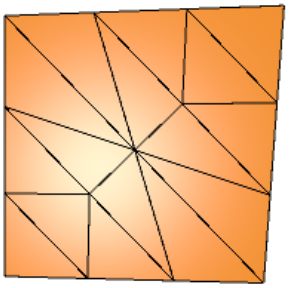
- Original



- Flip



- Split

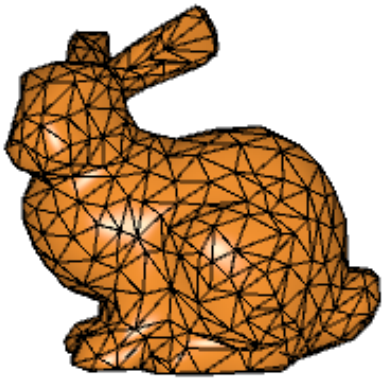


- Collapse

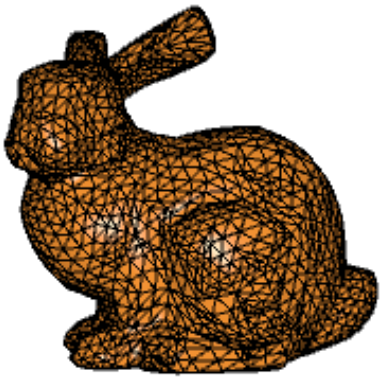
§5 Loop subdivision

§5.1 Bunny

number of subdivision iterations —— result



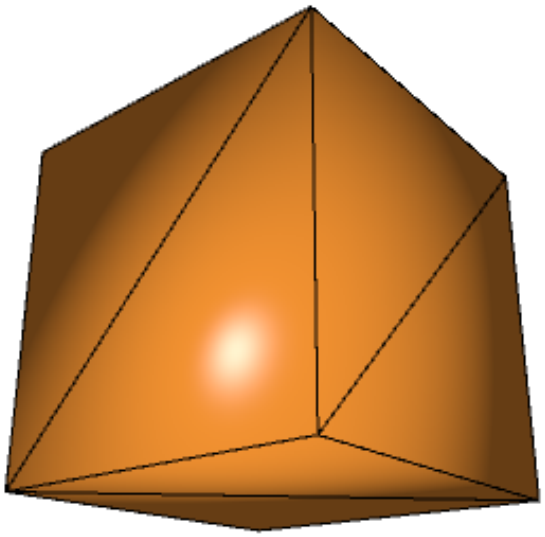
- 0



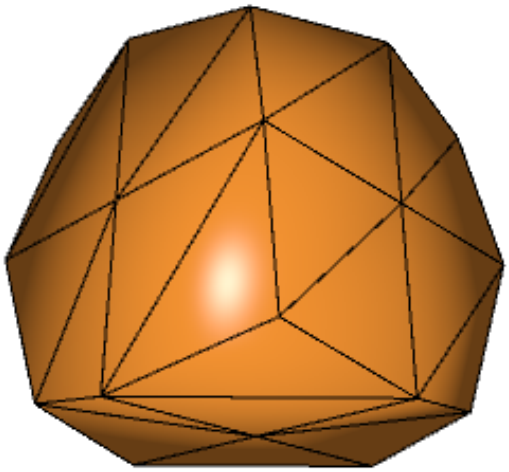
- 1

§5.2 Cube

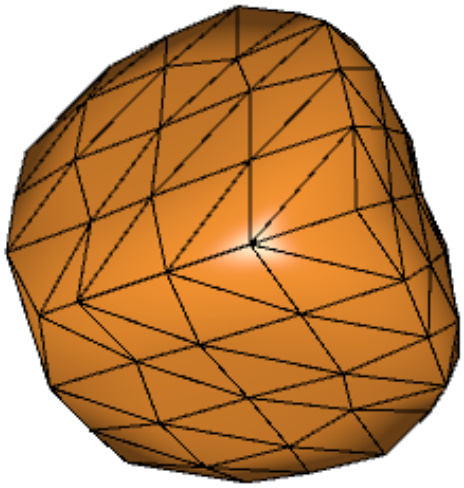
number of subdivision iterations —— result



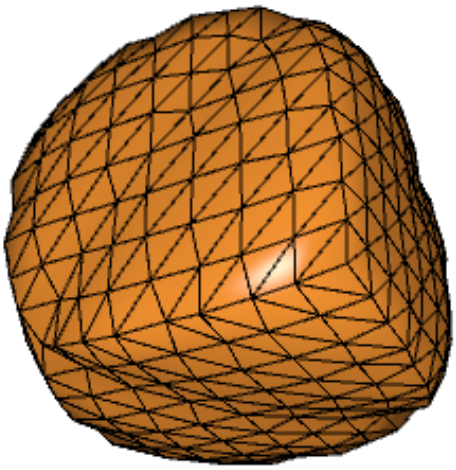
- 0



- 1



- 2



- 3

§6 Testing mesh connectivity

We will be testing a triangular mesh half-edge data structure for connectivity. We want to ensure that the data structure correctly represents the connectivity between vertices, edges, and faces. Here are the **invariants** that we are checking in our testing function -

- **Vertex-Edge Connectivity**: Verifying that each vertex has a reference to one of its incident half-edges. Ensuring that each half-edge has a reference to its starting vertex.
- **Edge-Face Connectivity**: Checking that each edge has a reference to one of its incident faces. Confirming that each face has a reference to one of its incident half-edges.
- **Next and Previous Half-Edges**: For each half-edge, we make sure its next and previous references form a valid loop within a face.
- **Opposite Half-Edges**: Ensuring that opposite half-edges (belonging to the same edge) correctly reference each other.
- **Consistent Orientation**: Checking that the orientation of the half-edges is consistent within a face (e.g., all half-edges around a face should follow a consistent order).

Additional things we can add to the testing function -

- **Boundary Handling**: If the mesh has boundary edges, check that the data structure handles them correctly. For instance, the next and previous references for boundary half-edges should still form a loop or in our case, "-1".
- **Mesh Traversal**: Implementing functions to traverse the mesh (e.g., walking through vertices, edges, faces) and verify that you can visit all elements without encountering errors.
- **Error Handling**: Testing the data structure's response to invalid operations or inconsistent data, and ensure that it raises appropriate errors or handles such cases gracefully.
- **Performance**: For larger meshes, check the performance of your data structure in terms of access time, traversal time, and memory usage.