Assignment 1 (Geometry Processing)

Mesh Smoothing

Library used - **PyMesh**(Geometry processing library for Python)
Scheme 1: Simply computing the average of vertex positions.

Mesh Data Structure:
mesh (Geometry, Connectivity, Attributes)
Geometry: Geometry consists of vertices, faces and generalized voxels
Connectivity: The connectivity contains adjacency information between faces, vertices and voxels.
Attributes: Attributes are arbitrary value field assigned to a mesh.

For every vertex V, it's neighboring vertices list has been traversed. After that for each adjacent vertex U from the neighboring vertices list, the average of vertex position is calculated. At the end of one iteration of vertex V, it's new position has been set by the average vertex position.
Constraints - i) Vertices of boundary edge can not be moved

As PyMesh does not have any pre-defined function to find the boundary edges so I developed following algorithm to find boundary edges-

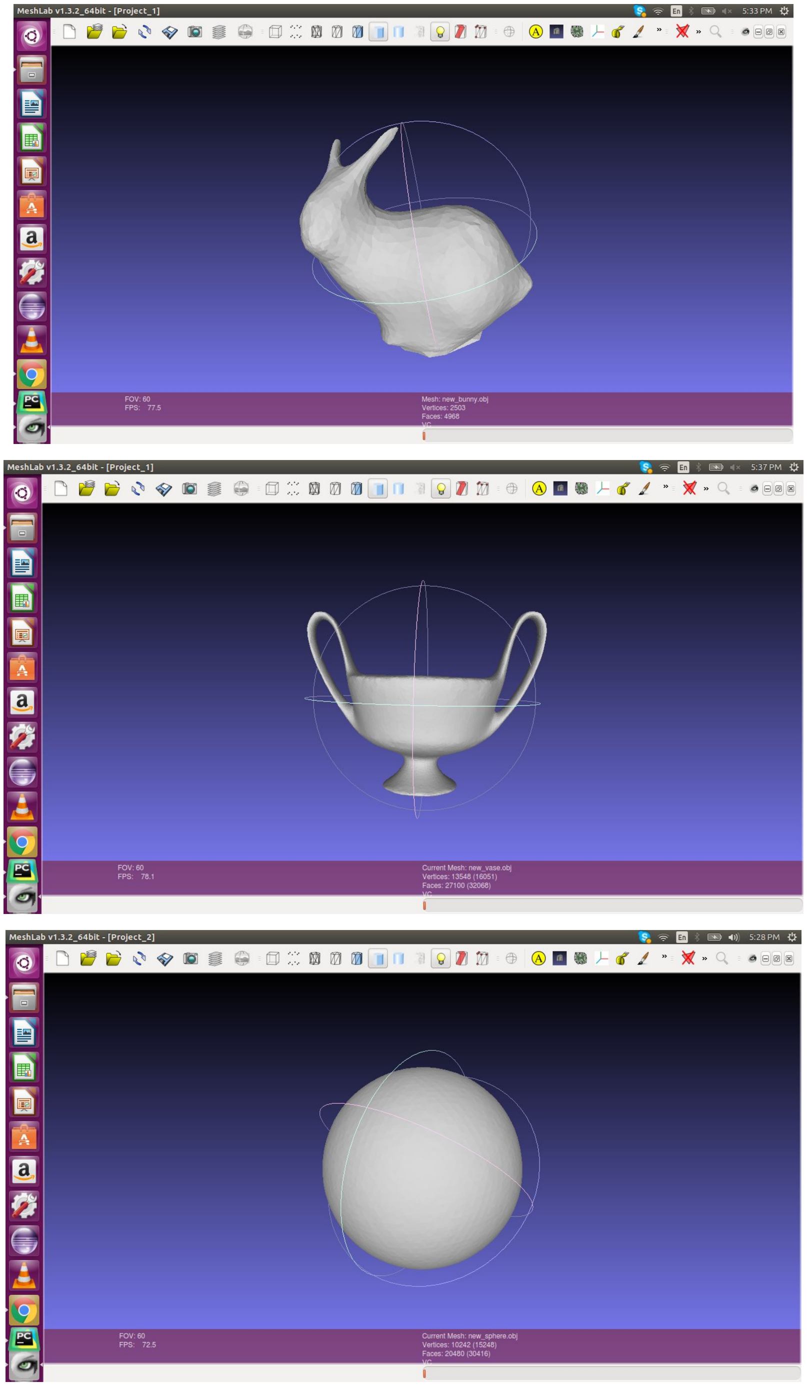
```
for each face(Fi) in the list of Faces
    find adjacent-faces list for that face
    if the adjacent-faces list is less than three
        add the face(Fi) into the list of boundary-faces
for every face(Fi) in the boundary-faces list
    Find it's three adjacent vertices(Vi)
    Calculate the degree of these 3 vertices
    The 2 vertices with lowest degrees among the three form the boundary edge
```

After using the above algorithm, the boundary edges are fixed (means vertices involved with boundary edges does not move) while updating the vertex position.

Disadvantages:

1. For larger iteration (n >=50) the algorithm takes more time as the complexity is $O(n^2)$
2. The mesh smoothes properly for larger iteration(n >= 50) but some regions around the holes becomes very pointy with the increase of iteration

For iteration(n = 15) the following models noisy_bunny.obj and noisy_vase.obj shows the output


MeshLab v1.3.2_64bit - [Project_1]
5:33 PM
MeshLab v1.3.2_64bit - [Project_1]
5:37 PM
MeshLab v1.3.2_64bit - [Project_2]
5:28 PM

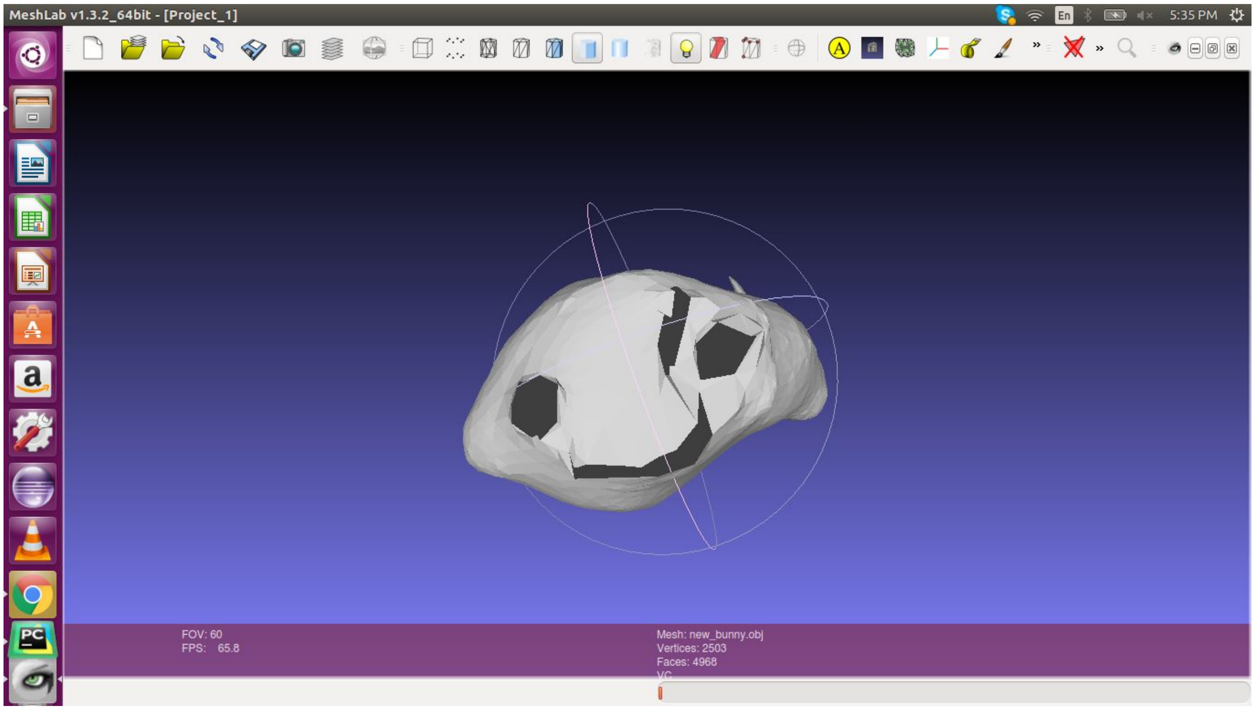
```
FOV: 60
FPS:    77.5

Mesh: new_bunny.obj
Vertices: 2503
Faces: 4968
VC
```

```
FOV: 60
FPS:    78.1

Current Mesh: new_vase.obj
Vertices: 13548 (16051)
Faces: 27100 (32068)
VC
```

```
FOV: 60
FPS:    72.5

Current Mesh: new_sphere.obj
Vertices: 10242 (15248)
Faces: 20480 (30416)
VC
```

Fig: Mesh smoothing

Preserving the holes while smoothing,



```
FOV: 60
FPS:    65.8
```

```
Mesh: new_bunny.obj
Vertices: 2503
Faces: 4968
VC
```

Scheme 2: Weighting each neighboring vertex by the area of the two faces incident to it. Make sure to properly normalize the weighted sum.

For this scheme, the average vertex position is now weighted by the area of the two faces incident to it. The algorithm is following -

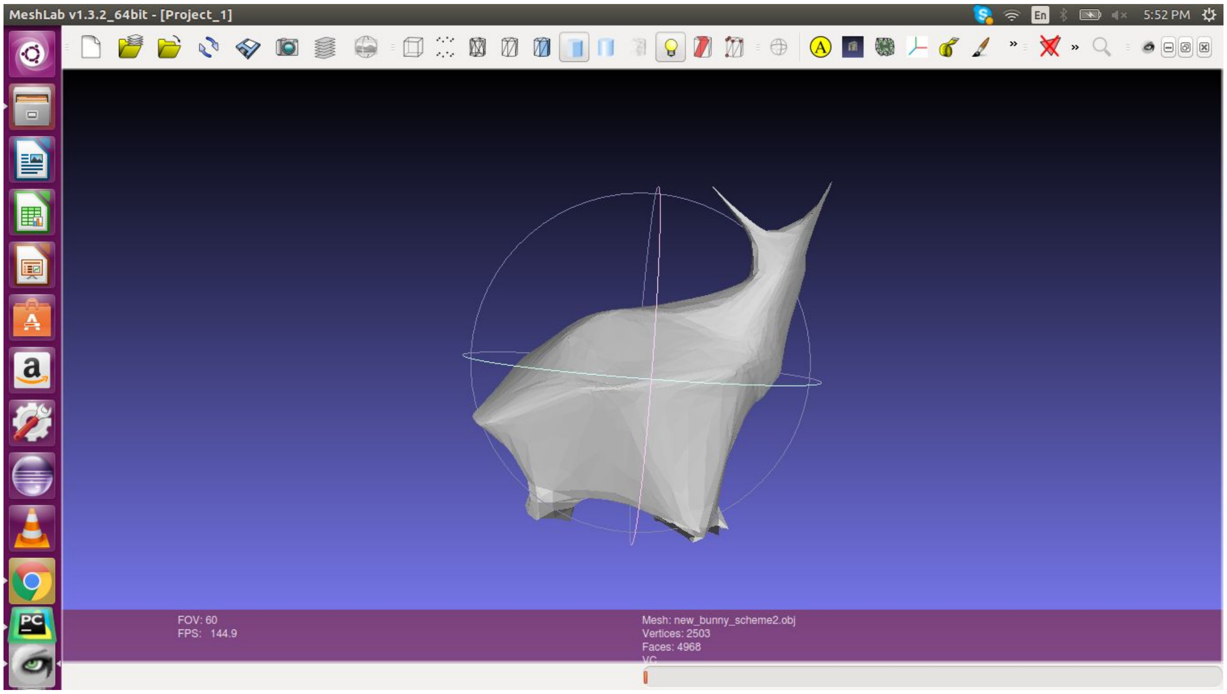
```
For each vertex V
        Find it's adjacent faces (faces_Of_V)
        For each adjacent vertex U of V
                Find it's adjacent faces(faces_Of_U)
                Find the two common faces between faces_Of_V and faces_Of_U
                Calculate the areas as weight of faces_Of_V and faces_Of_U
                Update the neighboring vertex position to get new_position
                         by adding and normalizing the weight
Update the  position of vertex V with new_position
```

Disadvantage:

1. Take much longer time(around 4 mins) than the previous algorithm if the iteration is large (n=50)
2. Regions around the holes becomes very pointy.

3. Though the mesh smoothes properly but as the iteration grows (larger than 60) the mesh starts to deform.

For iteration(n = 60) the following models noisy_bunny.obj starts to deform using the above algorithm -



```
FOV: 60
FPS:   144.9
```

```
Mesh: new_bunny_scheme2.obj
Vertices: 2503
Faces: 4968
VC
```