CS3242 Modeling Lab Assignment Submission

Name: Gabriel Benedict Teo Jian Cheng **Student number:** A0184480B

Task Checklist

Tasks	*Completion Level*	
Computing Normal Vectors (main, 20 marks)	100%	
Compute Angle Statistics (main, 30 marks)	100%	
Write an OBJ file (main, 20 marks)	100%	
Read Some Other Type of Files Other Than OBJ (optional, 20 marks)	100% (STL)	
Implement `enext()`, `sym()` (20 marks, main)	100%	
Implement `org()`, `dest()` (20 marks, main) .	100%	
Implement `fnext()` (80 marks, main)	100%	
Compute the Number of Components (20 marks, optional)	100%	
Implement `orientTriangles()` (20 marks, optional)	0%	
Compute Vertex Normal Vectors for Smooth Shading (10 marks, optional)	100%	
Visualize boundary edges (10 marks, optional)	0%	
Implementing Selection of Triangle by User Marquee (20 marks, optional)	0%	

Final Task

Topic: Removing Self-intersections

Deliverables:

Highlight intersecting triangles
Retriangulate intersecting triangles (using vertex insertion)
Remove hidden triangles (after retriangulating intersecting triangles)
Draw selected triangles (useful to visualize intersecting triangles)

Final Task

Reflection:

For my final task, I was interested about how self-intersections can be automatically removed in 3D meshes. To achieve this, I found an article (see references) that proposed a high-level algorithm and wanted to try implementing it. In order to implement this algorithm, I had to learn the following techniques in 3D graphics:

- Triangle-Triangle intersections (Möller)
- Triangle-Ray intersections (Möller–Trumbore)
- Vertex Insertion (Adapted from Bowyer-Watson)

These techniques were difficult to understand. Therefore, I am very happy with the result of this assignment as my final solution seems to implement them correctly. To demonstrate, please follow these steps:

1. Read in a small mesh with self-intersections
    - "cat.obj"
    - "deer.obj"
    - "demo.obj" (included in .zip)
    - "sphere.obj" (included in .zip)
2. Turn off smooth shading (vertex normals are not accurate after removing self-intersections)
3. Press 'X' to highlight self-Intersections in the mesh
    - Press 'T' to draw a subset of triangles. This is useful to see self-intersecting cases.
    - Press 'D' to toggle the face culling. Some intersections may occur beneath the surface of the model.
4. Press 'C' to retriangulate triangles with self-intersections
5. Press 'Z' to remove hidden faces.

Note to Prof Alan:

I removed the duplicate model in the code so that there is no more issue when loading in a huge mesh like "pikachu.obj". After our presentation, I realized there was no reason to since it is easier to just read in the model again.

References

Zhu, Jiang & Hosaka, Yurio & Yoshioka, Hayato. (2019). A Robust Algorithm to Remove the Self-intersection of 3D Mesh Data without Changing the Original Shape. Journal of Physics: Conference Series. 1314. 012149. 10.1088/1742-6596/1314/1/012149.