Fall 2015

Report for Advance Algorithm Programming Assignment 1

Youngeun Lee

1 Implementation Details

The goal of this assignment is to compute Delaunay triangulation. I changed only Delaunay function in delanay.c. It has 3 steps as follow:

1. Create points in 4D $(x, y, z, x^2 + y^2 + z^2)$
2. Compute convex hull in 4D by qhull library[1]
3. Project the lower convex hull into 3D

I will explain each steps in detail.

The first step is very easy. The vertices of the input file are stored in vertices. I made a pointer (pt) of $coordT$ type and inserted the vertices and 4th coordinate.

The program calls qhull to compute convex hull in the second step. I referred to $QHull()$ function in main.c file. This function compute convex hull in 3D but the program should create convex hull in 4D. The option for qhull is $delaunayQJPp$. The pointer pt is delivered to $qhullbyqh_init_B()$ function. The 3th component of $qh\_init\_B()$ is 4 instead of 3 because I need to compute the convex hull in 4D.

The third step is difficult and I spent most of time in the step. The last coordinate of a normal from the convex hull determines that its tetrahedron is in upper side or lower side. The upper convex hull should look down the 4th axis. So the 4th normal value of the lower convex hull is less than zero. But it is not a enough condition for triangulation. A tetrahedron can be projected as a plane (Fig. 1) since it lose one-dimension. These coplanar points occur crossed edges and faces. The program computes the volume of tetrahedron to avoid duplicated faces. $Volumei$ function in shape.c file computes a volume of a tetrahedron. The function returns 0 when the points of tetrahedron is coplanar. A tetrahedron which has 0 volume is not used to face construction.

I used Visual Studio 2012. I set up input argument $\%s\ d../../data/filename$ in the project property (Fig. 2).

2 Example Output

Fig. 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14 show results of each test file. The results of triangulation are not perfectly matching with given points because the program projects convex hulls. As you can see, there is no crossed faces and edges.

[1] www.qhull.org/

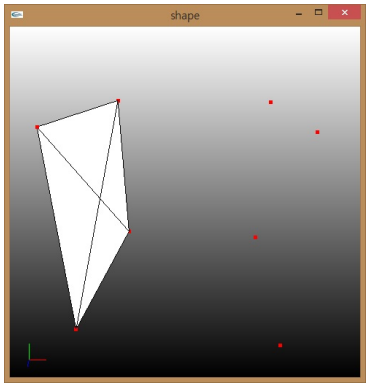


Figure 1: Coplanar Tetrahedron. The original model is *i.cube*. The 4 points are coplanar points, though they are components of a tetrahedron in 4D.

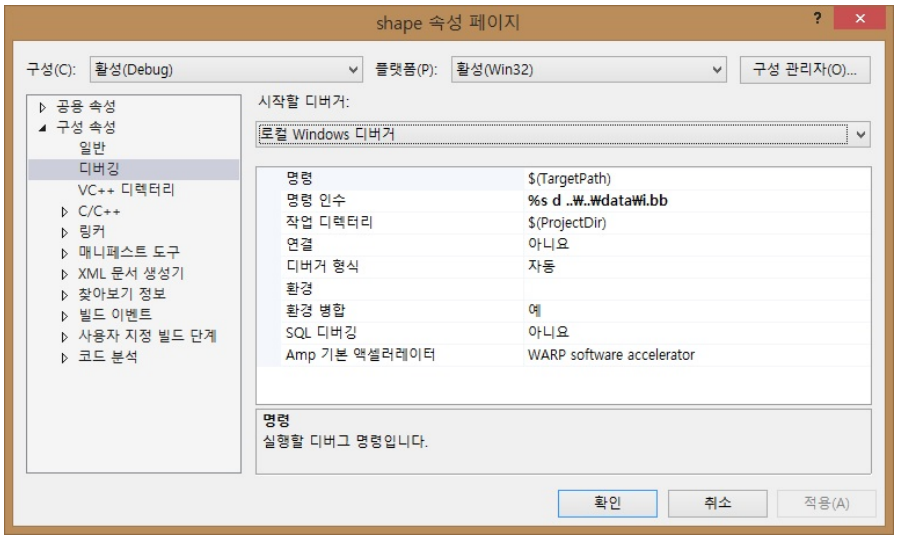
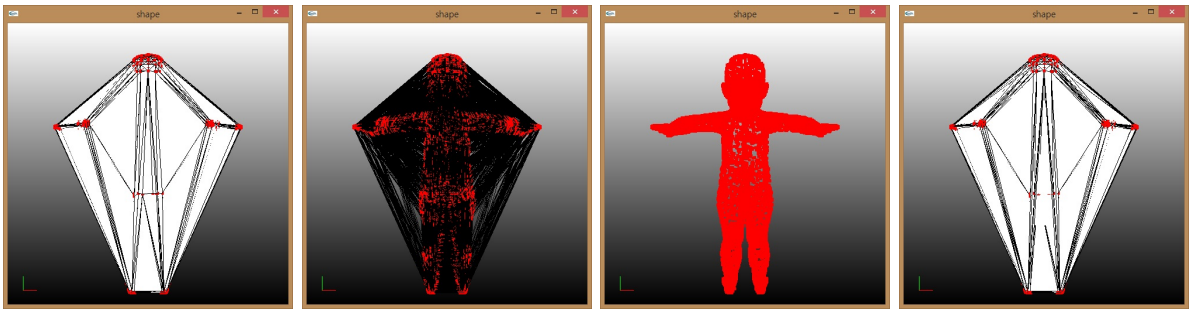
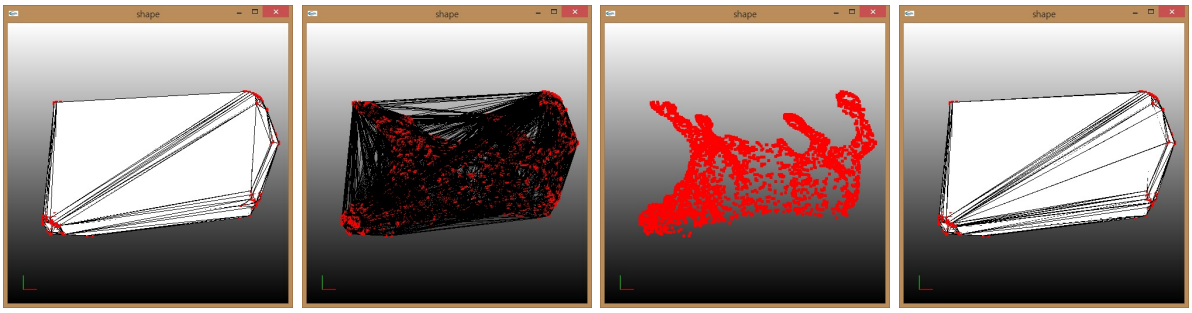


Figure 2: Input argument.



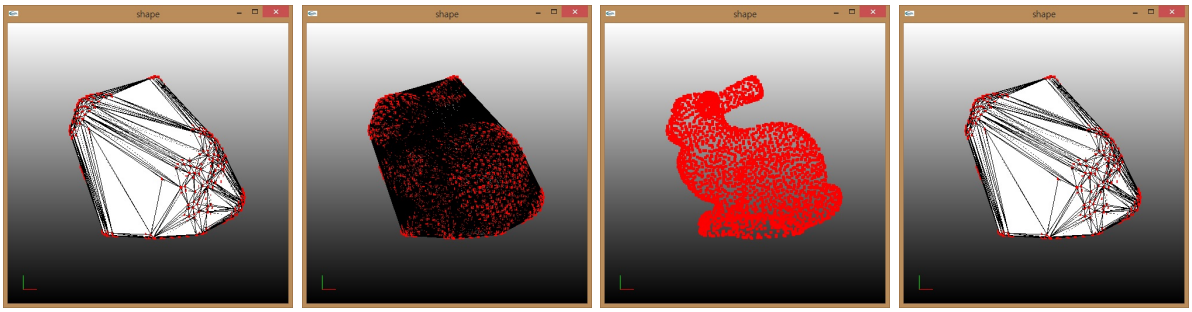
(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 3: i.bb



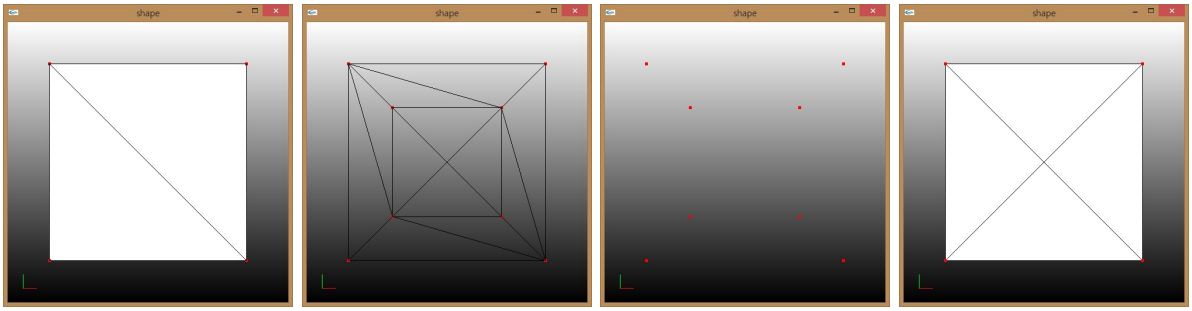
(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 4: i.bull



(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 5: i.bunny

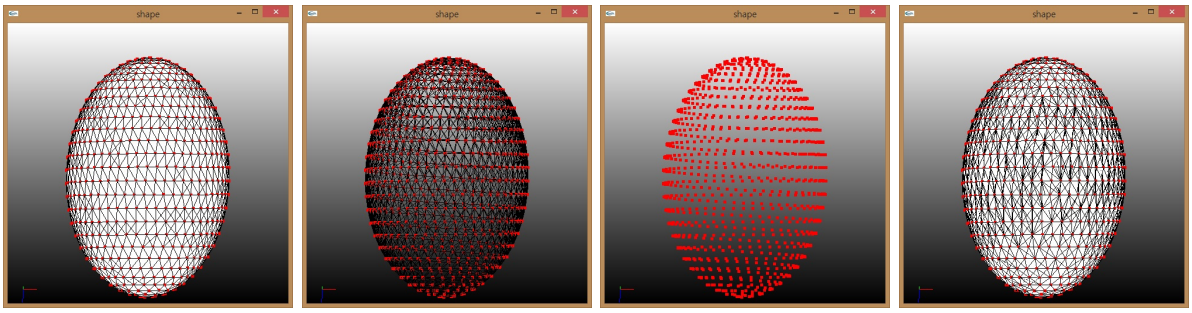


(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 6: i.cube

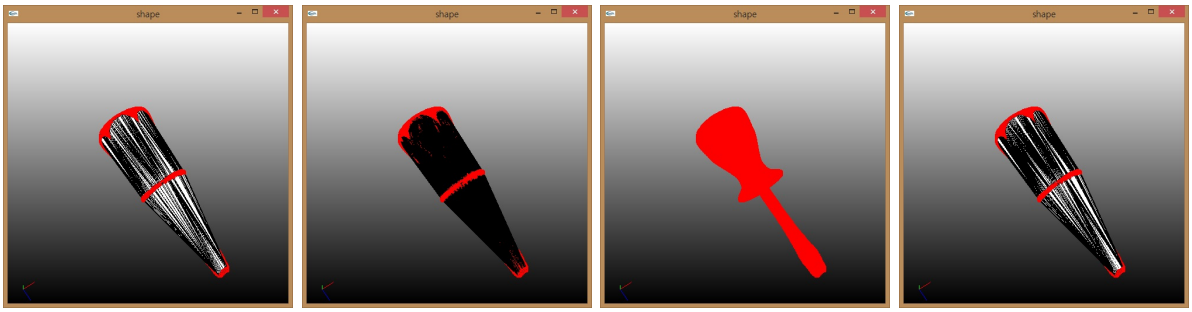
3 Know bugs/limitations

The program does not check duplication of edges. A edge exists as many as the adjacent faces in *edges* since the program inserts all edges according to faces. Also $face$ in *tetras*, *edge* in $faces$ and $adjface$ in *edges* are not implemented but they does not influence on the results. The results are not close to the original model. It may approximate concave shapes to convex shapes such as ears of the bunny (Fig. 5). However, it shows perfect matching for convex models as illustrated in Fig. 6 and 7. The program requires more method such as alpha shape to remove extra convex



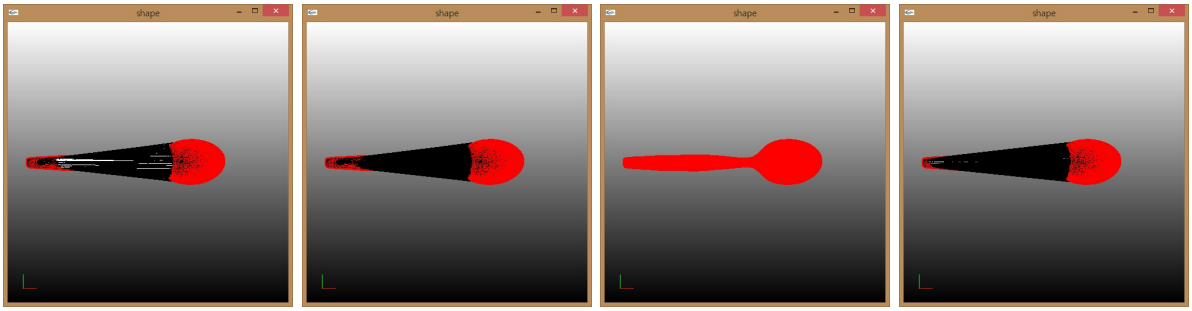
(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 7: i.ellipsoid



(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

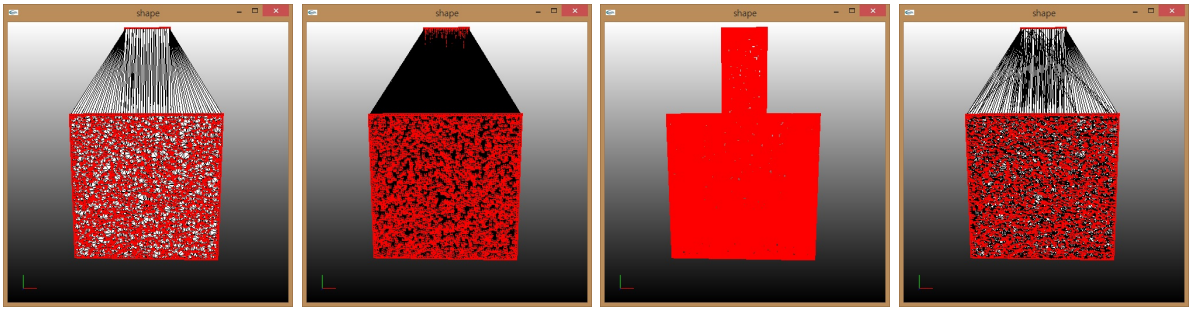
Figure 8: i.screwdriver



(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

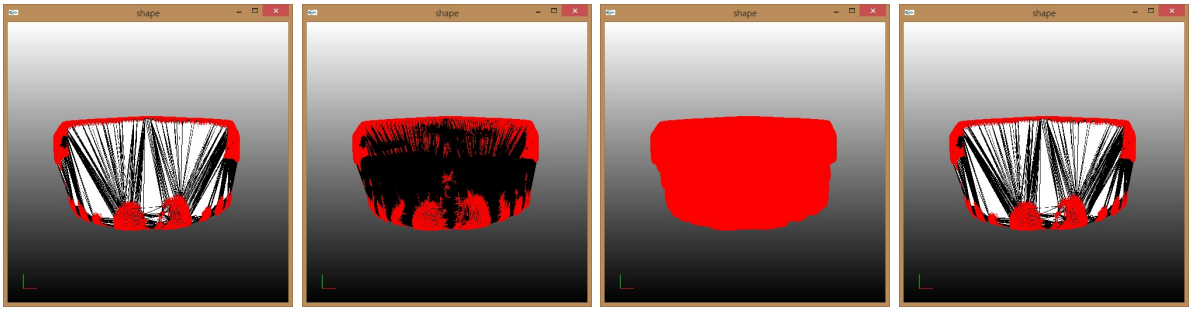
Figure 9: i.spoon

areas.



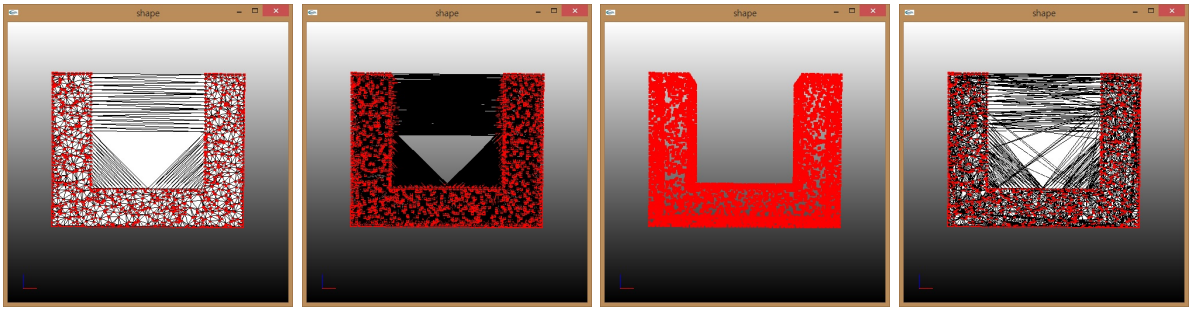
(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 10: i.T



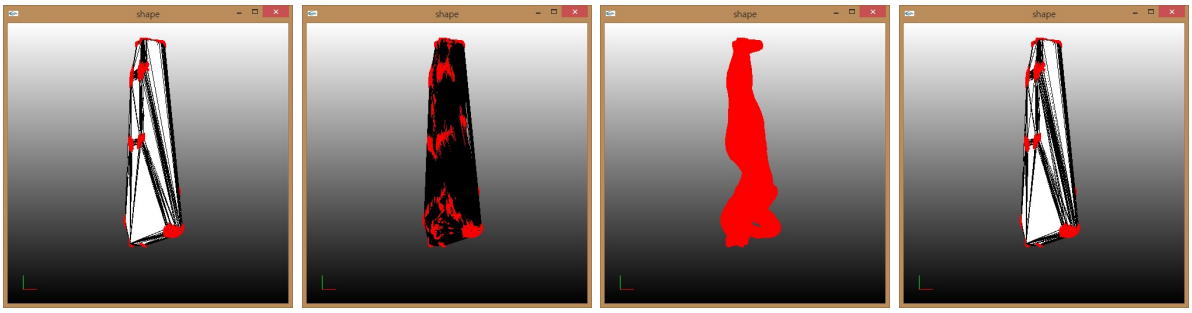
(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 11: i.teeth



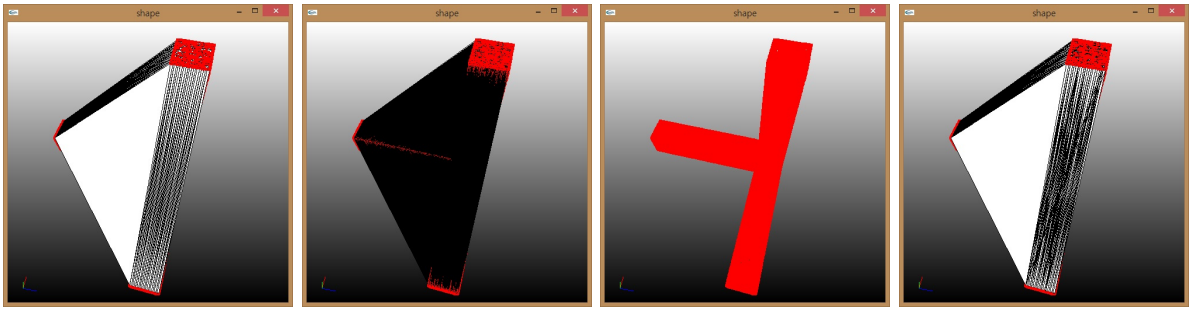
(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 12: i.U



(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 13: i.woman



(a) Faces and vertices (b) Edges and vertices (c) Vertices (d) Tetrahedrons

Figure 14: i.Y