Mesh Merging

UPPSALA UNIVERSITET

Author:
Fredrik Viström
frvi.4047@student.uu.se

Supervisors:
Nader Salman, Schlumberger
Bjarte Dysvik, Schlumberger
Maya Neytcheva, Uppsala University

Department of Information Technology

Box 337

SE-751 05 Uppsala

Sweden

Introduction

A surface mesh can be used to enclose a volume. If two volumes, enclosed by two meshes, overlaps a mesh for the combined volume can be desired to find. This mesh is found by merging the two meshes.

Method

The algorithm for merging two meshes can shortly be described as:

- Find the intersecting triangles from the two meshes.
- Extract a boundary for each mesh from the intersecting triangles.
- Connect the two boundaries by triangles.
- Assemble the final mesh.

With this approach there are two crucial points, the speed for finding the intersecting triangles and how the two boundaries are connected. The execution time for merging two meshes are almost exclusively determined by the time it takes two find the intersecting triangles.

The naïve way for finding intersecting triangles from mesh A is to test wither all the triangles one by one against all the triangles in mesh B. To reduce the number of triangle-triangle-intersection tests some geometrical properties are used. Among other things a center point and a radius for each mesh are computed, where the radius is defined as the largest distance from any vertex in the mesh to the center point of that mesh. If a triangle from mesh A lies further away from the center point of mesh B than the radius of mesh B, then this triangle from mesh A can not intersect with any triangle from mesh B.

The only new triangles created are the ones generated when connecting the two boundaries. The rest of the triangles are taken from the two meshes that are to be merged. This means that the quality of the mesh is only dependent on the previous meshes and the connecting triangles. Hence the importance of this step.

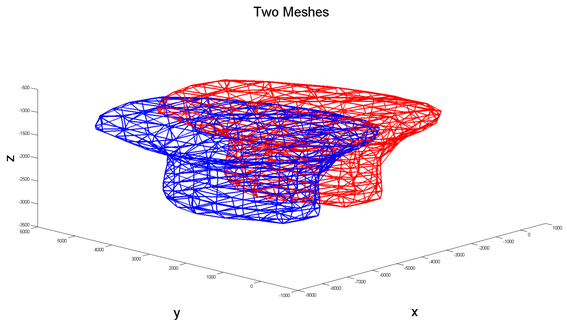

Figure 1: *The two meshes that are merged together, one mesh shown in red and one in blue.*

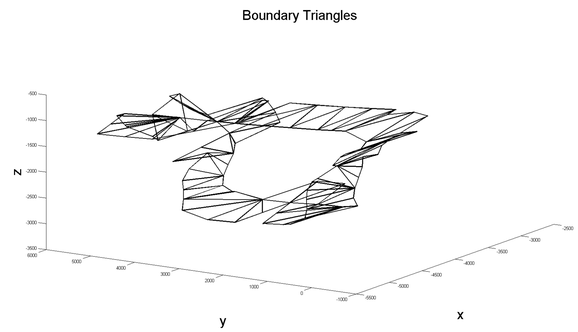

Figure 2: *The triangles connecting the two boundaries.*

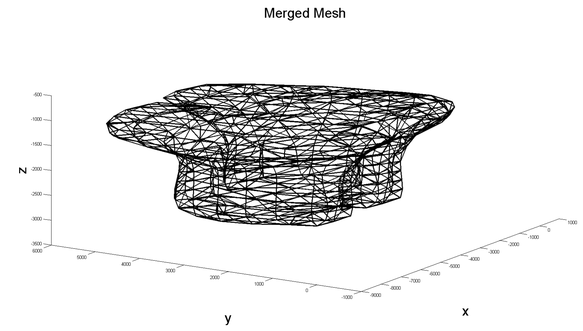

Figure 3: *The resulting, merged mesh.*

If the two boundaries are of different length the short one is interpolated. In order for the final mesh to be a correct surface mesh two vertices lying next to each other on the boundary has to take part in a triangle together. When connecting these boundaries one boundary is chosen as a base (A). Then the triangles are created by finding the vertex at boundary B that makes the best triangle with the two vertices from boundary A. This is performed for all the pairs of vertices on boundary A, the rest of the triangles are found by filling the holes not yet covered by triangles.

Result and Discussion

If the two meshes shown in Figure 1 are merged the resulting mesh will be the one presented in Figure 3. The connecting triangles between the two boundaries are shown in Figure 2. The execution time for this merge is shown in Table 1, with execution time for an other merge as well. For the two merges the meshes have the same shape but the resolution of the meshes differ.

Since the shape of the triangles in Figure 2 looks good the quality of the merge is considered to be fine. However, it would be desirable with a faster execution, since the algorithm has to be able to handle meshes with many more triangles. One solution for this is to parallelize the algorithm, preferably on GPU. The algorithm also has to be generalized in order to handle cases when the intersection between the two meshes are more complex, leading to several of boundaries for each mesh.

Table 1: *Execution times for two merges.*

Number of triangles (mesh A × mesh B)	Time for finding intersecting triangles (ms)	Tota time for execution (ms)
764 × 760	236	246
19 580 × 19 568	129 729	129 863