Documentation for the Mesh Processing Processes for Vorticity Project 18 April 2017

# Mesh STL Output from Paraview

*Paraview (Mainly to be written by JR)*

After running the various CFD experiments, we exported 3D geometry from the fluid dynamics program, Paraview. Paraview is a voxel-based 3D physics program, and doesn’t work natively on mesh geometry, however the final geometry for the game needed to be a mesh to import to Unity. This required converting all the time slices of the experiment to mesh, and thankfully Paraview has an in-built STL exporter. We used a resolution of XXXX when exporting to STL, yielding very large files. For Sim 0 each vector of each time slice averaged 72.36MB per file and Sim 1 was smaller at 28.83MB per file. For importing the meshes to Unity for the final game assets, they would need to be both reduced in complexity, and stitched together the filaments at the 2 pi edges.

**Sim 0** **Sim 1** **Sim 2**

Pre Post Pre Post Pre Post

**Total** 30.39 GB 3.38 GB 12.11 GB 1.27 GB ? ?

**Per File** 72.36 MB 8.05 MB 28.83 MB 3.03 MB ? ?

# Mesh decimation for reducing file size and complexity

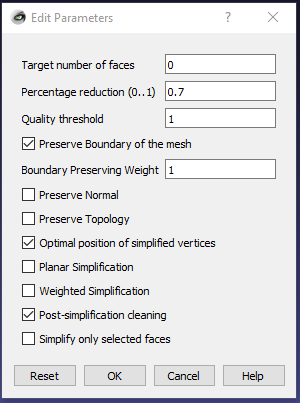
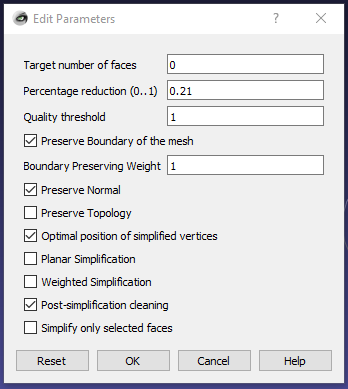
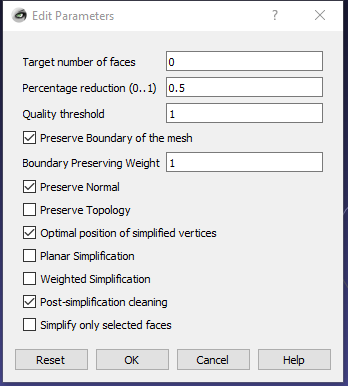
*Meshlab*

**Quadric Edge Collapse Decimation**

For ease of computation we reduced the face count and resulting file size first, before the geometric operations. We used Meshlab, an open source mesh processing program to reduce the files with a Simplification filter, Quadric Edge Collapse Decimation. This function simplifies a mesh while trying to retain the original geometry as much as possible.

Our goal for the final mesh size was a significant reduction, about 90% reduction on file size while maintaining a high level of tolerance to the original. Using a Hausdorf difference, we decided 88% was as much as we could reduce the original mesh while staying within whatever tolerance Janet wanted. Memory limits for the Meshlab program required three passes of thee decimation algorithm at incremental steps, instead of reducing 88% all at once, we reduced compounding 30%, then 50%, then 79%, to arrive at a final size of 11.85% of the original size. After reduction the meshes were saved out again as STLs.

Pass 1 Pass 2 Pass 3



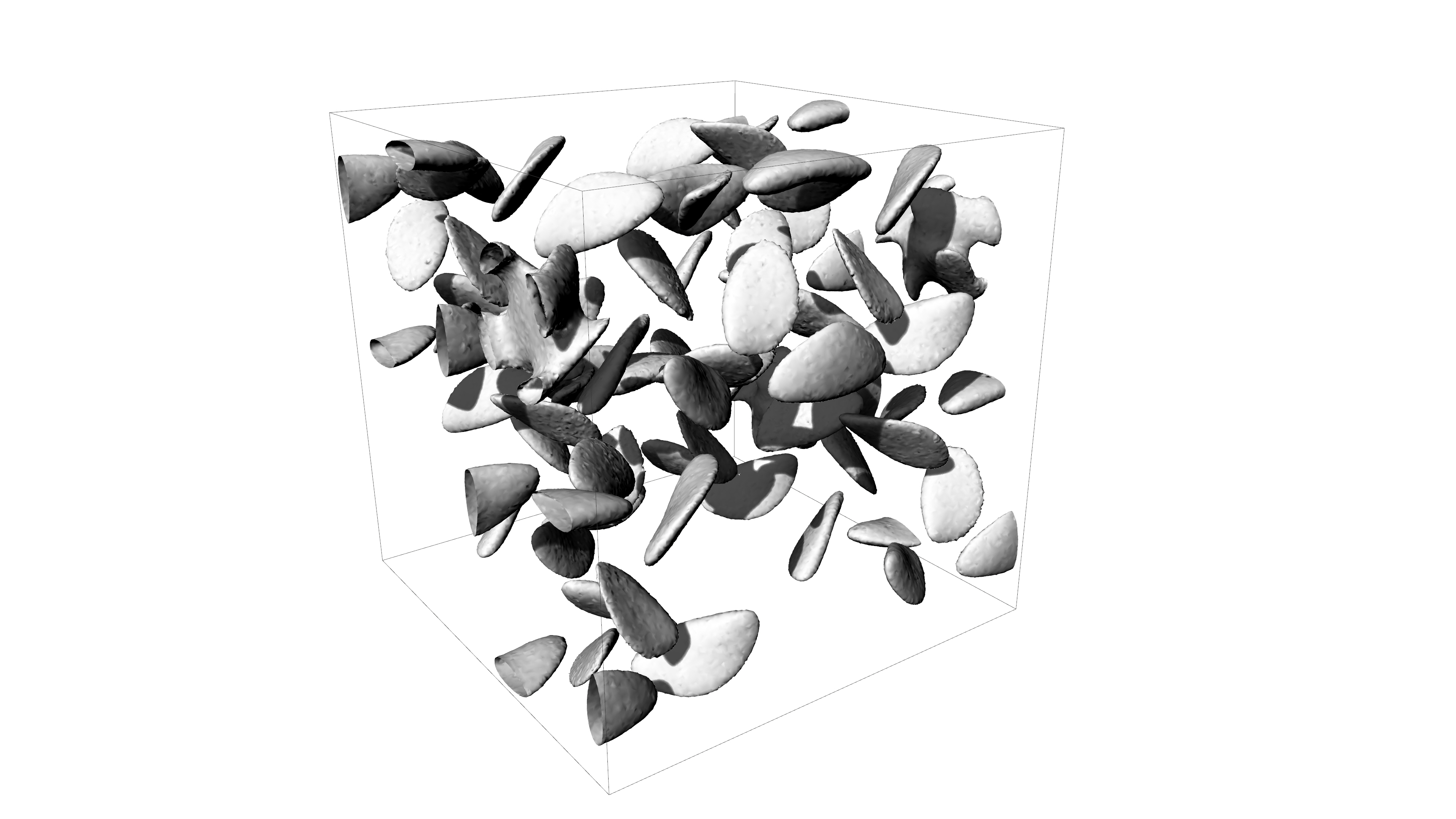
Janet’s computation on the Haussdorf distance here.

I sampled a random timeslice, reduced, and Hausdorff, and it said this. Maybe it means something to you?

Hausdorff Distance computed  
Sampled 3942830 pts (rng: 0) on 0\_42\_2.stl searched closest on 0\_42\_2\_small\_1185.stl  
min : 0.000000 max 0.383292 mean : 0.001130 RMS : 0.002204  
Values w.r.t. BBox Diag (10.747447)  
min : 0.000000 max 0.035664 mean : 0.000105 RMS : 0.000205

# Grasshopper / Python

Stitching meshes into solid objects



**Importing and Tolerance**

Once the STLs were reduced in Meshlab, it was much easier to perform the welding operations. As more flexibility was required for manipulating and welding, Rhino and its Grasshopper plugin were used for this step. The files required two main operations – moving open meshes cut open at the positive and negative planes of pi (e.g. ( 0, 0, 3.14159…)) from one plane to its negative plane, and then welding separate cut meshes into one. (Picture needed) All discrete meshes were exported as individual FBX files, an Autodesk mesh format popular for game development.

For the Rhino file, the absolute unit tolerance was 0.0001 units.

**Moving and Welding Together**

For game play experience (and maybe because it didn’t matter for the physics?) we culled imported meshes with less than 10 faces. Most were easily in the thousands or 10,000s of faces. Once import, the meshes that intersected or were cut by the planes at PI in any positive direction, were moved and welded to they’re reciprocal planes in the negative direction. Once moved, the welding tolerance was 0.01. This means that vertices within 0.01 units of each other get combined and two meshes become one.

**Exporting to FBX**

Once the welding was complete, each discrete mesh with a face count over 100 was exported to FBX as a separate file.

# Citations

General Paraview citation

Ahrens, James, Geveci, Berk, Law, Charles, ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook, Elsevier, 2005, ISBN-13: 978-0123875822

General Meshlab citation

*P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia*   
**MeshLab: an Open-Source Mesh Processing Tool**   
Sixth Eurographics Italian Chapter Conference, page 129-136, 2008

Calcing Haussdorff distance between meshes

*P. Cignoni, C. Rocchini, R. Scopigno*   
**Metro: measuring error on simplified surfaces**   
Computer Graphics Forum 17 (2), 167-174, 1998

Automating Meshlab credit

http://www.andrewhazelden.com/blog/2012/04/automate-your-meshlab-workflow-with-mlx-filter-scripts/