

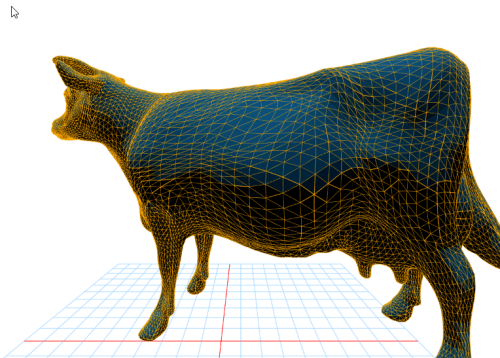
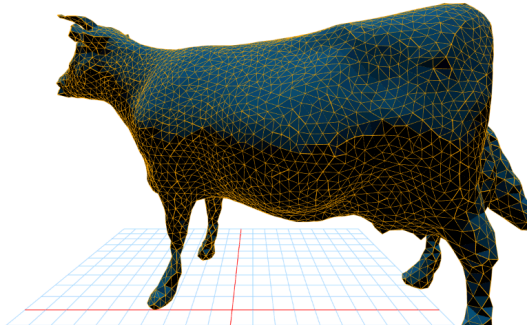
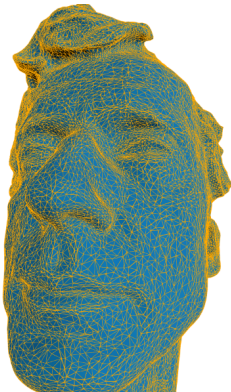
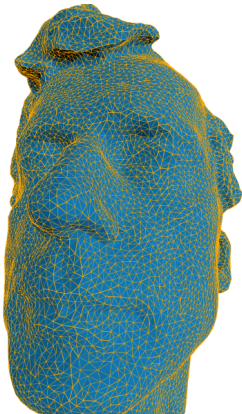
Mesh (final submission)

Please fill this out and submit your work to Gradescope by the deadline.

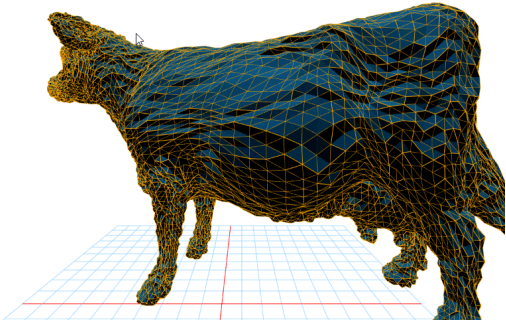
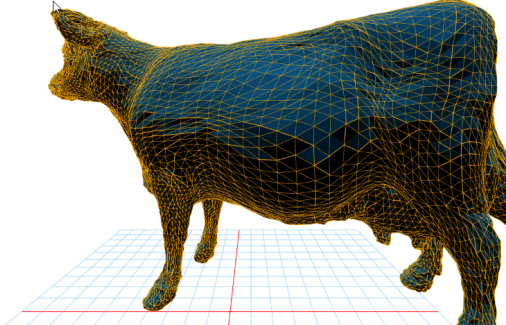
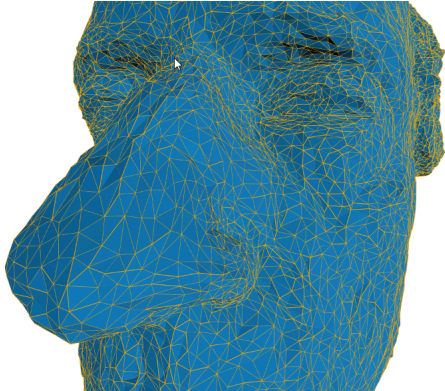
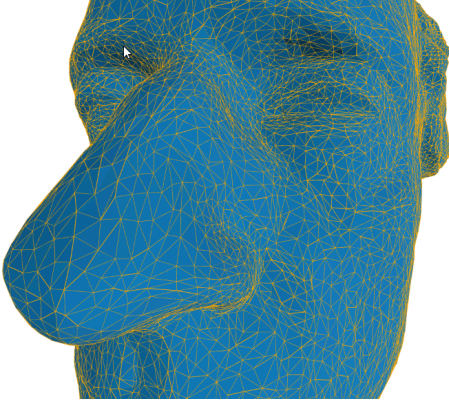
Output Comparison

Run the program with the specified `.ini` config file to compare your output against the reference images. The program should automatically save the output mesh to the `student_outputs/final` folder. Please take a screenshot of the output mesh and place the image in the table below. Do so by placing the screenshot `.png` in the `student_outputs/final` folder and inserting the path in the table.



- For instance, after running the program with the `subdivide_icosahedron_4.ini` config file, go to and open `student_outputs/final/subdivide_icosahedron_4.obj`. Take a screenshot of the mesh and place the screenshot in the first row of the first table in the column titled `Your Output`.
- The markdown for the row should look something like
`| subdivide_icosahedron_4.ini | |

.ini File To Produce Output	Input Mesh .png	Remeshed Mesh .png
/template_inis /final / remesh_cow_fine.ini		
/template_inis /final / remesh_peter.ini		

Output for Bilateral Mesh Denoising (Note: if you did not implement this you can just skip this part)

.ini File To Produce Output	Noisy Mesh .png	Denoised Mesh .png
<pre> /template_inis /final /filter_cow_noisy.ini </pre>		
<pre> /template_inis /final /filter_peter_noisy.ini </pre>		

Output for any other Geometry Processing Functions (Note: if you did not implement this you can just skip this part)

.ini File To Produce Output	Input	Output
<Path to your .ini file>	 Place screenshot input mesh here	 Place screenshot of output mesh here

Design Choices

Mesh Data Structure

Describe your mesh data structure here.

Mesh Validator

Describe what your mesh validator checks for here. This can be a list.

I implemented the halfedge datastructure to store my meshes. My mesh validator function checks for:

1. Every halfedge has a twin, and the twin's twin is the halfedge itself.
2. Every halfedge has a pointer to the vertex it is leaving.
3. Every vertex is attached to a valid halfedge.
4. Every halfedge has a pointer to the face it belongs to.
5. Every face has a pointer to the halfedge it belongs to.
6. Every triangle has three halfedges that point into each other (where you can get back to the start).
7. Double check that we have the right number of halfedges (h) to faces (f) where 3*h must equal f (because each face has its own three halfedges).
8. For every pair of triangles joined by two halfedges, we can walk around the "border" of these two triangles with next and twin pointers to check they are set right.
9. When we delete vertices, make sure they are completely deleted with no pointers pointing to them anymore (i.e. vertices without something referencing are not allowed).
10. When we delete faces, make sure they are completely deleted with no pointers pointing to them anymore (i.e. every face must have three halfedges pointing to it).

11. Double check that all the flags the vertex/halfedge data structures contain (which assist traversal / algorithms) are reset so that different mesh operations do not get messed up by ill-set flags.

Run Time/Efficiency

Describe how you achieved efficient asymptotic running times for your geometry processing functions, including the data structures you used.

Collaboration/References

Known Bugs