Mesh Simplification via Embedded Tree Collapsing Implemented in CUDA (Milestone)

Nicolas Mignucci, George Ralph

URL:

https://github.com/gdr22/15-418-Final-Project

Progress:

Currently, we have written code to create a half edge mesh from a Stanford format (.ply) mesh and store this information in a human readable file. We are also able to load the half-edge mesh from this file and onto the GPU in the format specified by the Lee-Kyung paper. We have also implemented the first step of the algorithm, which requires computation of plane equations over all triangles, followed by computation of the quartic error matrix coefficients for all vertices. This code tests our halfedge implementation. By observation of the code written so far, which makes extensive use of gather operations due to the halfedge mesh structure, our final program is likely to be memory bound.

Goals and Deliverables

Our goals for the milestone date were the following:

- Load meshes as halfedge meshes from a file and onto the GPU in the format specified by the paper
- Compute and return Quadric error at every vertex using the GPU
- Set up harness code to collect processing time results from Open3D's SimplifyQuadricDecimation()

As described above, we completed the first two goals (loading half-edge meshes onto the GPU and computing the quadric error matrix for each vertex). We have not yet set up the harness code to collect the processing time of our reference (Open3D) mesh simplification function. Due to version conflicts regarding the C development tools on the GHC servers, we were unable to get this code to compile on the machines we intended to use to test. We are currently investigating suitable alternatives including but not limited to, compiling the program locally and running it on andrew, and compiling and testing both implementations locally on one of our machines.

The last objective should not take very long to resolve meaning we are so far following our schedule fairly closely. Because we gave ourselves three days of slack with respect to our deliverables, we will still be able to complete them all including at least some of our "nice to have" features.

Poster Session

We plan to present the speedup we have measured between our implementation of mesh simplification to the sequential implementation found in Open3D. As described in the project proposal, we are hoping to at least achieve 10x speedup to match the Lee-Kyung paper's results compared to a competitive GPU algorithm.

We remain unaware of any methods to quantitatively demonstrate the quality of our results, however we do intend to demonstrate our algorithm's output qualitatively. We plan on including renders of the meshes produced by our algorithm on a few test examples at a few different reduction levels, as well as the results of our reference implementation to compare.

Issues and Unknowns

Our primary unknown at this point is which machine we will use to run our implementation and the reference implementation to get consistent runtime results. While the reference implementation does not compile on the GHC machines, our implementation did not compile locally for us in WSL due to compiler version conflicts. These can likely be resolved with more time to investigate, though currently, we have not given this much of our time.

Schedule

- 11/23/21
    - Patch write issue with vertex quadric error matrices (George)
    - Verify correctness of quadric error computations (George)
    - Investigate compiling Open3D on andrew, or CUDA code locally (Nicolas)
- 11/26/21
    - Construct embedded trees of collapsable edges (paper step 2) (Nicolas)
    - Adjust halfedge mesh structure to include boundary edges (George)
- 11/29/21
    - Collapse trees to root vertices (paper step 5) (George)
    - Update vertex positions using the conjugate-gradient method (paper step 6) (Nicolas)
    - At this point, we should have a functional mesh reduction algorithm, although the results it returns may have some errors
- 11/30/21
    - Implement a halfedge to .ply conversion tool to view output meshes (George)
- 12/3/21
    - Add tree splitting to satisfy the decimation quality constraint (paper step 3) (George)
    - Split trees further to remove illegal mesh transformations such as normal flipping or topological changes (paper step 4) (Nicolas)

    - Both of these steps can be done optionally if we remain on schedule up to this point (stretch goals) as they are merely filter operations meant to improve the quality of the output mesh
- 12/6/21
    - Collect performance data for our implementation and reference implementation (Nicolas)
    - Plot and comment on measured speedup (Nicolas)
    - Collect and visualize model quality of our implementation and reference implementation on various inputs (George)
    - Visualize and comment on quality between the two implementations (George)
- 12/9/21 (Final Project Report)