Point Cloud to Simplified Mesh

Team JAM
Jasper O'Leary
Annalise Hurst
Matthew Waliman

In this project we plan to implement a two-stage pipeline consisting of: transforming a point cloud to a mesh using the *ball-pivoting algorithm* described in Bernardini et al., and then simplifying the resulting mesh using *quadric error metrics* described in Garland and Heckbert.

Background
We seek to facilitate the transformation from raw 3D scans into easy-to-use meshes. When a 3D scanner scans an object, it does not directly create a mesh that we can manipulate, but instead creates a cloud of points sampled from the object. In order transform this point cloud into a mesh, one must first interpolate the points with edges using a mesh generation algorithm. For practical purposes, we should simplify the resulting mesh because using the raw points as vertices in the mesh pegs the complexity of the mesh to the resolution we use for the 3D scanner.

Going from a set of points with essentially any topology to neatly constructed mesh is a challenging problem. For example, Prof. Jonathan Shewchuk here at UC Berkeley teaches an entire course on techniques for generating meshes. Additionally, simplifying the topography of meshes is difficult because we have to balance several factors, for example: running time, simplifying while preserving the fidelity of the object, and consistency across the topography.

While there are many such algorithms, for this project we focus on the ball-pivoting algorithm proposed by Bernardini et al. for the mesh generation and the quadric error-based mesh simplification algorithm proposed by Garland and Heckbert. We chose these algorithms for the pipeline because they are both: a) reasonable to implement in two weeks only, and b) interesting algorithmically in how they function. The ball-rolling algorithm is especially cool. Note that the mesh simplification via quadric error was also touched upon briefly in lecture.

This pipeline is relevant because it allows to go from real-life object into a robust computer mesh.

Resources
Stage 1: Point Cloud to Mesh
This is currently the least well-defined portion of the pipeline and could use a lot of input here. We plan on using .PLY files, processing them and generating the mesh points in

python and then outputting a .dae file from there. A point cloud viewer could be helpful as well but we've had difficulty finding ones that accept the Stanford files.

We will use the `meshedit` software from Assignment 2 as a viewer for point clouds in order to visualize our process. This will important for debugging. We will be implementing the functionality to view point clouds in `meshedit`. This will allow us the keep the entire pipeline (stage 1 and stage 2) in one editor, which we will call `meshedit++.`

Stage 2: Mesh Simplification
We will also be using the `meshedit` software to implement our mesh simplification.

Goals and Deliverables
What we plan to deliver:
We plan to deliver an ***interactive point cloud-to-mesh pipeline*** implementing both the ball-rolling algorithm and quadric error-based mesh simplification. In the end the user will be able to: a) open a point cloud in `meshedit++`[1], b) convert the point cloud into a mesh in `meshedit++`, and c) simplify the mesh.

At the moment, we plan to gauge the accuracy of our system by comparing our generated meshes against ground truth meshes. We will use the Stanford point cloud repository's point clouds to generate meshes that we will compare against their reference meshes. `CBbunny.dae` for example, corresponds with a `.PLY` file on the repository. While we plan to compare our generated meshes with ground truth meshes by visual comparison alone, we may also create error metrics if we find that eyesight alone does not suffice for comparison.

What we hope to deliver:
If we complete our planned deliverables we will try to implement remeshing to create similar areas for all of the triangles in the mesh. We would also like to implement a visualization of the ball rolling during mesh generation, but this depends not only on time, but also on how we decide to set up our resources for the point cloud-to-mesh stage.

Schedule
April 18th - Get input on our setup for resources.
April 20th - Finalize setup for viewing point, and have approximating errors with quadrics completed.
April 25th - Implement ball pivoting, and complete vertex decimation and vertex clustering.
April 27th - Implement join and glue operations, and complete iterative edge contraction.
May 2nd - Finalize details on multiple passes, and complete algorithm implementation of quadrics with the surface simplification techniques.
May 4th- Final Presentations.

[1] Or potentially with the help of an additional resource, see Resources, Stage 1 above