Summary of *A Simple, Fast, and Effective Polygon Reduction Algorithm*

Victor Shu

The article talks about an approach to simplifying meshes and its implementation details. There could be many situations that one may need to simplify a 3D mesh, which includes fitting different hardware specifications on a web application or achieving some special effects in a game. The author used this technique to eliminate small triangles that do not contribute to the visual quality in a game prototype.

The approach introduced by the article is called Collapsing Edges, where one of the selected two vertices moves or collapses into the other one, which leads to the removal of one vertex, two faces, and three edges. The "cost" of removing an edge is decided by its length multiplied by a curvature term, which is then determined by the dot product of the two face normals adjacent to the edge so that the small details are removed first thus preserves the features of the mesh.

While a typical data structure for storing polygons uses a list of vertices and a list of triangles that contains the vertices, the article extends the data structure so that it has access to which vertices the triangle uses and which triangles that a vertex bounds. The article also caches the face normals since the edge selection frequently uses them. The implementation also stores the removed vertex data so that when required, the algorithm can retrieve the data and construct the mesh without having to recompute.

There are other alternative techniques to edge collapse, with the most obvious one of making artists make different versions of the model, which can preserve the mesh quality the best. There is also an approach of simplifying meshes, which is to represent the geometry with parametric surface patches and tessellate them during run-time. While the shape can be preserved as much as possible, it is not easy to generate polygons without gaps or T-intersections at render time, and the data required would be more than the number of polygons.