Project1: Mesh Subdivision & Simplification

Chunxu Xu

Contents

1 Required Tasks

This project requires to implement and analyse the methods of mesh subdivision and simplification. The following is a minimal requirement:

- Implement the Loop subdivision on triangular mesh;
- Implement the quadratic error metric simplification on triangular mesh;
- Analyse the implemented methods in various way.

Implementing other schemes on subdivision and simplification is optional. As this is a well-studied field in computer graphics and geometry, there are a lot of references describing different methods each of which has their own feasible applications. Thus in the end of this document, we list some representative references for you to go for the methods' mathematica principles and implementation details.

On the other hand, we provide some extra materials besides the document:

- A toolkit for 3d geometry operations and mesh organizing. It is provided in the form of cpp source code, which you can modify yourself for the convenience of implementation. You can also choose to construct the data structure of the mesh from scratch.
- Some 3d testing models. You can also use other models to verify your implementation.

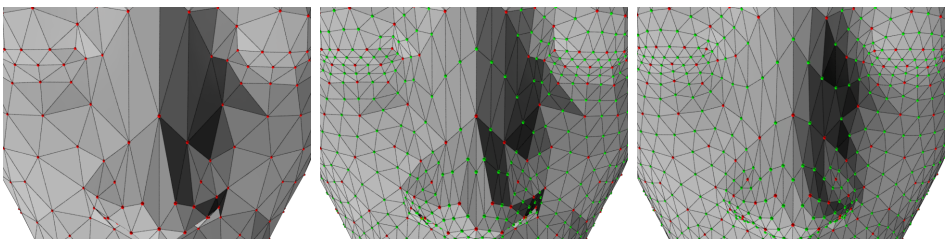
2 Detail Description

This part will talk about the tasks. Note that here we will only give some outlined descriptions, and for more details please refer to the papers listed in the references.

2.1 Subdivision

The subdivision of meshes has a wide application in many fields in computer graphics such as rendering, mesh editing etc.. This section will take *Loop subdivision* as an example to illustrate the general procedure for subdivision.

Loop subdivision was proposed by Charles T. Loop in [1]. It's an iterative method for the purpose of subdividing triangle meshes. One typical iteration is shown in Figure 1.



(a) Original mesh (b) Split points on edges (c) Refine

Figure 1: A single Loop subdivision iteration. New points (shown as green points in (b)) are sampled on the edges, and each face are divided into four new faces. And for the smoothness of the generated mesh, a refinement is necessary (shown in (c)).

For the new generated points on edge and the original vertices, a refinement step is necessary. Figure 2 shows a typical method to refine the points' positions.

For the details of the Loop subdivision (such as how to deal with the boundaries), refer to the Chapter 3 and 4 in [2], which describes a lot of other general subdivision methods. Usually these methods share a universal framework, so it's easy to implement others if you have one of them at hand.

2.2 Simplification

A simplification procedure of meshes can be viewed as a reverse process of subdivision, but usually the process of subdivision is hard to reverse. There are different strategies according to geometry primitives to decimate, such as vertex and edges. We will take *quadratic error metrics simplification* method (described in [3]) as an example to illustrate a simple mesh simplification method which belongs to the edge decimation strategy.

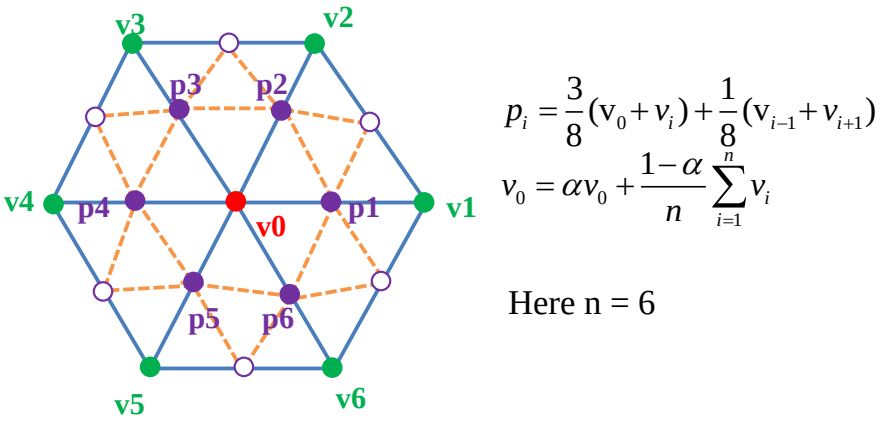
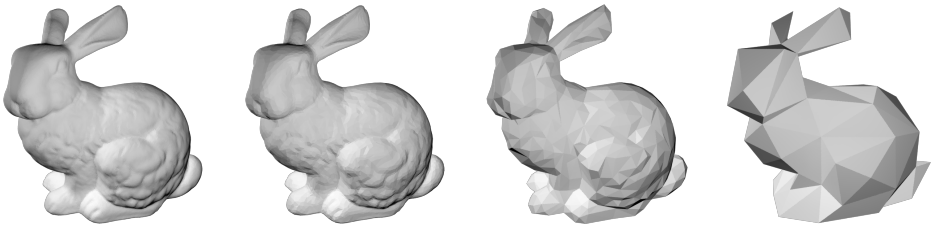


Figure 2: Refinement



#face = 144K #face = 14.4K #face = 1440 #face = 144

Figure 3: Quadratic error simplification

A typical procedure of quadratic error metrics simplification is shown in Figure 3. This algorithm merges two adjacent vertices in the mesh into a single one to reduce the mesh's size. This is called *edge collapse*, shown in 4.

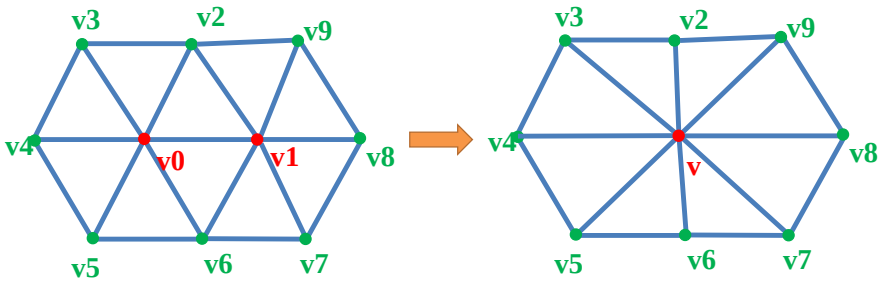


Figure 4: Edge collapse

To keep the outline of the mesh, it uses a metric attached to each vertex to measure the error produced by the edge collapsing. With error measured with this metric, an optimal position for the merged vertex will be calculated. For more details, please refer to [3].

2.3 Analysis

We'd like to see some analysis on your implementation in various aspects such as efficiency (both time and space) and effects (by comparing with other implementation) and so on.

For the comparison, there are different tools[1] to produce subdivision and simplification results. Here are two suggested ones:

- *MeshLab*. It's an open-source software and has versions for different platforms. Besides, it has both graphics and command-line interface for the convenience of different uses.
- *Computational Geometry Algorithm Library (CGAL)*, which is an open-source cpp library for the solutions of general computational geometry problems. It's easy to check out and modify its implementation, though the set up might be a little complicated.

3 Bonus

Any extra features can be considered for an extra credits. The difficulty and effect will determine how much extra you'll get. Please describe the features you implement in the final report in detail.

Below are some suggested options:

- Compare different subdivision and simplification methods to distinguish their differences such as feasible situations, efficiency etc..

References

[1] Charles T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.

[2] The Course Notes form SIGGRaph 99: http://multires.caltech.edu/pubs/sig99notes.pdf

[1] Note that there may be some differences such as parameter selections between different tools. So you don't have to feel confused about the differences in the results. On the other hand, you can compare them as the analysis part of your report.

[3] Garland M, Heckbert P S. Surface simplification using quadric error metrics[C]. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 1997: 209-216.*

[4] Hoppe H, DeRose T, Duchamp T, et al. Mesh optimization[C]. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques. ACM, 1993: 19-26.*

[5] Hugues H. Progressive meshes[C]. *Computer Graphics (SIGGRAPH 96 Proceedings). 1996: 99-108.*