Literature Review: Simplifying Zamani models for practical use

Daniel Burnham-King
Department of Computer Science
University of Capetown
BRNDAN022@myuct.ac.za

ABSTRACT

In the following paper I will review different papers and methods centered on the simplification and efficient display of large polygon meshes. In particular I am looking at adapting previous research on efficiently rendering large meshes to the data generated by the Zamani Group. The majority of the research took place more than a decade ago, and with advances in modern computing, memory limitations of old simplification algorithms may no longer play a factor. The Paper explores various possibilities, from hierarchical level-of-detail schemes to simplification methods.

1. INTRODUCTION

The African Cultural Sites and Landscapes Database project is dedicated to the recording and preservation of cultural heritage monuments and landscapes throughout Africa.

The project recognizes the importance of increasing international awareness of African heritage, documenting spatial information of heritage sites as well as their environmental context to provide material that can be used for education and research as well as restoration in conservation work.

The Zamani research group focuses on recording the physical and architectural nature of historically significant structures and landscapes on behalf of the project.

State-of-the-art laser scanning allows the Zamani group to generate highly detailed models that document the sites. These models may contain anywhere from 20 million to as many as 2 billion points [1] (Recent scans have more points than those mentioned at the time of publication-2012). The vast amount of memory required to load these models makes interacting with them practically infeasible, with long rendering times models must be simplified or split-up in order for current modern hardware to cope with the data load. Although it is possible that hardware will catch up with the high data load of the models, that alone cannot be relied on. With advances in scanning technology the size of scanned models will also increase, as they have in the past (resulting in the current predicament). An algorithmic solution is thus sought after.

This literature review examines some of the solutions that have been proposed for displaying and processing large models. The focus on literature centered on the feature preserving simplification of 3D meshes. For this project it will be important to review the literature on these methods and consider the implications that different simplification methods may have on the uses of data from the Zamani project.

2. EFFICIENT DYNAMIC DISPLAY OF LARGE 3D MODELS

Effectively displaying a 3D model requires render times short enough to provide real time feedback. Due to the large amount of vertices and primitives in Zamani models which may exceed a few gigabytes per model, manipulating the models is an exercise of the users' patience. Multiple seconds may be required for each frame to render, resulting in frustration for the users. The number of vertices being rendered is the primary cause of performance issues, reduce the number of vertices (and thus faces) and you reduce the required rendering time of each frame. This can be done by simplifying the models through the removal of vertices and faces (and edges), by loading lower detailed versions of the models if they are further away from screen (since the distance is obscuring the detail already) or by a combination of the above mentioned - hierarchical level-of-detail rendering [2, 6].

2.1 MESH SIMPLIFICATION

There are many existing mesh simplification methods - these methods all build a simplified approximation to the original model. They are devised to minimize the number of vertices or faces (or both) and may preserve topology, vertex locations and feature edges. In terms of maintaining a level of accuracy or maintaining features from the original models, different methods may also make use of global or local error metrics. It should also be noted that some algorithms or restricted to working on manifold surfaces (specifically speaking edge/face decimation) although these do generally preserve mesh topology [7]. [6] Describes the need for out-of-core simplification, processes that may run on external memory.

2.2 HIERARCHICAL LEVEL-OF-DETAIL SCHEMES

QSplat has been able to demonstrate real time progressive rendering of large models [2]. Through the use of bounding sphere hierarchies, QSplat was able to render a complex model (containing more than 127 million points). Decreasing render time greatly by only rendering 8 million points when the model was static (less when it was moving). The detail to which each bounding volume is rendered is dependent on the distance of the volume to the screen. Volumes that are far away will be rendered with simplified points. It was worth noting that although QSplat has been shown to be quite successful, the models it was used to render contain only a few million points - not the billions of points that Zamani is currently working with [1]. QSplat does not render meshes, instead it effectively displays a multi-resolution point cloud.

Lindstrom [8] proposed a system that uses out-of-core simplification as well as an oct-tree based hierarchical view dependent level of detail system. The large amount of data that 3D scanned models can contain renders most simplification algorithms inadequate, since most require the mesh to be loaded into memory. However, scanned meshes may greatly exceed the memory available in modern computers. Lindstrom's method performs all computations on disk, allowing for large models to be processed with minimal memory (tested on a model containing

372,767,445 vertices). A far stretch from the billions of points Zamani works with. Due the complex nature of architectural models, together with the large number of points contained in the Zamani meshes. It is not clear how successful this method will be with Zamani data.

The progressive mesh representation provides a scheme for "keeping track" of the iterations used to simplify a mesh, allowing them to be undone iteratively. Providing a level-of-detail model as well as selective refinement, allowing a specific region of a simplified mesh to be de-simplified (refined). The mesh can be constructed using an edge collapse operation together with a minimizing energy function to insure that surface geometry is preserved [3]. Zamani has simultaneous need for fast render times (thus simplification of models) and high detail meshes. Progressive meshing seems to be able to provide for both of these criteria. [5] Proposes a way of view-dependent refinement of parallel meshes. Allowing a mesh to be loaded in at low detail, and relevant parts of the mesh to be refined in parallel on modern GPU hardware. Thus saving large amounts of memory and maintaining high performance. Progressive meshes have not been used to represent models containing billions (or even hundreds of millions) of points, or for models representing architectural sites.

3. MEASURING ACCURACY WITH METRO

Comparing the accuracy of a simplified meshes approximation to that of the original is not easy [4]. Identifying critical flaws when comparing hundreds of millions of points and surfaces is practically infeasible to do manually. Many simplification algorithms do not provide an accuracy measure of their approximations.

For this purpose Metro [4] has been developed (Metro has since been incorporated into the Visualization and Computer Graphics library [9]). It is defined a as general and simple to implement tool which numerically compares two triangle meshes describing the same surface at different levels of detail. The tool is ignorant of the simplification algorithm used, thus it can be used to compare the accuracy simplified models regardless of the method used to simplify them.

4. CONCLUSION

There has been a substantial amount of research invested in the simplification of meshes to increase rendering performance. Enhancing a user's ability to interact with a complex mesh.

However, these simplification methods have not been tested on complex architectural models.

Zamani require interactive meshes with a high level of detail. Considering the above reviews. A level-of-detail hierarchical rendering scheme may provide a working platform for Zamani.

Different out-of-core and in-core simplification algorithms thus need to be explored for their effectiveness on Zamani data.

5. REFERENCES

[1] Rüther, H. Held, C. Bhurtha, R. Schroeder, R. Wessels, S. 2012 From Point Cloud to Textured Model, the Zamani Laser Scanning Pipeline in Heritage Documentation. SAJG. DOI= http://www.sajg.org.za/index.php/sajg/article/view/20

[2] Rusinkiewicz, S. Levoy, Marc. 2000 QSplat: A Multiresolution Point Rendering System for Large Meshes. ACM. DOI= http://dx.doi.org/10.1145/344779.344940

[3] Hoppe, H. 1996 Progressive Meshes. ACM. DOI= http://dx.doi.org/10.1145/237170.237216

[4] Cignoni, P. Rocchini, C. Scopigno, R. 1998 Metro: measuring error on simplified surfaces. VCG. DOI= http://vcg.isti.cnr.it/publications/papers/metro.pdf

[5] Liang Hu, Pedro V. Sander, and Hugues Hoppe. 2009. Parallel view-dependent refinement of progressive meshes. In Proceedings of the 2009 symposium on Interactive 3D graphics and games (I3D '09). ACM, New York, NY, USA, 169-176. DOI=10.1145/1507149.1507177 http://doi.acm.org/10.1145/1507149.1507177

[6] Eric Shaffer and Michael Garland. 2001. Efficient adaptive simplification of massive meshes. In*Proceedings of the conference on Visualization '01* (VIS '01). IEEE Computer Society, Washington, DC, USA, 127-134.

[7] Cignoni, P. Rocchini, C. Scopigno, R. 1997. A comparison of mesh simplification algorithms Link= http://webdocs.cs.ualberta.ca/~anup/Courses/604_3DTV/Presentation_files/Polygon_Simplification/7.pdf

[8] Lindstrom, P. 2003. Out-of-Core Construction and Visualization of Multiresolution Surfaces DOI= http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=432AE4C6CA1D470AC7CE6F7D2F13BB77?doi=10.1.1.15.8515&rep=rep1&type=pdf

[9] Visualization and Computer Graphics Library Link= http://vcg.isti.cnr.it/~cignoni/newvcglib/html/