3
VIEW-DEPENDENT SIMPLIFICATION ON OUT-OF-CORE DATASETS IN 3D REAL-TIME GAME ENGINE DEVELOPMENT

Tan Kim Heok, Daut Daman, Abdullah Bade,
Mohd Shahrizal Sunar

INTRODUCTION

Rapid technology growth in modern 3D scanning technology and the high complexity of computer simulations have led to a boost in the size of geometry data sets. Even the most powerful graphics hardware also is not able to handle the rendering of the extremely massive data. Noticeable delay or jagged effects always bother the rendering quality. It is significant especially in real-time and interactive application. The conventional simplification approaches are no more sufficient to handle it. Thus, out-of-core approach is invented. Besides, in many applications, surface attributes are important to show the details of the mesh. Therefore, automatic simplification is done on massive datasets while preserving its surface attributes. In this chapter, we present a preliminary approach to represent the data in an Octree structure and then to simplify the model using modified memory insensitive technique. During run-time, the portion of the visible mesh will be rendered view-dependently.

LEVEL OF DETAIL (LOD)

Since the mid nineteen-seventies, programmers have used Level of Detail (LOD) techniques to improve the performance and quality of their graphics systems. The LOD approach involves retaining a set of representations of each polygonal object, each with different levels of triangle resolution (Figure 3.1).

Figure 3.1 Three levels of detail of the Ludwig model. Left, 4998 faces; center, 1250; right 250 faces (Ribelles, et al. 2001)

ALGORITHM FRAMEWORK

This section introduces an approach for end-to-end and out-of-core simplification and view-dependent visualization of large surfaces. Besides, appearance preservation will be proposed as well. Here, the arbitrarily large datasets, which are larger than memory size, can now be visualized by giving a sufficient amount of disk space (ie. a constant multiple size of the input mesh). The preprocess work starts

with data preprocessing and then an Octree is constructed to partition the space efficiently. Consequently, a modified memory insensitive simplification is preceded (Lindstrom & Silva 2001). Finally, the view-dependent rendering on output mesh will be carried out during run-time. The off-line phases are performed on secondary memory whilst the run-time system only pages in the needed parts from the mesh in a cache coherent manner for rendering purpose. The framework overview is shown in Figure 3.2.

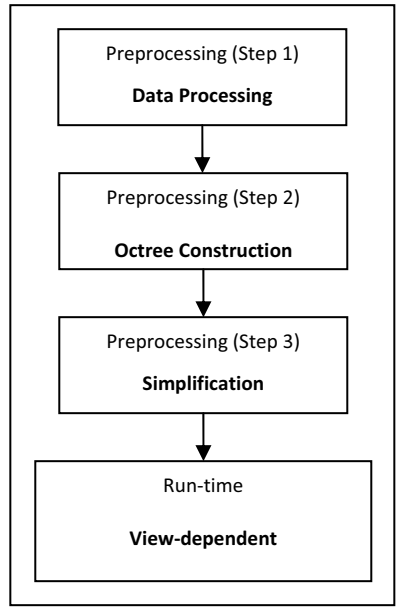

Figure 3.2 Framework overview

Algorithm starts with data processing process. It involves data loading into our system and dereferencing of triangle indices to their corresponding vertices. The experimental data is in PLY data format. It is one of the common file formats in storing the large datasets. However, these raw data are in indexed mesh. Even though the format is compact, the processing time is slow. Thus, it needs to be further processed before proceeding to simplification process. Therefore, we dereference a list of triangle to its vertices so that a triangle soup mesh is generated.

Secondly, an Octree is constructed to divide the loaded data into spatial space. The purpose is to make sure the data processing step becomes easier and neater. The triangular mesh is subdivided into its appropriate location. Because of the datasets size is too large for the available main memory size on commodity computer; hence the data in each Octree node is kept in its end node file. The end node files size is small enough to fit in main memory and are stored in an organized directory format. Thus, the file searching is easier to be performed.

In the run-time phrase, the visible nodes are extracted from Octree. Each of them is considered as active nodes and the vertex information is loaded into a dynamic data structure. The active nodes are expanded and collapsed based on view-dependent criteria. Besides all above, the rendering and refinement stages are run in parallel.

DATA PROCESSING

PLY data file has a header and followed by a vertex list and

face list. The file header starts with PLY and ends with END_HEADER. The Following is the general header structure:

ply
format ascii 1.0
comment …
element vertex num_of_vertices
property float32 x
property float32 y
property float32 z
element face num_of_faces
property list uint8 int32 vertex_index
end_header

A vertex list consists of *num_of_vertices*, a triangle list and *num_of_faces*. It is in indexed format which is space efficient. However, it slows down the processing time because the indexed triangle list needs to be converted to triangle soup style.

OCTREE CONSTRUCTION

From previous data processing step, the complete triangle soup file is loaded portion by portion into our Octree. Octree is chosen as it eliminates the computation time spent on processing on the empty space in a data model. Each time, the space is divided into eight cubes recursively until the tree is fully subdivided. Each internal node stores their directory path, so that end node file searching becomes

easier. Only the leaf nodes hold the triangle list.

Since the datasets could not fit into main memory, the vertices in each leaf node are written into every end node file. The file is small and is kept in organized directory structure. Each child node is contained in their parents' node directory (previous parents' directory). Hence, the tracking of every end node file is simple and better organized.

By using the constructed Octree as our spatial data structure, it makes our view-dependent rendering faster in general. It contains the whole world information. Every detail mesh in it is small and stored in end node files, thus makes the simplification easier to be performed.

CONCLUSION

There is some future works, including performing pre-fetching to accelerate the data paging and enhancing on-disk data handling. Besides, application like in medical visualization needs very accurate visualization and that requires better quality of simplified mesh. We can also extend this application to be used in network game.

REFERENCE

LINDSTROM, P. AND SILVA, C. 2001. A Memory Insensitive Technique for Large Model Simplification. In IEEE Visualization 2001, San Diego, CA, 121-126.

LINDSTROM, P. 2000. Model Simplification using Image and Geometry-Based Metrics. Ph.D Thesis, Georgia Institute of Technology.

RIBELLES, J., LÓPEZ, A., BELMONTE, O., REMOLAR, I. AND CHOVER, M. 2001. Variable Resolution Level-of-detail of Multiresolution Ordered Meshes. In Proc. of 9-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2001), Plzen: Czech Republic, 299-306.