A Real-time Rendering Method Based on Precomputed Hierarchical Levels of Detail in Huge Dataset

Zhou Kai[†], *and Tian Feng*[†]

[†] School of Computer and Information Technology, Northeast Petroleum University, Daqing, China

Summary
Traditional preprocessing methods, such as discrete LOD or continuous LOD, are inapplicable to large dataset consists of huge amount of models. Division granularity of discrete LOD is coarse while continuous LOD is time-consuming. Real-time rendering is a challenging and open problem to solve. A real-time rendering method to solve preprocessing problem in huge dataset based on pre-computed hierarchical levels of detail is proposed by different granularity and hierarchy division. Experimental result shows that the proposed method is more suitable for huge dataset.

Key words:
hierarchical levels of detail, real-time rendering, huge dataset.

1. Introduction

With rapidly increasing collections of 3D data and the development of automatic 3D modeling technology, there is a contradiction between rendering capability and accuracy for the computer. LOD technology is a major method in accelerated rendering. In 1976, Clark [1] proposed the concept of levels of detail (LOD). When objects cover a smaller area of the screen, you can use the lower-resolution model, otherwise higher-resolution model for fast rendering scenes. Now the LOD technology is divided into discrete LOD and continuous LOD. In discrete LOD, different hierarchy is gradually simplified, and there is not external contact between them, so we should choose the appropriate level to rendering [2] [3]. Discrete LOD is simple and it's efficiency is high, but its level is limited, so discrete LOD can't be suitable for a variety of applications. Division granularity of discrete LOD is coarse, for large scale models, the same LOD level is used in near and far viewpoint region, which affects the details of rendering. Continuous LOD is proposed to solve these problems. It allows the continuous variation of the same model on LOD level [4], and is widely applied to render large scale models such as terrain models. Continuous LOD is complicated in computing mechanism; some kind of record structure is generated, then 3D models are reconstructed during rendering. In comparison with the discrete LOD, the effects of rendering improved using continuous LOD, but the cost of rendering is also increased.

Erikson [5] proposed HLOD (the Hierarchical Levels of Detail) which is used to fast render huge dataset. The main idea of HLOD technique is granularity division and hierarchy based on discrete LOD. It adopts tree hierarchical LOD storage structure and takes the detail and rendering efficiency into account during rendering large scenes. But the method doesn't support the model texture, only support the color information.

A real-time rendering method for huge dataset based on preprocessing HLOD is constructed by Visual C++ and OpenGL to improve rendering effect which support texture. The following first part will briefly describe the HLOD principle. Then the detail of implementation is introduced. Finally, the experimental results and the future work are given.

2. HLOD Theory

The HLOD adopts hierarchy strategy where data is saved by tree hierarchical structure to simplify the scene rendering. The rendering effect of multi-level discrete LOD is similar with continuous LOD. For a scene, HLOD can be divided into two levels: the first level is the internal level of 3D model where a HLOD unit is one or more parts of 3D division. The second level is the scene HLOD where one unit is one or more models. The HLOD construction of the internal level is briefly described by quad tree. A model is subdivided twice to obtain C_1 ... C_9, as shown in Figure1; Figure 2 shows the obtained quad tree.

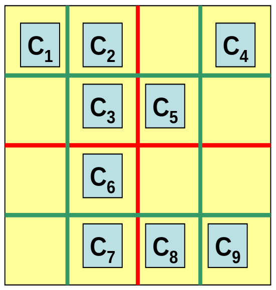

Fig.1.The model subdivision

Manuscript received July 5, 2014
Manuscript revised July 20, 2014

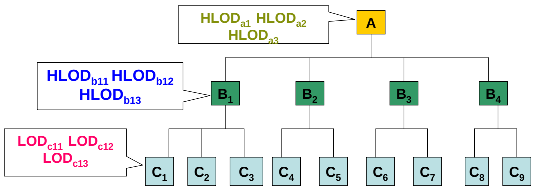

Fig.2.The model HLOD tree construction

Each node in quad tree contains geometric information of 3 different resolutions. For each leaf node, such as C_1, LOD_{c11}, LOD_{c12}, LOD_{c13} are get by simplifying geometric information, and LOD_{c11} is regarded as the finest layer, LOD_{c13} is regarded as the roughest layer. You can get 3 layers LOD information in each leaf node by this way, then construct HOLD though the bottom-up process. LOD data contained in the non-leaf nodes called HLOD. Such as the finest layer $HLOD_{b11}$ consists of $LOD_{c13,}$ $LOD_{c23,}$ LOD_{c33} and the boundary. $HLOD_{b12}$ and $HLOD_{b13}$ are obtained by directly simplifying $HLOD_{b11}$. We can build the HLOD tree of the entire model by repeating the process.

C_1 ... C_9 are regarded as models for scene HLOD, and the scene HLOD tree is constructed by a similar process. The difference is that the boundary of scene HLOD doesn't exist, and simplified unit is one or more models. After establishing model HLOD tree and scene HLOD tree, we can choose right HLOD hierarchy to render scene according to the user's view and error thresholds assigned by users.

3. Rendering Based on Pre-computed HLOD

Rendering based on pre-computed HLOD mainly need to solve the following problems:

(i) the pre-computation of the single model HLOD. It includes model space subdivision, model boundaries, bottom-up establishment of the single model HLOD and the design and storage of single model HLOD file;

(ii) the pre-computation of the scene HLOD. It includes the HLOD data ofall models, the scene bounding box, the scene space subdivision and the bottom-up establishment of scene HLOD;

(iii) the HLOD selection and rendering based on viewpoint.

3.1 The Pre-computation of the Single Model HLOD

Figure 3 is the pre-computation flowchart of single model HLOD:

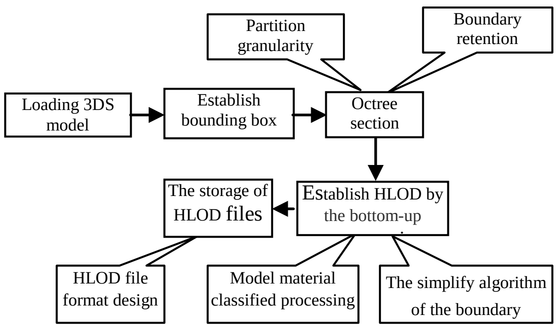
Partition granularity
Boundary retention
Loading 3DS model
Establish bounding box
Octree section
The storage of HLOD files
Establish HLOD by the bottom-up
HLOD file format design
Model material classified processing
The simplify algorithm of the boundary

Fig.3.The single model HLOD precomputation flow chart

Because 3DS file format is more widely used, the format is adopted in the paper. The two parameters which are the center of the bounding box and the size of the bounding box are computed. The average value of all vertex coordinates is the center of the bounding box. The bounding box size is twice the maximal distance from vertices to the center.

(i) The Octree Model Subdivision

The octree is one of spatial subdivision method. 3D space is divided into eight parts, and then each part is divided into eight parts again. Followed by the recursive process, the result tree structure is built. Each parent node in the structure contains eight child nodes that may save the relevant information of its parent node. The model data can be organized well and vertex number threshold can be predefined. The node will stop dividing if the vertex number is less than minVertNum.

The divided space will produce boundary patches. Because the boundary patches can't be simplified, otherwise the gap will exist between the various parts of the model, so each boundary will be retained. Octree subdivision algorithm is summarized as follows:

Step1. Determine the number of vertices in the node. If vertNum <= minVertNum, the current node stops dividing, otherwise continues with the following steps;

Step2. Obtain the center and the size of 8 child nodes in each node according to the center and size of the node bounding box.

Step3. Traverse the child nodes. According to the center and the size of the child nodes, the index array of each vertex in parent node is assigned to the sub-node;

Step4. Traverse topological sequence in node. For each topology group (C1, C2, C3), if the vertexes C1, C2 and C3 belong to different child nodes, (C1, C2, C3) should be stored in boundary patch array.

Step5. Repeat Step1 - Step4, until all nodes stop dividing.

Figure 4 shows the first layer boundary of the bunny model octree.

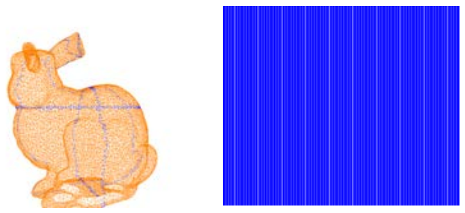

Fig.4.The octree subdivision boundary of the model bunny

(ii) The Bottom-up Construction of Model HLOD

Once the octree is constructed, the single model HLOD is built by the bottom-up process after the octree is constructed. The single model HLOD is constructed from the leaf nodes. The divided model is simplified by the simplification method with geometry and property boundary preservation. Single model HLOD algorithm is summarized as follows:

Step1. If the current node is a leaf node, the geometric information which is not simplified is regarded as the finest layer (LOD1). Then the boundary patch of LOD1 is reduced to half to get the finer layer (LOD2), the boundary patch of LOD2 is reduced to half to get the roughest layer (LOD3). During the whole process, we obtain the spatial distance error (distanceError) which is from each LOD to original geometric information.

Step2. If the node is not a leaf node, firstly call Step1 and Step2 to build HLOD for nonempty node. This is a recursive process. After all nonempty child nodes finish building HLOD, the geometric information of the roughest layer and the boundary reduce the boundary patch to half. The finest layer (HLOD1), the finer layer (HLOD2) and the roughest layer (HLOD3) are got in turn. For non-leaf nodes, its space distance error (distanceError) is the maximal value.

The HLOD structure for the octree root node is built by above algorithm. Figure 5 shows a single model HLOD rendering example. Eight child nodes of the root node are selected and rendered .The first child node selects the finest layer HLOD, others select the roughest layer HLOD. The border has been well preserved.

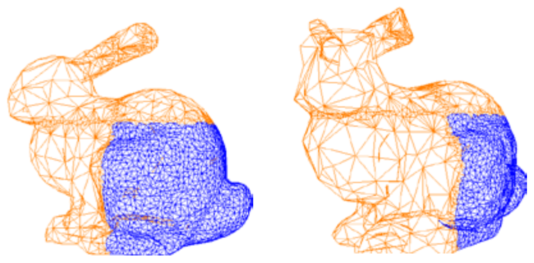

Fig.5. Single model HLOD rendering

(iii) The HLOD Files Design

Table I shows the time cost of the different models to establish the HLOD.

TABLE Ⅰ A single model hold time consuming

Model	The Patch Number	Division Granularity	Time-consuming (/s)
horse	39698	500	125
bunny	69451	500	180
bunny	69451	1000	130
P47	96720	1000	224
Armadillo	345944	1000	833
dragon	847414	1000	2212

When the model is larger, the establishment of the HLOD is relatively time-consuming. In order to improve rendering efficiency, HLOD file format is designed and HLOD model is stored by the binary block. Even the scene is changed again, the HLOD data file of each model is obtained without pre-processing, and directly builds scene HLOD. So it greatly reduces the waiting time. HLOD file format is shown in Figure 6:

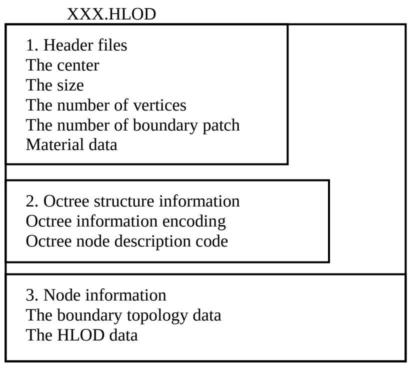
XXX.HLOD
1. Header files
The center
The size
The number of vertices
The number of boundary patch
Material data
2. Octree structure information
Octree information encoding
Octree node description code
3. Node information
The boundary topology data
The HLOD data

Fig.6.The HLOD file

The header file includes all data that are the center and size of the bounding box , the number of vertices and boundary patch, and material data. In order to record octree structure information, octree node information has been encoded for obtaining octree node information and octree node description [6]. Node description code is a code segment that identifies whether the node is leaf or not. We can reconstruct the original octree structure with node information code. The following is the brief algorithm for building the node code segment (OctreeCode) and the node description segment (NodeDiscription).

Step1. The octree root node is added to the queue (node queue);

Step2. After taking out a node from node queue, traverse its eight child nodes. If the child node is empty, put zero into the OctreeCode container. If the child node is not empty, put one into the OctreeCode. If the child node is a leaf node, put one into the NodeDiscription container, if non-leaf node, put zero into the NodeDiscription and the non-leaf nodes are added to the node queue;

Step3. Repeat Ste1 - Step2 until node queue is empty.

The original octree structure of the model must be reconstructed before reading HLOD data, so the octree structure information is put prior to the node information in the HOLD file. The third part of HLOD file is the HLOD information and boundary information of nodes. So one can put the HLOD information into the appropriate node in order to correctly read the entire HLOD file. 3DS file and HLOD file data size are illustrated in table 2.

TABLE Ⅱ HLOD file size table of the single model

model	The boundary number	The size of 3DS file/KB	The size of HOLD file/KB
blackface	36298	965	2210
horse	39698	975	2368
bunny	69451	1542	4560
P47	96720	1840	5131

3.2 The Scene HLOD Pre-computation

After obtaining HLOD data of all models, we begin to build scene HLOD. Because models are placed in the scene according to a certain position, the single model bounding box information which is loaded in HLOD file format is transformed according to the model space transformation information. After traversing the single model bounding box, their average value is treated as the center of the scene bounding box (SceneCenter). The max distance which is parallel to axial of the bounding box is treated as the scene bounding box size (SceneSize). After getting the scene bounding box, the scene is divided by octree method that is similar with the single model division. If the model number is less than 2 in the bounding box, stop dividing. You can get a scene octree after the division. The method of the scene HLOD establishment is similar to single model HLOD. After obtaining the single model HLOD and the scene HLOD, HLOD selection and view-based rendering will be done according to the error and roaming viewpoint assigned by the user.

When traversing the nodes in the scene, find the simplification error which is satisfies the Equation 1 and close to the right value to render. Equation 1 can be obtained according to Figure 8 where N is the width of the rendering window, P is pixel error and e is the simplification error of boundary patch when establishing HLOD.

$$e \le E = d.\frac{P}{N}\tan\theta \qquad (1)$$

The algorithm of the view-based rendering is described as follows. Fist, the scene root node is joined the queue (nodeQueue).

Step1. Get node from nodeQueue and traverse three HLOD of the node, then compare to simplification error e and E. If they meet the formula (1), the node pointer and the HLOD pointer are added to render queue (renderQueue);

Step2. After traversing the node's HLOD, if there are not nodes meet the formula (1), the non-empty child node is joined nodeQueue;

Step3. Repeat Step1 - Step2, until nodeQueue is empty;

Step4. Traverse the queue and render the queue.

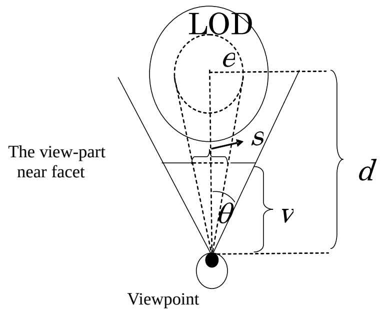

Fig.8.The view-based HLOD selection

4. Conclusion

A real-time rendering method based on pre-computed hierarchical levels of detail is proposed. It provides better rendering quality and reduces the complexity. The pre-computed results format is designed for HLOD model and increases the reusability of the method. Experimental result shows that the proposed method is more suitable for huge dataset.

Acknowledgment

This work is supported by Youth Foundation of Northeast Petroleum University (NO: 2013NQ120). We also would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Clark J. Hierarchical Geometric Models for Visible Surface Algorithms [C]// Communications of the ACM, New York, NY, USA: ACM Press, 1976, 547-554.

[2] T. Funkhouser, C. Squin, S. Teller. Management of Large Amounts of Data in Interactive Building Walkthroughs [C]// Proc. 1992 Symposium on Interactive 3D Graphics, Special issue of Computer Graphics. New York, NY, USA: ACM Press, 1992: 11-20.

[3] H. Hoppe. Progressive Meshes [C]// Proc. SIGGRAPH'96,

New York, NY, USA: ACM Press, 1996: 102-108.

[4] M. de Berg, K. Dobrindt. On Levels of Details in Terrains [R].Technical Report UU-CS-1995-12, Department of Computer Science, Utrecht University, New York, NY, USA: ACM Press, 1995: 426 – 427.

[5] Erikson C, Dinesh M. HLODs for Faster Display of Large Static and Dynamic Environments [C] // Computer Graphics (SIGGRAPH 01 Proceedings), New York, NY, USA: ACM Press, 2001: 113- 117.

[6] J Peng, C-C J Kuo. Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition [J]. ACM Transactions on Graphics(0730-0301), 2005，24(3): 609-613.

Kai Zhou received the M.E. degrees from Northeast Petroleum University in 2006. She is research assistant since 2006 and is working as a lecturer from 2009 in the Department of Computer and Information Technology, Northeast Petroleum University. Her research interest includes pattern recognition, virtual reality.

Feng Tian born in 1980. PhD and associate professor. Received his PhD degree in computer application technologies from the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University in 2014. His main research interests include image annotation, image tagging, cross media analysis, multimediamining, 3D model Retrieval, virtual reality and pattern recognition.