A Level of Detail Selection Method for Multi-Type Objects Based on The Span of Level Changed

Zhuo Yu
State Key Lab. of Virtual Reality Technology and Systems
School of Computer Science and Engineering, Beihang University
P.R. China, 100083
thursday@vrlab.buaa.edu.cn

Xiaohui Liang
State Key Lab. of Virtual Reality Technology and Systems
School of Computer Science and Engineering, Beihang University
P.R. China, 100083
lxh@vrlab.buaa.edu.cn

Zhiyu Chen
State Key Lab. of Virtual Reality Technology and Systems
School of Computer Science and Engineering, Beihang University
P.R. China, 100083
chenzy@vrlab.buaa.edu.cn

***Abstract*—Level of Detail (LOD) selection method is a key component in complex scene rendering and is widely used in time-critical systems. Many studies focus their efforts on developing the LOD selection method to deal with the problem which the scene contains small objects or objects with a single type. This paper gives a LOD selection method handling the situation that the scene contains multi-type objects while the data amount of these objects change dramatically. The preprocessing step of our work is using the extreme simplification way to simplify the objects with large number of primitives and using the Quadric Error Metrics to simplify the small ones. Then we calculate the span of level changed to each type of object separately. In runtime, we organize all the objects into a max heap by using the importance factor for rendering. We get the top element of the heap and change its level until the total data of objects satisfy the condition of time-critical method. Compared to existing solutions, our method is more suitable to the situation when the scene contains different type of objects.**

***Keywords*—LOD selection, multi-type objects, time-critical, span of level changed**

I. INTRODUCTION

With the development of the data acquisition technique, the types of objects in the virtual environment are increasing fast. The data amount of the complex scene with these different kinds of objects is larger than before. People often use the LOD (level of detail) method to simplify the models in order to decrease the total size of data. However, the rendering speed is changing dramatically while the total primitives of objects alter in the view frustum as the objects are located randomly. So a way which supports the rendering of the whole scene at constant speed is more imperative than before in some applications such as computer games, the emergency training, and military simulation.

Time-critical method is a good way to solve this problem. The core of this idea is to guarantee the rendering speed [1]. The first step of this method is to create the LOD models of each type of object, and then pick the right LOD model by using the limitation of rendering speed and some other factors such as the view position, projection area and the relative distance. Some researchers proposed lots of modifications to time-critical method [2, 3]. Most of them handled the problem which the scene contains many objects of a single type and the number of primitive is small.

Complex scene often contains more than one type of objects and the primitives of each type are different dramatically. Therefore, instead of considering all the objects with a single type, our method focus on the problem with the scene contains multi-type objects. In the former situation, users can select the LOD model by distance or projection area. In latter situation, the distance or projection area is not the most importance factor because the more primitives the object contains the more importance effects the object has when rendering. So we should distinguish the degree of primitive changes between different levels among multi-type objects first, and then give a method which can select the right LOD model for more important ones in a short time.

In order to solve the problem introduced above, we offer a LOD selection method based on the span of level changed (*SLC*) using time-critical method. The aim of our method is to give more attention to large object which has more importance factor to rendering than small ones. We build the LOD models of each type of object first. And then we calculate the *SLC* of each object and create the max heap by using importance factor which composed of *SLC* and other factors before selecting. We get the top element of max heap and change its level each time until the total primitives satisfy the condition for stable rendering when the total primitives beyond the limitation of time-critical method.

The main contributions of the algorithm presented in this paper are stated as follow:

- It gives a way to distinguish the importance factor for each type of object when the change of levels happens among different type of objects in rendering.
- It provides a quick way to solve the LOD selection between multi type objects based on our definition, especially the number of some objects' primitives are far bigger than others.

II. PREVIOUS WORK

Funkhouser and Séquin demonstrated the method which used a predictive selection scheme mainly based on the complexity of the current frame [4]. They formulate their way as an optimization task which is equivalent to a constrained version of the Multiple Choice Knapsack Problem (MCKP).

They give a way to solve this problem, but the result of their method is about half as good as the optimum one.

Gebbtti extended the method of Funkhouser and Séquin [4], and proposed the time-critical method to render the objects by using the LOD selection way in the scene [1]. The basic idea is to limit the total number of primitives in each frame. They assumed if the total number of primitives in each frame was the same, the render speed should be constant. The advantage of their method is considering the relationship of rendering quality and the rendering speed. The shortcoming is obviously as the process of calculating is quite complicated.

Some researchers have done lots of works under this method. Zach proposed a selection method which supported discrete and continuous LOD [5]. They project the objects on the screen and use the projection area as the key factor to select the LOD models. They regard that the bigger the projection area is the more importance factor the object has. This method is limited to handle the situation like the object is rotating. Zach implemented a solution to select the level based on the event-driven condition [6].

Zhang proposed a way to render complex scene with many type of objects, they combined several optimization techniques into their method [7]. They use the user pre-defined factor to each object when selecting. They can also handle the LOD selection of small dynamic objects.

Hern´andez applied a system to render the scene which contains different types of trees [2]. They allocate a value to each visible object by using the projection area of this object and estimate the rendering time in each time. If the object is not important to visibility, they decrease the time of current object and take the remainder time to other visible objects.

Gumbau offered a selection method which deal with the situation that the objects in a virtual environment are of a single type and the amount of objects is large [3]. They also use GPU to accelerate the speed of their algorithm.

III. DEFINITION AND OVERVIEW

We will introduce some definitions in the first part in order to present our method easily. The second part of this section introduces the overview of our method.

A. Definition

1) 1Level

Level of a LOD model is the value which can present the degree after simplifying the original one. There are lots of ways to define the level of an object. One way is to use the ratio of mesh number after simplifying the original mesh number to define the level. In this paper, we use the way proposed by Cohen to define the level [8]. To each level of detail model, let M_{object} be the mesh number of current model, and l be the level of current model. So we use the following formula to calculate the level of each object [8]:

$$l = log_2(M_{object})$$

The level defined here is a float value which multi-type models can contrast with different LOD models. In order to use this value more conveniently, we convert the result to the integer by using (1).

$$l = \begin{cases} floor(l) & (l - ceil(l) \geq 0.5) \\ ceil(l) \end{cases} \quad (1)$$

2) Span of Level Changed (SLC)

From the above definition, we can see that the more primitives the object has the more level the object is. But it is not enough to compare the changing of levels correctly by using the value of levels alone. So we use *SLC* (span of level changed) to present the degree of changing. Let *SLC* be the value of the LOD model changed from level i to level j, and we use (2) to present the definition.

$$SLC = max(l_i, l_j) * \| l_i - l_j \| \quad (2)$$

The symbol of $\| \|$ is the absolute value. There are two components in (2): the max of two levels and the subtraction of level changed. From (2) we can see that the *SLC* is only related to the amount of primitives and it can present the degree of level changed. For example, the mesh number of current LOD model is 5000, and the mesh number of next level is 2000, the value of *SLC* = 12*1=12.

B. Overview of Our Method

The method proposed in this paper is divided into two parts: the preprocessing part and the runtime part.

The preprocessing is the step to process the models. Most methods we introduced in section 2 use one way to simplify the model. In our method, we use two ways to simplify the models in order to give more attention to the difference between the different type models. We use QEM method to handle small models and use extreme simplification ways for big ones. We introduce our method in details in section 4.

The main step of runtime is introduced as follow: we get the objects in the view first and organize them into the visible set. Then we calculate the importance factor of each object in the visible set. We create the max heap by using the importance factor as key value. We change the level of top element in the heap and recalculate until the total amount of data is smaller than border when the sum of data beyond the boarder which supported for constant rendering.

IV. PERPROCESSING STEP

The total primitives of a complex scene which contains multi-type objects are often beyond memory space, so we should simplify the model before rendering. The preprocessing step is to build the LOD model for multi-type of objects. Luebke gives a good review of these techniques in [9]. In generally, there are two kinds of ways to simplify the models: the discrete method and the continuous method.

The method of progressive meshes was proposed for creating view-dependent continuous level of detail model by Hoppe [10]. He records every collapse process and creates a forest to store the corresponding data. In runtime, he uses the split way to get the detail of model when the view is closer to the model. The result of his method is good but the shortcoming is also obviously as the structure is big and the speed of traversing is slow.

So we use the discrete way to simplify the models in our method to avoid the time of traversing when the number of primitives is large. We also offer different methods to create LOD model by the amount of primitives belongs to each type of objects. To the object which contains small primitives, we use the traditional way as Quadric Error Metrics (QEM) to build the LOD models [11]. To the object which has big dataset like the happy body, we use the extreme simplification method to simplify the model aggressively and preserve the feature of objects better [12].

V. RUNTIME SELECTION

A. Condition of Time-Critical

Traditional time-critical method used the limitation of rendering time to achieve the target of time-critical and kept the quality of rendering result in runtime. In our situation, we want to give the solution of time-critical rendering when lots of different type objects in the scene especially one type of object contains huge primitives. The problem is the number of primitives in the view may beyond the limitation of time-critical when the view moves. So we should give a way to decrease the total primitives by using the LOD selection on different type of objects. In order to attach this target, we use the following assumption that the aim of our method is only focus on the rendering speed while not giving more attention to rendering quality.

Our solution follows the condition of time-critical method which the speed of each frame is constant. This can also be presented as the number of primitives for rendering in each frame is constant. So we can use the definitions denoted in section 3 to describe this constrain. We present the total primitives in the system which can be handled for stable speed rendering as M , use n as the total number of objects in the scene, and use the l_i as the current level of each object. Therefore, the objects in current view condition should satisfy the (3):

$$\sum_{i=1}^{n} 2^{l_i} \le M \tag{3}$$

Two situations will happen when rendering. Firstly, the objects in current view satisfy the (3), we do nothing but rendering. Secondly, t the objects in view do not satisfy the (3), we should change the level of objects in order to decrease the sum of primitives for constant speed rendering. The changing of level is determined by the importance factor of each object which introduced in the following.

B. Important Factor Based SLC

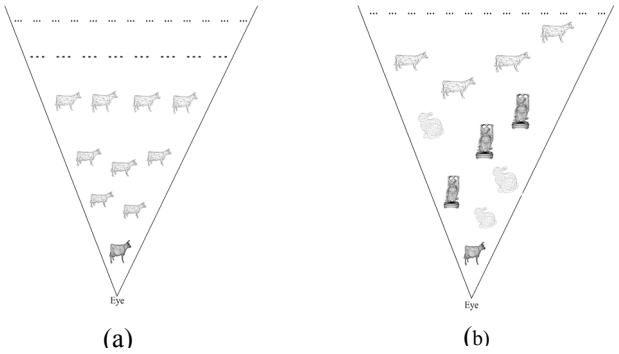

Figure 1. Scene with (a) same type objects (b) multi-type objects

The scene which contains many objects of a single type in one time is shown in Fig. 1 (a). The importance factor used for LOD selection in this situation depends on the distance between objects and view-point or the projection area in the screen easily because each object has the same number of primitive.

But to the situation like Fig. 1 (b), there is more than one type of objects in the scene. When the level is changed, the number of primitives belongs to an object may be the sum of other different objects. In this situation, the LOD selection can not only decide by the distance or projection area because the objects has different importance factor to system as the amount of primitives is different. The object containing more primitives should have more importance factor when rendering because it spends more traversing time and hardware resources.

Therefore, we give a way to present the importance factor for multi-type objects. The importance factor is composed of three elements: the relative distance, the *SLC*, the relative projection area of object.

1) Relative Distance

Distance is the basic factor in LOD selection [13]. However, the distance in this paper is a relative value instead of an absolute one. Use variable $D_{nearest}$ as the distance from view to the nearest object, let variable $D_{current}$ be the distance from view to current object (in Fig. 2). Use variable $d_{current}$ as the value of relative distance in (4).

$$d_{current} = \frac{D_{nearest}}{D_{current}} \tag{4}$$

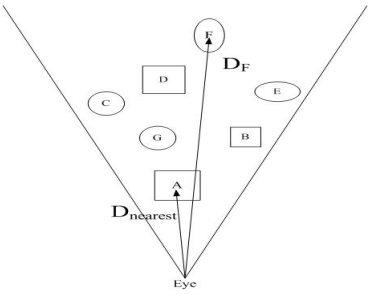

Figure 2. Relative distance

Once the variable $D_{nearest}$ is decided in one moment, the value of $d_{current}$ is determined by $D_{current}$.

2) Relative Projection Area

The projection area is a widely used factor for LOD selection. The shortcoming of this factor is that the size of projection area is related to the rotation and coordinate of objects. But in our situation, the multi-type objects have their own coordinate when modeling. So we just use the relative projection area to present the degree of its impact on rendering. Let $PA_{smallest}$ be the smallest projection area of all the objects in the current view condition, let $PA_{current}$ be the projection area of current object. Let $S_{current}$ be the value of relative projection area of the current object.

$$S_{current} = \frac{PA_{current}}{PA_{smallest}} \tag{5}$$

The meaning of (5) presents the relative importance for rendering quality while not decided by the area of one objects only.

3) Importance Factor for Rendering

In runtime, we give (6) to calculate the importance factor for each object in rendering when level is changed:

$$f = d * \frac{SLC}{S} \tag{6}$$

Let f present the affection factor. The d presents the relative distance and the S is the relative projection area of object. Equation (6) presents the way we calculate the importance factor when the view moved. So the meaning of the importance factor is the prediction degree of level changed for the rendering.

C. Structure and Selection Algorithm

The aim of our method is to select the right level of the object which has more importance factor and can satisfy (3).

In order to select the right level for objects quickly, we create a max heap by using the importance factor as the key. If the objects do not satisfy the (3), we just change the level of top element and recalculate after updating the heap.

1: get the visible objects set

2: calculate the importance factor of these objects by (6) and create max heap by importance factor

3: If the visible objects satisfy the (3)

4: render the visible objects and sleep until next frame

5: If the visible objects not satisfy the (3)

6: get the top element of heap and change its level

7: update the heap

8: go back to 3

Figure 3. Runtime LOD selection algorithm

If there are m objects in the current view condition, the complex of the algorithm is $m*log_2(m)$ which presents the worst situation. The whole algorithm is presented above (Fig. 3).

VI. EXPERINMENT

In our first experiment we use 3 types of objects including cow, armadillo and happy body whose original number of primitives is 5804, 345944, and 1087716 [14]. We create five LOD models for cow: 100%, 50%, 25%, 12%, and 6% using QEM method. We create eight LOD models: 25%, 12%, 6%, 3%, 1.5%, 0.75%, 0.37%, and 0.18% for armadillo and happy body using extreme simplification method. The value of *SLC* presented in the following.

TABLE I. SPAN OF LEVEL CHANGED WITH HAPPY BODY

SLC \ Level / Level	19	18	17	16	15	14	13
18	19						
17	38	18					
16	57	36	17				
15	76	54	34	16			
14	95	72	51	32	15		
13	114	90	68	48	30	14	
12	133	108	85	64	45	28	13

We can see from the Tab. 1 that the bigger the original level of an object is the more *SLC* value the object has when the level changed. So the *SLC* is a good way to present the degree of level changed which happens to large objects.

We create the virtual environment with 120 objects which composed of 50 cows, 40 armadillos and 30 happy bodies. We render the scene into an 800×600 window on a PC with a 2.4 GHz Pentium 4, 2 GB memory, and a NV GeForce 6800 card with 128 MB video memory. The rendering speed is about 51 frames per second by using back face culling and occlusion culling.

The Fig. 4 presents the number of primitives the scene contains and the primitives the system to rendering.

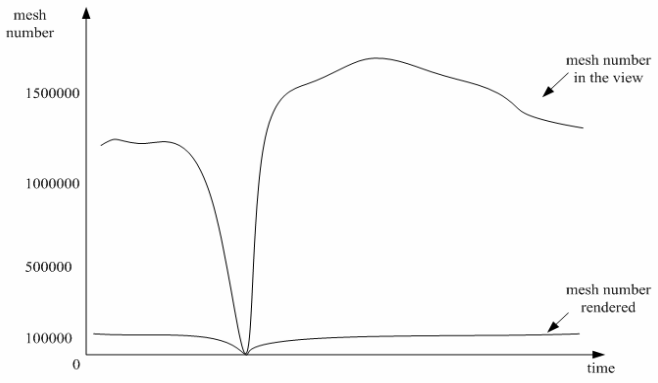

Figure 4. Mesh in view contrast to mesh for rendering

From the Fig. 4, we can see that the number of the primitives in the view is far bigger than the system rendered in practice, the method of ours can decrease the amount of primitive for rendering successfully. Moreover, we can also see that there is a dramatically drop happens in the Fig. 4 when the objects in the view is of a single type and contains small primitives like model cow. So the mesh number for rendering also changed correspondingly, but we use the mechanism of sleeping to deal with this situation. The speed of rendering is not affected by this sudden dropping of primitives.

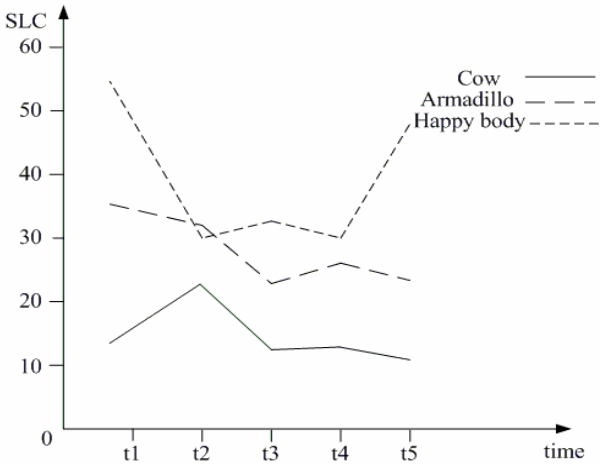

Figure 5. *SLC* with three different objects

Fig. 5 presents the *SLC* of three type objects when they are all visible in the view at five different moments. The *SLC* of these five moments does not contain zero value in order to present the level changing clearly. But in practice, zero value appears usually in lots of moments when the level does not change. From the Fig. 5 we can also find that in order to satisfy the time-critical rendering, the objects which contain more mesh changes more dramatically than small objects.

Fig. 6 shows the rendering result while the scene contains 5 happy bodies, 6 armadillos and 5 cows. These 16 objects are distributed randomly and 2 happy bodies near to the view point.

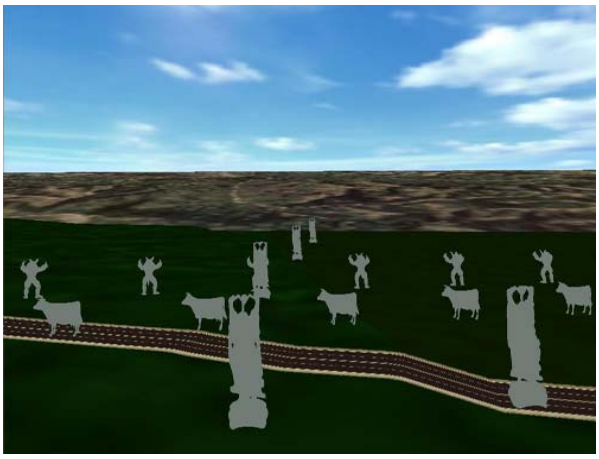

Figure 6. Rendering with few objects

In the second experiment, we create the same virtual environment with 200 armadillos and 200 bunnies [14]. The primitive number of bunny is 69451. We create five LOD models for bunny: 50%, 25%, 12%, 6% and 3% using QEM method. Fig. 7 shows the rendering result. The rendering speed is about 30 frames per second using back face culling and occlusion culling.

Figure 7. Rendering with lots of objects

VII. CONCLUSION

This paper provides a fast LOD selection method for multi-type objects using time critical method. The core component is to use the span of level changed (*SLC*) as the main component to calculate the importance factor for each type of objects when level changed. In order to give more reliable result, we use the relative distance of objects and the relative projection area when estimating the effect of object's level changing. The method introduced in this paper can select the appropriate LOD model for each object in the situation when there are lots of objects which contain massive meshes in runtime.

Although we can use our method to render the multi-type objects at a constant speed, the pixel error of rendering result is changed dramatically. The further work is to give more attention to the image quality of rendering. The prediction method in our method is conservatively, we only changed one level to the top element in the heap. It should be more exploitatively. We simplify the models in a static way in preprocessing step, so we should offer dynamic capability for users who want to change the LOD models in runtime.

Compared to our complex structure, another approach is to use more hardware friendly and simpler structures to store the data after simplification, so this can use the ability of hardware acceleration.

ACKNOWLEDGMENT

This research is supported by National High Technology Project (863 Project) (No.2006AA01Z333), Program for New Century Excellent Talents in University (NCET) and Natural Science Foundation of China (No.60533070).

REFERENCES

[1] Enrico Gobbetti, and Eric Bouvier, "Time-Critical Multiresolution Scene Rendering". Proceedings of the conference on Visualization '99, San Francisco, California, United States 1999, pp. 123–130.

[2] Eduardo Hernandez and Bedrich Benes, "Robin Hood's Algorithm for Time-Critical Level of Detail", Computer Graphics and Applications (GraphiCon'2005), Fifteenth International Conference, 2005.

[3] J. Gumbau., O. Ripolles., M. Chover , "LOD Manager: A framework for Rendering multiresolution models in real-time applications" , WSCG 2007. Plzen (Czech Republic), Feb. 2007.

[4] Thomas A. Funkhouser, and Carlo H. Séquin, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments", Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 1993, pp. 247–254

[5] Christopher Zach, Stephan Mantler and Konrad Karner, "Time-critical rendering of discrete and continuous levels of detail", Proceedings of the ACM symposium on Virtual reality software and technology , Hong Kong, China, 2002, pp. 1-8

[6] Christopher Zach, and Konrad Karner, "Fast Event-Driven Refinement of Dynamic Levels of Detail", Proceedings of the 19th spring conference on Computer graphics, Budmerice, Slovakia, ACM Press New York, NY, USA, 2003, pp: 55 - 62,

[7] Mingmin Zhang, Zhigeng Pan and Pheng-Ann Heng ,"Time-critical rendering algorithm with incorporation of LoD, visibility culling and object impostor" , The Journal of Visualization and Computer Animation, Volume 14, Number 4, September 2003, pp 211–223.

[8] Jonathan David Cohen , "Appearance-preserving simplification of polygonal models", North Carolina at Chapel Hill,1998.

[9] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, "Level of Detail for 3D Graphices", Morgan Kaufmann, 2002

[10] Hugues Hoppe, "View-Dependent Refinement of Progressive Meshes", Proceedings of the 24th annual conference on Computer graphics and interactive techniques, 1997, pp. 189–198

[11] Michael Garland and Paul S. Heckbert, "Surface Simplification Using Quadric Error Metrics", Proceedings of the 24th annual conference on Computer graphics and interactive techniques, Aug. 1997, pp 209–216.

[12] Pablo Diaz-Gutierrez M, Gopi Renato, and Pajarola, "Hierarchyless Simplification, Stripification and Compression of Triangulated Two-Manifolds", In Computer Graphics Forum (Eurographics), 24(3) 2005, pp. 457-467

[13] Miliano, V. "Unreality: Application of a 3D Game Engine to Enhance the Design,Visualization and Presentation of Commercial Real Estate". Proceedings of 1999 International Conference on Virtual Systems and MultiMedia (VSMM '99),1999, pp. 508–513.

[14] http://www.cs.princeton.edu/gfx/proj/sugcon/models/