# Research on Real Time Rendering Algorithm for Large Scale Complex Scenes based on Unity

Ziyi Wang [1, a], Meng Tan [2, b] and Gang Liu [1, c]

[1]School of Xihua University, Chengdu 610000, China

[2]School of Southwest Jiaotong University, Chengdu 610000, China

[a]ziyi0531@163.com, [b]1123508684@qq.com, [c]wangziyi@amazon.com

## Abstract

**Real time rendering of large-scale complex scenes is widely used in video games, virtual reality, film and other fields. The development of graphics hardware technology. Powerful graphics hardware technology to provide powerful technical support for the gorgeous game screen and the impact of the movie, but the user's requirements for graphics rendering quality far exceeds the scope of graphics hardware rendering capabilities. How to improve the rendering efficiency of large scale complex scenes with software is becoming a hot topic in the field of computer graphics. In recent years, a large number of complex scene rendering algorithm. The commonly used algorithms include occlusion culling algorithm, LOD algorithm, instantiation technology and so on. The author on the basis of previous research, the software realization of LOD occlusion culling algorithm and view dependent parallel optimization, and combine with the same rendering pipeline, the formation of a new large-scale complex scene rendering algorithm efficient. First of all, the author abandons the traditional hardware occlusion query algorithm, and implements the parallel optimization of the occlusion culling algorithm. The key of the hierarchical occlusion map algorithm is the construction of occlusion map. In the preprocessing stage, the occlusion database is derived by removing the specific conditions of the occlusion and removing the object which is not suitable for the occlusion. In the operational phase, covering algorithm from the database to extract the current set of view under the shelter, which rendered to the image space and construct the hierarchical occlusion map, finally use the hierarchical occlusion map in vivo with visual objects by calculating occlusion culling. The view dependent LOD algorithm and occlusion culling algorithm are combined by an array of visible objects. In the preprocessing stage, the algorithm is used to process the scene objects. At runtime, the algorithm extracts objects from the array of visible objects one by one, and calculates the appropriate LOD according to the distance between the object and the current viewpoint and the visibility of the object in the preprocessing phase. Finally, the triangle of the LOD of the calculated object is written into the triangle sequence to be rendered and sent to the rendering pipeline. In this paper, the algorithm is applied to the large-scale factory scene rendering in real-time. Experiments show that the algorithm can guarantee the rendering quality, real-time rendering of large-scale factory scene than the single LOD algorithm or occlusion culling algorithm have different degrees of improvement in the efficiency of rendering, and the scene is more complex, more can reflect the efficiency of algorithm.**

## Keywords

**LOD, Occlusion Culling Algorithm, Large-Scale Complex Scene.**

## 1. Main Research Contents

The main content of this paper is the optimization algorithm of large-scale complex scene rendering. In this paper, through the analysis and comparison of domestic and foreign related research algorithms, the algorithm of scene rendering algorithm and LOD algorithm are deeply discussed. Hardware occlusion query is a commonly used algorithm in recent years. Opengl and DirectX, such as the latest version of the graphics API has a functional interface to achieve hardware occlusion

query algorithm. The advantage of hardware occlusion query is simple and easy to use, but it also has obvious shortcomings. The asynchronous nature of the hardware occlusion query can cause CPU wait, CPU frequent interrupt and so on[1]. In recent years, more and more researches have adopted the method of batch occlusion query. As for the LOD algorithm, the related researches in recent years mainly focus on how to parallelize the LOD algorithm in GPU or combine it with the instantiation technology. The author consults the related literature and found that the occlusion culling algorithm and LOD algorithm combined with the scene to accelerate the algorithm with much, this paper studies how the occlusion culling algorithm based on LOD algorithm to accelerate the large-scale and complex scene rendering is proposed in such a consideration under the.

Different from the traditional scene rendering algorithm, this paper focuses on how to make use of the powerful computing power of modern GPU, and combine the occlusion culling algorithm and LOD algorithm in parallel. In particular, the author abandons the current popular hardware occlusion query algorithm, put forward to the "Visibility Culling using Hierarchical Occlusion Maps, a software for the improved occlusion culling algorithm in parallel, to avoid the hardware occlusion query while waiting for the results CPU wait for problem.

The "LOD View-dependent pruning for real-time rendering of algorithm trees" further applied in the rendering process, and designs a scene rendering pipeline with the occlusion culling algorithm efficiently combine.

The algorithm can be divided into two stages: preprocessing and runtime. The task of the preprocessing stage mainly includes the generation of a database of occlusion from the scene database, and the data of the scene database and the occlusion database are organized by the spatial data structure of the bounding volume level[2]. In the preprocessing stage, the object is simply meshed. At runtime, the algorithm maintains a rendering pipeline in each frame. Firstly, a suitable occlusion is selected from the occlusion database, and the scene is removed and the occlusion is removed. The calculated results are recorded in an array of visible objects. Next, the LOD algorithm is used to select the appropriate level of detail for the visible objects. Finally, the mesh is sent to the rendering pipeline for rendering.

In this paper, an example of a complex factory scene with about 5 million 810 thousand patches is used to implement the complex scene rendering algorithm. The algorithm not only realizes the real time rendering of the factory scene, but also improves the rendering efficiency greatly compared with the simple culling algorithm or the LOD algorithm.

## 2. Software implementation of hierarchical occlusion culling

Due to the asynchronous nature of hardware occlusion query method, it will lead to the problem of CPU waiting, CPU frequent interrupt and so on. In order to avoid this problem, this paper optimizes the occlusion culling algorithm, that is, the hierarchical occlusion graph algorithm.

The main idea of the hierarchical occlusion map algorithm is to divide the objects in the scene according to the specific standards for the collection and shelter shelter will be set, covering set mapping to the pixel level, and on the basis of constructing the hierarchical occlusion map, finally using the hierarchical occlusion map to be covered for occlusion culling test set[3].

The core of hierarchical occlusion culling algorithm is the construction of hierarchical occlusion map. The hierarchical occlusion map is the projection set of the different resolution of the occlusion in the pixel space, which records the occlusion effect of the occlusion in the scene. The hierarchical occlusion map can be constructed quickly and aided by occlusion culling. The hierarchical occlusion culling algorithm decomposes the visibility problem into two sub problems: the overlap test of two dimensional space and the depth measurement. The former is used to test whether the projection of the occluded object in the screen space is completely in the projection of the occlusion, and the latter is used to test whether the occluded object is behind the occluded object. The algorithm uses a hierarchical occlusion graph to perform overlapping testing, and uses the depth estimation buffer to

carry out the conservative depth test. Different from the traditional Z-buffer algorithm, overlapping test is an important part of the algorithm, not only as a secondary test of the depth test.

In addition to maintaining a database of the scene, hierarchical occlusion culling algorithm also maintains a database covering, covering database is in the preprocessing stage from the scene database according to the specific conditions of shelter derived. The two databases are surrounded by hierarchical spatial data structures that organize data[4]. The algorithm is divided into two stages: the construction of hierarchical occlusion map and the use of hierarchical occlusion map for visibility culling.

## 3. Build hierarchical occlusion map

The construction of the hierarchical occlusion map can be divided into three stages: the removal of the visual object, the choice of the occlusion, and the construction of the hierarchical occlusion map.

First, the algorithm traverses the bounding layer of the occlusion database, removing the occlusion outside the visual body. Next, according to the temporal coherence between frames, the algorithm selects the appropriate occlusion from the occlusion collection. Then the selected occlusion is rendered to the frame buffer to form an original occlusion map. The original occlusion map is a pixel image, the pixels in the image are not white, black, no light, no texture. The black part represents the pixels in the pixel space. The original occlusion map is the basis of constructing the hierarchical occlusion map, and the upper level of the occlusion map is constructed by the original occlusion map.

## 4. View frustum culling

First of all, the scene database and the occlusion database are removed from the scene, and the objects that are not visible in the current view are deleted. Visual object culling can be combined with bounding volume level to accelerate the culling. The algorithm starts from the root node of the bounding volume hierarchy, and if all the nodes of the root node do not intersect with the visual object, then all the nodes of the root node do not intersect with the visual object. If the root node is completely in the body, then all the nodes of the root node are in the visual body. If the root node and the visual body intersect, each node of the root node should be tested one by one to see whether the intersection of the body, remove the completely disjoint parts. This layer by layer scanning will be able to get rid of objects outside the visual body[5].

## 5. Occlusion selection.

In each frame, the algorithm selects a set of occlusion according to the current point of view. The optimal set of objects is the collection of all the objects in the scene. Of course, if you can find all the visible objects, then the object can be rendered between the large, do not have to remove the occlusion. Moreover, it is not necessary to find out the amount of all the visible objects in the scene. Therefore, the algorithm should be able to choose as much as possible to see the object as a shelter.

The hierarchical occlusion culling algorithm can be used to find a set of occlusion that can effectively block a considerable part of the scene. First, we derive a database of objects from the original scene database. At runtime, the algorithm dynamically selects the occlusion collection from the occlusion database.

## 6. Build hierarchical occlusion map

In order to select the appropriate occlusion from the occlusion database, we construct the hierarchical occlusion map using the selected occlusion set. If an opaque object is projected into the screen space, the area of the projection becomes opaque. The opacity of a block in a screen is the ratio of the area in the block to the area occupied by the block. The occlusion map is a two-dimensional array of the opacity of the record block. Here you need to explain, here the "block" is not a strict sense of the pixel block, it may also be a block of 2*2 pixels, but also may be a block of 4*4 pixels. The original image

is obtained by the occlusion of the object set. Each block of the original occlusion map corresponds to the pixel block of the screen space.

## 7. Occlusion culling using hierarchical occlusion map

This section mainly introduces how to use the hierarchical occlusion map for occlusion culling. The occlusion culling algorithm is divided into two parts: overlapping test and depth comparison. The relationship between two objects in space should be calculated in three dimensions. The hierarchical occlusion map is divided into two parts: the overlapping test of two-dimensional graph and the depth comparison of one dimension.

## 8. Overlap test

The overlapping test of the hierarchical occlusion map is a method of overlapping the projection of the pixels which are hidden in the scene and the projection of the pixels. If the occlusion is not overlapped or partially overlapped with the projection of the pixel and the projection of the object in the two-dimensional plane, the occlusion is highly likely to be visible and should be sent to the next stage of the rendering pipeline. If the projection of the occluded pixels is within the projection of the occluded object, the depth of the occluded object needs to be tested to determine whether the occluded object is occluded.

The exact overlap test needs to scan the potential occlusion and each pixel of the occluded object one by one, and the time complexity is O (N*M), where N, M is the number of pixels of the occlusion and occlusion respectively. Obviously this efficiency is unacceptable[6].

A simple, efficient and conservative overlapping test method is used in the hierarchical occlusion map algorithm. For all objects in the scene, the algorithm calculates the bounding rectangle in the screen space. The bounding rectangle is the projection of the space bounding box of the object in the screen space. Surrounded by a rectangle is a superset of the real objects on the screen space projection. In the overlap test, the algorithm uses the projection of the space around the rectangle instead of the object. Not only is it easy to compute the bounding rectangle, but because of its simple geometry, it is much faster to traverse a rectangular cover than to cover a specific object.

The hierarchical occlusion map algorithm is used to accelerate the overlap test by using different resolution occlusion maps. The overlapping test of the object's bounding rectangle and the occluded image does not start with the 0 layer, nor does it start at the top, but it starts with a block size that is about the same as the size of the rectangle. If the bounding rectangle starting from the top, and the top block surrounded by more than a rectangle, then surrounded by the most can only cover a rectangular block, then the algorithm through the overlap test several necessary to eliminate the possibility of being blocked. If the overlap test is started from the 0 layer, the algorithm does not need to construct the hierarchical occlusion map, which is contrary to the original intention.

The pixel level occlusion map and object graph algorithm to check if the bounding rectangles overlap, all pixels are completely transparent, then we can conclude with no obstructions surrounded by rectangular overlapping objects are visible, should be sent to the rendering pipeline. If each pixel is surrounded by a rectangle with the cover, the depth test should be performed. If it is not the top two cases, it should be put out of which completely transparent pixels, into the next layer of the pixel is not completely transparent pixels overlap test. This process is repeated until it satisfies any of the above.

## 9. Depth comparison

Occlusion is proposed to consist of two steps of overlap test and depth comparison. The hierarchical occlusion map can only be used to provide the overlapping test of two dimensional plane occlusion and occlusion. The overlap in the two-dimensional plane of the shelter and shade, the depth is unknown, algorithm cannot detect the occluded object is sheltered behind, so the algorithm needs to

provide another depth information to complete occlusion culling. This section describes the algorithm used in depth comparison method.

Single Z plane Single Z plane is a simple depth comparison method. The Z plane is a plane parallel to the near clipping plane. The depth value of the Z plane is larger than the depth value of any barrier, which separates the underlying occlusion from the occlusion. The depth of any object in the plane is considered to be greater than the depth of the object. Therefore, an object that is within the projection of the object and is in the Z plane is a completely occluded object. Single Z plane is relatively simple to achieve, but the accuracy is not high, is a simple extensive depth comparison method.

## 10. Depth estimation buffer

Depth estimation buffer provides a conservative estimate of the depth of occlusion. Unlike the Single Z plane, the depth estimation buffer divides the screen space into several regions, not just a plane to represent the depth values of the entire set of occlusions. Depth estimation is more fine-grained, and the results are more accurate. Essentially, depth estimation buffering is an improvement of the Single Z plane.

Each frame of the scene must construct a depth estimation buffer. The estimation of the depth of the structure is needed to determine which pixels are projected to the image space. If the projection and scanning directly to the shelter, the workload will be very large, the efficiency will be very low. The algorithm uses the same method as using the hierarchical occlusion map to overlap, and the object is used to estimate the projection of the object in the screen space. The depth value of the occlusion is conservatively estimated by the bounding volume of the object from the farthest point of view. Before the depth value is written to the depth estimation buffer, the algorithm first checks whether the depth values of the pixels in each rectangle are overwritten. If not, it is written directly to the depth estimation buffer, otherwise the depth value is compared with the original depth value. If the depth value to be written is larger than the original depth value, the depth value is updated. The algorithm performs this operation for all the barriers, and the final depth estimation buffer is obtained.

Depth estimation buffer support for conservative depth comparison of potential occluded objects. The algorithm gives the depth value of the nearest point of the object's bounding body from the area covered by the rectangle that is hidden in the screen space. Then, the pixels covered by the rectangle are detected one by one to see whether the depth value of the pixel is higher than that of the surrounding rectangle. If so, then the object is conservatively estimated to be placed in front of the shield and needs to be sent to the rendering pipeline for the next step.

## References

[1] U Assarsson, T Moller.Optimized view frustum culling algorithms for bounding boxes [J] . Journal of graphics tools,2000,5(1): 9-22

[2] N GREENE, M KASS, and G MILLER.Hierarchical Z-buffer visibility [J] .In Computer Graphics (Proceedings of SIGGRAPH’93), 1993:231-238.

[3] J BITTNER, M WIMMER, H PIRINGER and W PURGATHOFER.Coherent hierarchical culling: Hardware occlusion queries made useful [J] .Computer Graphics Forum, 2004, 23(3): 615–624.

[4] M GUTHE, A BALAZS, and R KLEIN.Near optimal hierarchical culling: Performance driven use of hardware occlusion queries[J].Eurographics Symposium on Rendering 2006:207-214.

[5] O MATTAUSCH, J BITTNER and M WIMMER.Chc++:Coherent hierarchical culling revisited [J] .Computer Graphics Forum,2008, 27(3):221–230

[6] Clark J H. Hierarchical geometric models for visible surface algorithms[J]. Communications of the ACM, 1976, 19(10): 547-554.