DYNAMIC SCENE OCCLUSION CULLING IN ARCHITECTURAL SCENES BASED ON DYNAMIC BOUNDING VOLUME

BALDEVE A/L PAUNOO @ BALU

A thesis submitted in fulfilment of the
requirements for the award of degree of
Master of Science (Computer Science)

Faculty of Computer Science and Information Technology
Universiti Teknologi Malaysia

APRIL 2006

In dedication to my beloved mother, family and friends.

ACKNOWLEDGEMENT

The author would like to extend his grateful appreciation to all those who have contributed directly and indirectly to the preparation of this thesis. First of all, the author would like to extend his gratitude and thanks to Professor Madya Daut bin Daman for his supervision and efforts in improving the technical content and quality of this work. The author is also grateful to the Ministry of Science, Technology and the Innovation (MOSTI) for sponsoring his studies. Thousand thanks to Wong Yin Fong and my family upon their boundless support, help and encouragement during this research. The author extends his gratitude to Lee Kong Weng, Saandilian and Yusrin for their help and support in the course of his work. The author also would like to express his thanks Mr. Zaprjagaev Alexander who is a freelance graphics programmer in Tomsk, Russia for his kind assistance in new rendering technology.

ABSTRACT

Visibility algorithms have recently regained attention because of the ever increasing size of polygon datasets and more dynamic objects in a scene. Dynamic objects handling makes large polygon datasets impossible to display in real time with conventional approaches. Therefore, occlusion culling techniques are required for output-sensitive rendering. Most scenes are displayed with static objects and only a few use dynamic objects in their visualization. In this thesis, the aim of the research carried out, is to handle dynamic objects efficiently with faster frame rate display. This algorithm is implemented using portal occlusion culling and kD-tree, which is suitable for indoor and architectural scenes. An occlusion culling technique was developed for handling dynamic objects in static scenes using dynamic bounding volume. Dynamic objects are wrapped in bounding volumes and then inserted into spatial hierarchical data structure as a volume to avoid updating the structure of every dynamic object at each frame. Dynamic bounding volumes are created for each occluded dynamic object by using physical constraints of that object and are assigned with a validity period. These bounding volumes and validity periods are later inserted into kD-tree. The dynamic objects are ignored until the bounding volume is visible or the validity period has expired. After numerous tests and analysis have been done, dynamic bounding volume culling shows better performance than portal culling especially when there are many low speed dynamic objects in the scene. Dynamic bounding volume culling proved to be efficient in avoiding enormous calculations of dynamic object’s position thus improves the rendering speed.

ABSTRAK

Algoritma ketampakan mula mendapat perhatian semula kerana saiz dataset poligon di dalam suatu pemandangan menjadi bertambah besar dan bilangan objek dinamik pula bertambah banyak. Adalah mustahil untuk memaparkannya objek dinamik dalam masa nyata dengan menggunakan teknik yang sedia ada. Oleh yang demikian, pengabaian pemandangan adalah diperlukan untuk penjanaan lorekan output yang sensitif. Dalam pemvisulan, kebanyakan pemandangan dipaparkan menggunakan banyak objek statik berbanding dengan paparan yang menggunakan bilangan objek dinamik yang sedikit. Dalam tesis ini, matlamat penyelidikan adalah untuk mengurus objek dinamik dengan lebih efisien beserta keupayaan paparan kadar kerangka yang cepat. Algoritma yang dicadangkan, sesuai untuk pemandangan dalaman dan arkitektural, dilaksanakan menggunakan pengabaian pemandangan portal dan pepohon-kD . Satu teknik pemandangan telah dihasilkan dengan menggunakan isipadu kekangan dinamik bagi menguruskan objek dinamik dalam pemandangan statik. Objek dinamik terkandung di dalam isipadu kekangan dan kemudiannya dimasukkan ke dalam struktur data hierarki sebagai sebuah objek isipadu. Proses ini bertujuan untuk mengabaikan pengemaskinian struktur data tersebut dalam setiap objek dinamik pada setiap kerangka. Isipadu kekangan dinamik yang dihasilkan bagi setiap objek dinamik menggunakan kekangan fizikal objek tersebut akan dihadkan tempoh jangkahayat. Tempoh jangkahayat dan isipadu kekangan seterusnya dimasukkan ke dalam pepohon-kD. Objek dinamik akan diabaikan sehinggalah isipadu kekangan kelihatan atau tempoh jangkahayatnya telah tamat. Selepas beberapa ujian dan analisis dijalankan, teknik pengabaian pemandangan isipadu kekangan dinamik menunjukkan prestasi yang lebih baik berbanding dengan teknik pengabaian pemandangan portal terutamanya apabila kelajuan objek dinamik adalah rendah. Pengabaian pemandangan isipadu kekangan juga terbukti lebih efisien dalam mengabaikan pengiraan yang banyak dalam menentukan posisi objek dinamik dan juga boleh meperbaiki kelajuan lorekan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

S	-	Subset of polygons
$\mathbf{P_n}$	-	Polygon
$\boldsymbol{p}$	-	Light point
$\boldsymbol{A}$	-	Set of polygons in shadow
$\boldsymbol{R^n}$	-	Set of polygons in the scene
$\boldsymbol{P}$	-	Light surface

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

3D computer graphics have recently gained great popularity. In addition to the classical uses of three-dimensional (3D) graphics, such as computer-aided design (CAD), military simulations and special-effect animation in movies, games and commercials, new applications are sprouting that are accessible to computer users worldwide. Since the very beginning, visibility determination has been a major problem for applications with millions of polygons [4]. Even though graphic hardware capability has increased tremendously, fast and efficient computation is not achievable by rendering the whole scene. To achieve the reality illusion, finely-detailed and smooth scenes must be displayed at a high refresh rate at least 30 frames per second [7]. But these are contradicting requirements; detailed scenes require large, complex models and these models take a long time to render, even when hardware support such as Z-buffering is available. If the scenes are dynamic, containing moving objects, then this problem become worse because additional time is needed to update the model to reflect these objects motions. This makes the required rendering rate hard to attain.

Due to more demanding graphic requirement to render increasing size of datasets in the scene and more dynamic objects, visibility algorithms have been revisited by researchers. It is impossible to display these scenes in real time with conventional approaches. Visibility culling intend at omitting invisible geometry before actual hidden surface removal is performed. Many of the researches focus on algorithms for computing tight estimation of visible sets by only drawing visible sets which the subsets of primitives which contributes at least one pixel of the screen.

Interactive three-dimensional computer graphics and animation with dynamic objects have gained popularity in 3D game over the last few years. However, their further acceptance is inhibited for dynamic game scenes by the amount of computation they involve [4]. If the game scenes are dynamic, containing moving objects, then additional time is needed to update the model to reflect these objects' motions. This makes it hard to attain the goal of interactivity in real-time game which requires images to be rendered at rate of about 30 frames per second. Hence, unwanted calculation should be ignored with visibility culling before sending to rendering pipeline.

1.1 Background

Pioneering the work in visibility algorithm includes Jones [13] and Clark [14]. However, recently, Airey [15], Teller and Sequin [16] and Greene, Kass and Miller [10] have revisited visibility algorithm and discovered new method to speed up occlusion culling using object base and image base algorithm.

Plantinga [17] proposed important data structure in computer vision called scene graph. Aspect graph encodes analytically all the information for efficient display. However, the worst complexity is high and in three dimensions, it can be large as $O(n^9)$ number of segments. Therefore, it turns out that computing aspect graph is computationally impractical.

Airey et al. [15] and Teller and Sequin [3, 16] proposed an occlusion culling algorithm for static architectural scene and developed foundation for the recent works. This conservative technique needs to be pre-computed to calculate potential visible sets or PVS. Their technique is suitable for indoor architectural scenes where rooms inter-connected with another with portal. Their works were further extended by Luebke and Georges [18]. Instead of region based visibility, they proposed point based visibility. Luebke and Georges perform on the fly visibility calculations using depth traversal of the cells using screen space projection of the portals. However, to develop a dynamic scene, it needs to update hierarchical data structure.

A similar way to look at the occlusion is shadow generated from a light source from the viewpoint. Therefore, Hudson et al [19] proposed an approach on dynamically choosing a set of occluders and computing their shadow frusta, which is used for culling the bounding boxes of a hierarchy of objects. However, it still needs to depend on the speed to update hierarchy data structure for dynamic scenes. Brittner, Havran and Slavik [20] improved the method described by Hudson using kD-tree. They combined the shadow frusta of the occluders into an occlusion tree.

Greene [10, 21] uses an octree for object precision and a Z-pyramid for image precision. The Z-pyramid is a layered buffer with a different resolution at each level. While the hierarchical Z-buffer algorithm is primarily intended for static scenes, Greene's PhD thesis [21] mentions a few ideas for handling dynamic objects. However, none of them has been implemented or seriously explored. Z-buffer in the hardware needs to be modified to allow real-time performance.

A similar method with hierarchical Z-buffer was proposed by Zhang [22]. It decouples the visibility test into an overlap test and a depth test. It also supports approximate visibility culling where objects that are visible through only a few pixels can be culled using an opacity threshold. This method needs to preprocess a database of potential occluders. For dynamic scenes, both the potential occluder set and hierarchical data structure are omitted and object bounding boxes are used instead. However, due to the omission of hierarchical data structure, this method is not output sensitive for increasing size of dynamic models.

Naylor et al. [11, 12] proposed an algorithm that performs output-sensitive visibility calculation using kD-tree. A kD-tree (axis-aligned binary space partitioning) tree can represent the scene itself without additional data structure. The construction of kD-tree tree is very time consuming but it is only constructed once as preprocessing stage and subsequently used for visibility calculation from many viewpoints.

1.2 Motivation

Current occlusion culling algorithms scenes [3, 11, 12, 15, 16, 18, 19 22] are not suitable for dynamic scenes because these algorithms highly depend on spatial hierarchical data structures and pre-computation. It consumes longer time [35] to build the spatial hierarchical data structure compared to just render the whole scene from any viewpoint. The scene structure is built during pre-processing because it consumes a lot of time and all the polygons in the structure are static. Polygon datasets cannot be altered in real-time to represent current dynamic position.

Some of the existing algorithms allow the exploitation of temporal coherence in animation sequences [7]. However, they are restricted to walkthrough animations, in which the scene is static and only the viewpoint moves through it. If there is any dynamic object in the scene, data structured that represent the scene will be outdated. Data structure needs to be modified for every frame to represent dynamic objects' positions and their movements. Even though it is possible to update the hierarchical structure, it consumes a huge amount of time. For example, a small movement of a polygon in a scene which is represented in a BSP tree will cause the whole tree structure need to be changed. In addition, if this update is not done carefully, it might use more processing and might results far too much time longer than the normal rendering of all the objects in the scene. The hierarchical tree update for dynamic object movement should be avoided especially when they are not even visible to achieve output sensitive rendering. However, when the dynamic objects are visible, they should render include them in the scene hierarchical tree.

Obviously, it is impossible to reconstruct the data structure by utilizing current hardware capability when the dynamic objects move in the scene. More calculations needed to update the data structure for the scene with dynamic objects.

The data structure should not hold the dynamic objects. The existing data structure should be updated for the dynamic objects motions, rather than being initialized from scratch [7].

It will waste time to update the structure for occluded dynamic objects as well as for visible ones. The update should only be performed in the visible parts of the data structure to preserve output-sensitivity. Invisible dynamic objects should be ignored until the dynamic objects in predefined path might visible from the viewpoint. Most of the scenes are static and only few objects are dynamic. Therefore, more efficient algorithm should be developed without trading performance. A mechanism is needed to ignore hidden dynamic objects most of the time but notified when they might no longer be hidden. Using the proposed algorithm, real time dynamic scene can be produced using normal personal computers.

1.3 Problem Statements

Current algorithms are not suitable for dynamic scene occlusion culling because the data structure used by occlusion culling method is updated to indicate the dynamic object’s current position. Binary space partition tree data structure is not fast enough to update in real-time. The update should only be performed if the objects might be visible. The algorithm should ignore most of the occluded objects most of the time, but notified when they may be exposed, and displays them as necessary.

1.4 Objectives

The objectives of this research are as following:

i. To alter hierarchical Axis-Aligned BSP (kD-tree) tree to adapt occlusion culling of dynamic objects in indoors scene.

ii. To minimize updating the kD-tree structure for every dynamic object at each frame except when the dynamic objects are visible.

iii. To develop a method that will be able to handle multiple dynamic objects in the scene, while maintaining the image quality and frame rate.

1.5 Scopes

The scope of this research are as following:

i. Only indoor scenes will be used to test the algorithm.

ii. Maximum number of dynamic objects in the scene will be limited to 10 to simplify the scene.

iii. The proposed algorithm will be implemented with David P.Luebke and Chris Georges [18] algorithm. It is an intuitive algorithm ideally suited for visualization of indoor scenes such as those found in architectural models.

iv. kD -tree [11] will be used as hierarchy data structure for the scene.

v. This algorithm is hardware independent and focused in increasing the performance in handling dynamic object.

vi. The algorithm is based on the observation that the possible movements of dynamic objects may be subject to some known physical constraint, which might be known from physical simulation or by a user interface.

vii. Efficiency of algorithms will be measured frame per second quantitatively by using fixed amount of polygon in the scene.

1.6 Contributions

As discussed in the research motivation and problem statement, since most of the spatial hierarchical data structures are hard to update in real-time, they are unsuitable for dynamic scenes occlusion culling. Therefore, a technique is proposed in the thesis to adapt dynamic objects in current spatial hierarchical data structure. Stated here are the research contributions from this algorithm:

i. kD-tree can adapt and handle static and dynamic objects in the same spatial hierarchical tree.

ii. Updating the trees can be minimized in each frame for dynamic objects, thus increasing the application's performance.

iii. Dynamic objects are treated in different manner but still using the same occlusion culling algorithm as the static scenes. Therefore, there is no need to develop entirely different occlusion culling technique for dynamic objects.

iv. Using the proposed technique, more efficient application handling dynamic objects in real-time can be developed. Frame rate can be maintained since dynamic objects are also are culled.

The algorithm that proposed in this research suitable for the applications that emphasize on the performance for dynamic virtual indoor environments. Stated here are the applications that will benefit from this research:

i. Game
First and third person shooter game requires large dataset of polygon to represent the indoor scenes. Since most of the domestic computer capability is limited, there is a need to optimise the game to achieve playable performance to run the game. Therefore this algorithm can be used to increase the speed of the game together with other optimisation.

ii. Simulation Environment

This algorithm also can be implemented in simulations of indoor environment such as virtual reality applications while saving computations for other usage.

iii. Film Industry

Rendering models in an animation is a time consuming task. Rendering the environments can be faster by implementing this algorithm for the indoor scenes or closely wrapped scenes.

1.7 Organization of Thesis

This thesis consists of six chapters as described below:

i. Chapter 1 briefly introduces the culling methods and their research background. Current issues and problem statement are also defined. Objectives and scopes of research are stated clearly. Finally, research contributions are discussed.

ii. Chapter 2 discusses in details all the techniques available for occlusion culling and categorized them accordingly. Beside those, spatial hierarchical trees are also discussed. Comparison for occlusion culling techniques was done to reveal pros and cons of each technique reviewed.

iii. Chapter 3 explains the research methodologies that have been applied. Algorithms used for this method are stated clearly.

iv. Chapter 4 describes how DBV culling is implemented in rendering engine. Beside those, engine specifications and hardware and software requirements are stated.

v. Chapter 5 shows the test results and analysis for each test. Using three tests for each scene uses four types of scene to test the algorithm.

vi. Chapter 6 summarizes this research and states future works that can be done.

REFERENCES

1. Appel, A. Some techniques for shading machine renderings of solids. *AFIPS 1968 Spring Joint Computer Conf.*, volume 32, 1968.
2. Sutherland, E., Sproull, R. F and Schumaker, R. A. A characterization of the hidden surface algorithms. *ACM Computer Surveys*, March 1974.
3. Coorg, S. and Teller, S. Real-time occlusion culling for models with large occluders. *Proceedings of the 1997 Symposium on Interactive 3D Graphics, Providence, Rhode Island, ACM SIGGRAPH,* Apr. 1997
4. Coden, D., Chrysanthou, Y., Silva, C. T., Durand, F. A. Survey of Visibility for Walkthrough Applications. *ACM Computer Surveys,* 2000.
5. Foley, J. D., Dam, A. V., Feiner, S. K., and Hughes, J. F. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1990.
6. Sudarsky, O. and Gotsman, C. Dynamic scene occlusion culling. *IEEE Transactions on Visualization and Computer Graphics,* January 1999.
7. Sudarsky, O. *Dynamic Scene Occlusion Culling*. Phd Thesis, Israel Institute of Technology, January 1998.
8. Myllarniemi, V. Dynamic Scene Occlusion Culling, *Computer Graphic Seminar Spring*, Helsinki University of Technology, Spring 2003.
9. Saona-Vazquez, C., Navazo, I. and Brunet, P. *The visibility octree. a data structure for 3D navigation*, 1999
10. Greene, N., Kass, M. and Miller, G. Hierarchical z-buffer visibility. *Proceedings of SIGGRAPH 93,* 1993.
11. Fuchs, H., Zvi Kedem, and Naylor, B. F. On visible surface generation by a priori tree structures. *Computer Graphics, ACM SIGGRAPH,* July 1980.
12. Naylor, B. F. Partitioning Tree Image Representation and Generation from 3D Geometric Models. *ACM SIGGRAPH,* 1994.
13. Jones, C. B. A new approach to the 'hidden line' problem. *Computer Journal,* 14(3), August 1971.

14. Clark, J. H. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM,* October 1976.

15. Airey, J. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*. PhD thesis, University of North Carolina, Chappel Hill, 1991.

16. Teller, S. J. and Sequin, C. H. Visibility preprocessing for interactive walkthroughs. *Computer Graphics (Proceedings of SIGGRAPH 91),* 25(4):61–69, July 1991.

17. Plantinga, H. Conservative visibility preprocessing for efficient walkthroughs of 3D scenes. In *Proceedings of Graphics Interface '93*, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

18. Luebke, D and Georges, C. Portals and mirrors: Simple, fast evaluation of potentially visible sets. *In Pat Hanrahan and JimWinget, editors, 1995 Symposium on Interactive 3D Graphics*, pages 105–106. ACM SIGGRAPH, April 1995.

19. Hudson, T, Manocha, D., Cohen. J., Ming Lin, Hoff, K. and Zhang, H. Accelerated occlusion culling using shadow frusta. In *Proceesings of ACM Symposium on Computational Geometry*, 1997.

20. Bittner, J., Havran, V. and Slavik, P. Hierarchical visibility culling with occlusion trees. In *Proceedings of Computer Graphics International '98,* June 1998.

21. Greene, N. *Hierarchical Rendering of Complex Environments*. PhD thesis, University of California at Santa Cruz, June 1995.

22. Zhang, H., Manocha, D., Hudson, T., and Hoff III, K. E. Visibility culling using hierarchical occlusion maps. In *SIGGRAPH '97 Conference Proceedings*, pages 77–88, Los Angeles.

23. Durand, F., Drettakis G., Jo˜elle Thollot, and Puech, C. Conservative visibility preprocessing using extended projections. *Proceedings of SIGGRAPH 2000*, July 2000.

24. Ivan, E. Sutherland, Sproull, R. F., and Schumacker, R. A. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys*, March 1974.

25. Durand, F. 3D Visibility: Analitical Study and Applications. *Siggraph'2000 course notes on Visibility*. 2000.

26. Airey, J. M., Rohlf, J. H. and Brooks F. P. Jr. Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, March 1990.

27. Greene, N. and Kass, M. Error-bounded antialiased rendering of complex environments. In *SIGGRAPH ’94 Conference Proceedings*, pages 59–66, Orlando, Florida, July 1994. *ACM Computer Graphics*, vol. 28, no. 4.

28. Klosowski, J T., Held, M., Joseph S. B., Sowizral, H. and Zikan, K. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, January- March 1998.

29. Severson, K. Visualize *Workstation Graphics for Windows NT*. HP product literature.

30. Bartz, D., Meiner, M. and Httner, T. Opengl-assisted occlusion culling for large polygonal models. *Computer & Graphics*, 23(5):667–679, 1999.

31. Bartz, D., Meiner, M. and Httner, T. Extending graphics hardware for occlusion queries in opengl. In *Proc. Workshop on Graphics Hardware ’98*, pages 97–104, 1998.

32. Rubin, S. and Whitted, T. A 3-dimensional representation for fast rendering of complex scenes. *Computer Graphics (Proceedings of SIGGRAPH 80)*, 14(3):110–116, July 1980.

33. Glassner, Andrew. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 1984.

34. Batagelo, H. C. and Shin-Ting Wu. Dynamic Scene Occlusion Culling Using a Regular Grid. *XV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPH’01)*, October 2002.

35. Helin, V. Hierarchies for Occlusion Culling. *Seminar on Computer Graphic, Helsinki University of Technology*. Spring 2003.

36. Jevans, D. and Wyvill, B. Adaptive voxel subdivision for ray tracing. In *Proceedings of Graphics Interface ’89*, pages 164–172, June 1989.

37. Cazals, F., Drettakis, G. and Puech, C. Filtering, clustering and hierarchy construction: a new solution for ray-tracing complex scenes. *Computer Graphics Forum*, 14(3):371–382, August 1995. ISSN 1067-7055.

38. Bentley, J. L. Multidimensional Binary Search Trees Used for Associative Searching. *Communication of the ACM*, 18(9), September 1975.

39. Agarwal, P. K., Arge, L. and Vitter, J. S. Bkd-tree: A Dynamic Scalable kd-tree. *Proceedings of SSTD'03*, 2003

40. Koenderink J. J. and Doorn A. J. van: The internal representation of solid shape with respect to vision. *Biol. Cybernet.* 32, 211-216.

41. Schrocker, G. *Visibility Culling for Game Application*. Thesis. Institute for Computer Graphics and Vision, Graz University of Technology, April, 2001.

42. Ranta-Eskola, S. *Binary Space Partioning Trees and Polygon Removal in Real Time 3D Rendering.* PhD. Thesis. Computing Science Department, Uppsala University, 2001

43. Baumker, A. and Dittrich, W. Fully dynamic search trees for an extension of the BSP model. *Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, 233 – 242, 1996

44. Earnshaw, R. A., Chilton, N. and Palmer I. J. Visualization and virtual reality on the Internet. In *Proceedings of the Visualization Conference*, Jerusalem, Israel, Nov 1995.