Improved View Frustum Culling Technique for Real-Time Virtual Heritage Application

Mohd Shahrizal Sunar [1], Abdullah Mohd Zin [2] and Tengku Mohd Tengku Sembok [2]

[1] *Department of Computer Graphics and Multimedia, Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia.*
[2] *Faculty of Information Science and Technology, National University of Malaysia.*

Abstract—**Real-time virtual heritage application is normally executed with a high performance computer system. This is due to the complexity and highly computational cost which makes such system impossible to run in a lower specification computer system. Therefore, many developed virtual heritage application cannot be retrieved by the target users. One of the factors that contribute to the high computational cost is 3D graphics processing. In this paper, a new improved technique is developed to reduce the computational cost of virtual heritage application. This technique is named as Range Detection Technique (RDT). It is based on the View Frustum Culling (VFC) method. The conventional VFC method tested the intersection of six planes using the plane equation to determine the visibility scope. Unlike conventional VFC technique, RDT is based on camera referential points and test the 3D objects whether they are in the viewing range or not. RDT execute the testing with the combination of bounding volume. Based on the testing result, a virtual heritage application with RDT can be executed with higher Frame-Per-Second (FPS) rate. The time taken to complete a set of fixed testing path is shorter. The result shows that RDT is able to increase the rendering performance and reduce the computational cost of virtual heritage application without sacrificing the visual quality Using data taken from the Ancient Malacca project, this research indirectly helps to reduce the computational cost of the former, which previously can only run with the high performance computer system..**

Index Terms—**Range detection, view frustum culling, visibility algorithm, virtual heritage, computer graphics.**

I. INTRODUCTION

The key factors of real-time computer graphics application such as virtual walkthrough, flight simulator and games are realism and interaction between system and the user. In most real-time computer graphics application, maintaining the interactive frame rate and smooth movement is utmost important. This is important to give the user immersive experience during the navigation in the virtual world. Realism as define in [1] means how far the image can visualise in the users mental can be same as real experience in the real world. To generate the realistic 3D world, the geometric complexity level of 3D model has to be increased [2, 3]. Another need is for beautiful textured that will increase the memory usage simultaneously [4].

There is a trade-off between realism and real-time [5] and also between visual quality and rendering performance. Increasing realism will slower the processing speed. Therefore, when speed is more crucial, we may have to sacrifice realism to some level. Most of virtual reality application needs more real-time than realism. This is because lots of interaction by user is needed to navigate the application. But, realism still can be achieved without sacrificing too many realism needs [6].

Visibility culling is one of the method that can decrease the high computational of real-time computer graphics application. View frustum culling (VFC) [7] is the simplest visibility culling among the others method i.e. back faces culling [8] and occlusion culling [9, 10, 11]. Only the objects which are inside the frustum will be displayed. In this paper, we will focus on the construction of alternative view frustum algorithm for effective rendering. We present and examine a new improved view frustum culling approach by [12]. We named this approach as range detection approach which effectively eliminates the unseen objects in the virtual environment.

The purpose of this work is to develop speed-up techniques for Ancient Malacca Virtual Walkthrough project. The Ancient Malacca Virtual Walkthrough is a project that focuses on the modeling and visualization of Malacca city in 15th century. It is based on local and foreign sources, such as the Sejarah Melayu [13] and the descriptions by Portugese writer, Pires (1944) described the city and the empire as an opulent and prosperous centre of maritime Malay civilization. As a maritime empire, trading and commercial activities, both local and foreign, became the mainstay and the backbone of her economy. The focus area of visualization is Central Business and Administrative District of the Malacca Empire.

In the next section, we will describe the related work of the view frustum culling algorithm and some similar virtual heritage projects. In Section Ⅲ, view frustum culling definition is explained in detail. We illustrate the normal six plane view frustum culling here. Then we enlighten the range detection approach in the next section. In Section Ⅴ, we show some statistical representation to portray our results. We end up the paper with the conclusion and some suggestion room of improvement.

II. RELATED WORKS

Visibility is a wide research topic area in computer graphics. It has been studied since early era of computer graphics. The original idea on how to make the view frustum culling efficient is invent by [14] who used hierarchical approach for visible

Manuscript received on April 9th, 2008.
E-Mail: shah@fsksm.utm.my.

surface. When the area of computer graphics becomes more popular in 1990's, the algorithms also start growing [15]. We focus here on the evolution of view frustum culling techniques.

View frustum culling algorithm can be divided into two common approaches. The first approach is to transform the bounding volume into perspective view. The testing is performed with perspective coordinate system [16]. Second approach is by checking the bounding volume against view frustum volume illustrated by it six planes [17, 18].

The advantage of this approach is that we can make the trivial acceptance and rejection. Then the test recursively continued based on intersection between a box and a frustum. [18] shows how to detect the intersection of a rectangular and a convex polyhedron. [18] also introduced the fast intersection method between polygon and cube. A view frustum culling using probabilistic caching scheme is developed by [19]. The algorithm is implemented using Binary Space Partition tree. Statistical probability representation is used to get faster result than the hierarchical bounding box scheme. The method produced unacceptable errors in some cases.

Bittner had improved the view frustum culling base on traversal coherency of the scene hierarchy. Certain interior nodes of the hierarchy is avoids during the intersection test. This technique not doing the test to the object that is expected to remain visible [20].

Optimized bounding boxes for view frustum culling are explored by [21]. They used low degree of freedom of camera motion in the user-driven walkthrough scene. [21] and [19] algorithm are similar. Both algorithms try to minimize the work by caching information and avoiding more expensive intersection tests.

The VFC algorithms were improved by [12] who use slightly different way of defining the view frustum volume. In this approach, the view frustum volume is identified based on camera referential point and properties.

III. ANALYSIS OF VIEW FRUSTUM CULLING TECHNIQUE

View frustum can be imagined as a truncated pyramid representing user's scope of visible view. Essentially it is the field of view of the virtual camera which determines the region of virtual environment that may appear on the screen during run-time.

A view frustum is defined by six planes [22]:

$$\pi_i : n_i \cdot x + d_i = 0 \tag{1}$$

i = 0...5, where n_i is the normal and d_i is the offset of plane π_i, and x is an arbitrary point on the plane. We say that a point x is outside a plane π_i if $n_i \cdot x + d_i > 0$. If the point is inside all planes then the point is inside the view frustum.

To generate a view frustum, a pyramid with surfaces is used. There are two surfaces that represent the nearest and furthers viewpoint. The nearest and furthers representing the boundary of visible objects. Another four surfaces are representing top, bottom, left and right boundary. View frustum in our case is always set in the perspective view mode.

View frustum culling (VFC) is important because it cull away the invisible (unseen) objects in complex scenes from the rendering pipeline. It save the memory usage and it is definitely will increase the frame rates. View frustum culling is typically used in virtual reality software, walkthrough system, 3D games and serious visualization such as medical and military simulation for its efficient rendering.

Each frustum plane is tested if the object is inside or outside the view frustum. Only primitives that are totally or partially inside the view frustum need to be rendered.

Three possible output states for bounding box and view frustum test are:

OUTSIDE:	All eight points representing the bounding box are totally outside the view frustum. So the object is eliminated from the further processing.
INSIDE:	All eight points representing the bounding box are totally inside the view frustum. So there is no further culling calculation. The object is included for the further processing. This state is useful for the application to avoid unnecessary computation.
INTERSECT:	Any point of bounding box is inside the view frustum. So down traverse the hierarchy is required. Partial of object geometry is included for the further processing. This state will employ expensive cost of computation. For speed up reason, we consider this state as INSIDE.

IV. RANGE DETECTION TECHNIQUE

Conventional view frustum culling test discussed in section III was based on the idea that the volume of view frustum is surrounded by six planes. Testing is performed against every single plane to detect the bordered objects for viewing using plane equation. Each object is compared six times in each frame to resolve the visibility.

We introduce an alternative approach which improved the testing method by Placeres [23]. In [23], the VFC test is done to all point in the virtual environment. The improved approach is based on formula given in (2).

$$n(E_j) = N - \bigcup_{i=m \cap j^c}^{m} n(E_i) \tag{2}$$

n(E_j) = number of objects in group j which is the number of object we sent to the frame buffer; *N* = total of objects in the virtual walkthrough application; *j* = index of group; *i* = counter number of object in *N* but not in group *j*; m = number of *N* but not in group *j*; j^c = complement of *j*. In this section, we will describe the advantage of this approach. We name our method as Range Detection Technique (RDT).

The RDT approach test is based on three steps as follow: 1. Front object test, 2. Sphere bounding test, 3. AABB bounding test. The front object test is done to eliminate the objects that not facing the camera view. The step is arranged based on the fact that sphere testing is faster than AABB. Every village houses and trees in the scene is bounded by sphere and AABB.

Sphere bounding is generated based on center points and radius of each object in the application. To generate an AABB, the minimum and maximum point information for each object is needed. Then, based on the minimum and maximum point will generate a box with another six remaining points. Therefore for each object, eight points defining the AABB is tested to the range of the camera visible boundary. In the front object test, only object A is eliminated. After the sphere test is done using RDT, objects D, E and F are to be saved in the frame buffer. Only objects D and F are to be displayed as output when the AABB test is finished.

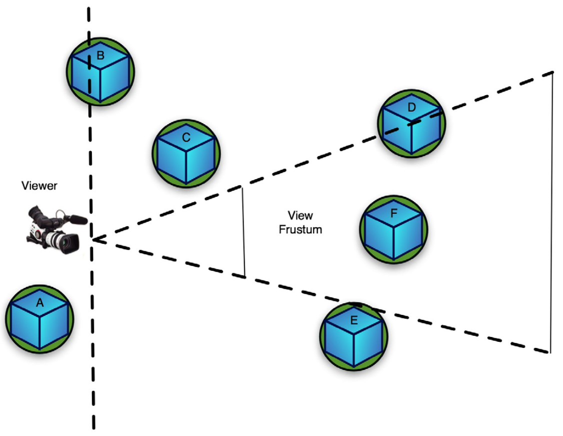


Fig. 1. This figure shows six objects in virtual environment that bounded by sphere and AABB.

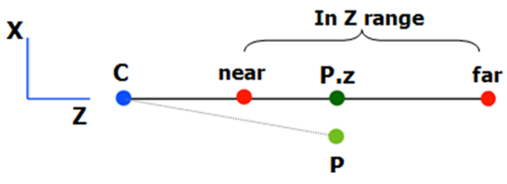


Fig. 2. This figure shows the range for point P in z coordinate.

Next, we explain the step taken to extract and determine the point in visibility range. As shown in Fig. 2, P is the point that is been tested against the view frustum. The virtual camera properties such as location, lookup and orientation are used to check the distance between point P and camera. The camera reference is based on three unit vector X, Y and Z which labeled as C.

First, we checked the z coordinate of P. Consider C is the point in camera orientation coordinates. To find z coordinate of P, we need to find the vector that goes from c to P. The length of projection of this vector on z must be computed. This can be done by dot product as z is assumed as a unit vector.

$$v = P - c \tag{3}$$

$$P \cdot z = v \cdot z \tag{4}$$

If the z coordinate is between the range of near and far then P is possibly inside the view frustum, then the coordinates x and y must be test. Otherwise, P is absolutely out of the view frustum.

$$near < P \cdot z < far \tag{5}$$

To find x and y coordinate of P in camera orientation coordinate, the similar procedure is followed.

$$P \cdot y = v \cdot y \tag{6}$$

$$P \cdot x = v \cdot x \tag{7}$$

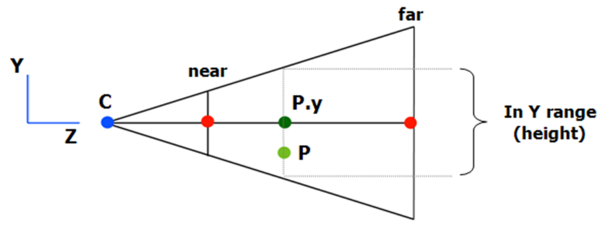


Fig. 3. Next, we need to check the y coordinate of P. From the z coordinate of P, we can find the height range of view frustum.

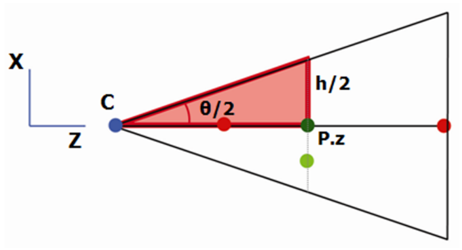


Fig. 4. Therefore, P.y will be compared to the height to test whether it is in view frustum range or not.

To get height range of view frustum, the following trigonometry formula is used.

$$h = P.z * 2 * \tan\frac{\theta}{2} \tag{8}$$

P is in the view frustum range if P.y value is larger than -h/2 and smaller than h/2.

$$-\frac{h}{2} < P.y < \frac{h}{2} \tag{9}$$

Then, we will test the x coordinate of P easily by checking P.x range of view frustum width. Width, w is computed based on the height and aspect ratio. Aspect ratio is the number of pixels horizontal divided by the number of pixels that can be displayed by the system.

$$w = h * aspectratio \tag{10}$$

V. RESULT

This section describes the implementation and testing results of both normal six plane VFC, Placeres's and RDT in Ancient Malacca Virtual Walkthrough project.

In this project, we develop our virtual heritage environment using C++, OpenGL, GLVU and GLUI. The 3D models of Ancient Malacca are modeled by 3D Studio Max, which is modified from previous project that run at SGI Onyx machine. We used Intel Pentium 4 3.2 GHz with 512MB RAM and nVidia GeForce FX5950 graphics display card to test the VFC in our virtual walkthrough application.

For testing purpose, we fixed the camera movement path in our virtual environment. This is important to ensure the camera to follows the same path for every testing with different culling method.

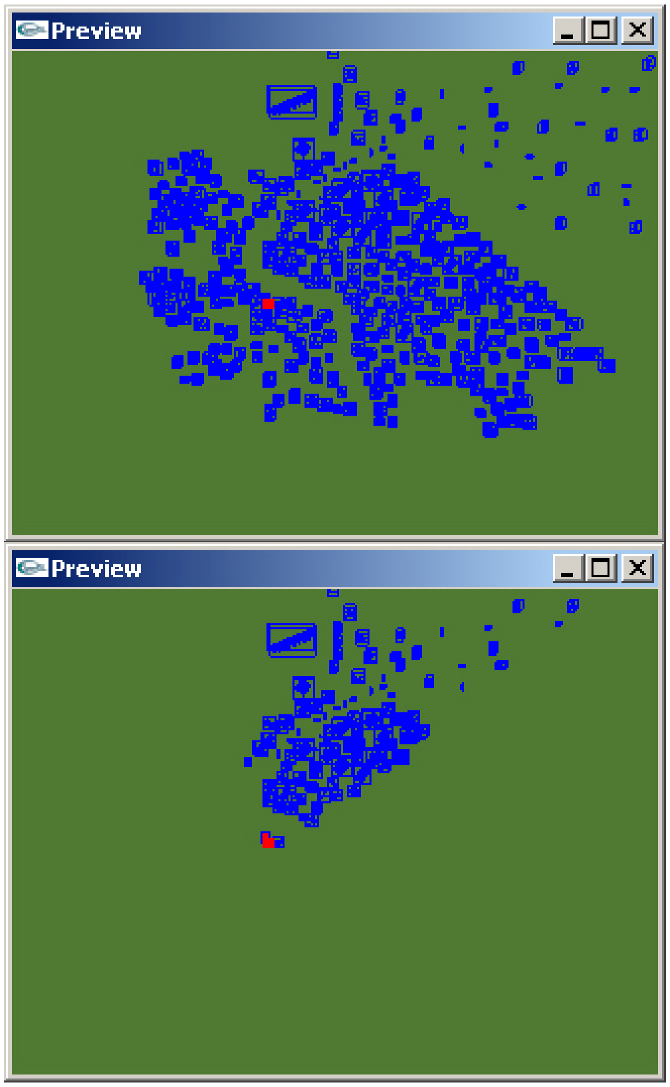
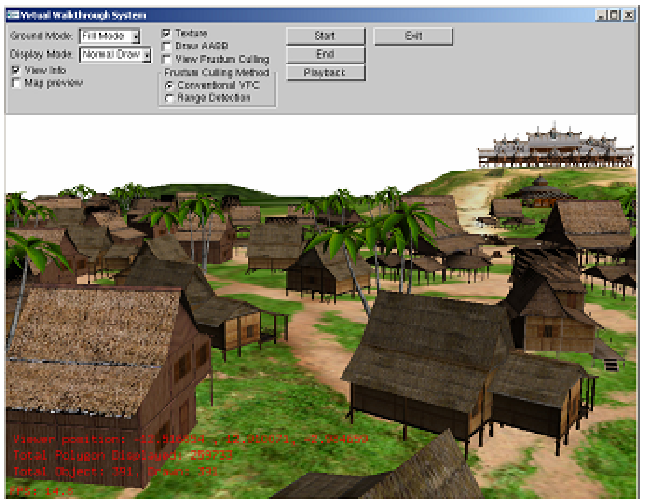


Fig. 5. This figure shows the difference virtual walkthrough executed between with (down) and without (top) RDT.

Fig. 5 shows the different base on the number of object drawn in the screen with and without RDT. All object in the virtual walkthrough is appears in blue. Red dot in the figure represent the camera position. As shown in the top figure, all objects are drawn without the implementation of RDT. This may consume a high computational cost. The virtual walkthrough will reduce more than three quarter number of object to be sent to the frame buffer with the implementation of RDT. This can be shown in the Fig. 5 (down).

The frame per second test is done to the Ancient Malacca virtual walkthrough. The purpose of the test is to measure the performance and the interactivity of the virtual walkthrough system with and without RDT. Instead of that, it also shows the different of smoothness during application run-time with RDT. Fig. 7 shows the result of frame per second test. The frame rates become higher when the camera reaches to the area with not many objects to be rendered. When, the camera arrived at the area with crowd of objects, the frame rates will decrease. The result shows that the RDT give the highest frame rates among others during the run-time test. Without VFC, the frame rates remain 23 and 24 frames per seconds, which is under the expectation of real-time rendering standard. This is shown in as a vertical line across at the bottom of the graph.



Viewer position: -12.916454 , 12.910071, -2.994650
Total Polygon Displayed: 259733
Total Object: 391, Drawn: 391
FPS: 14.8

Fig. 6. This figure shows the screen shot of the Ancient Malacca Virtual Heritage application that implemented the RDT to effectively determine the potential visible objects.

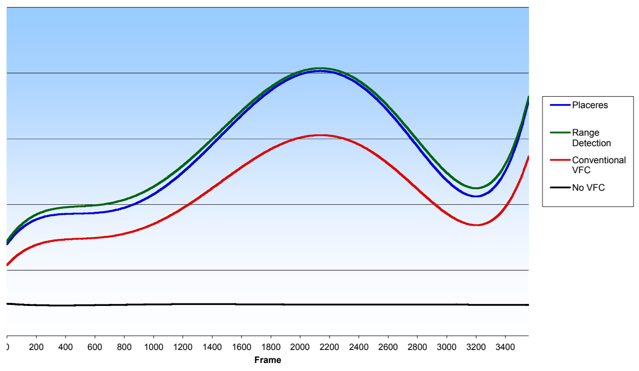


Fig. 7. This figure shows the result of frame per second for each normal conventional VFC, Placeres [23], RDT and without VFC.

We did five times testing for each approach including with no view frustum culling. Then we take the average of the result as shown in the bar chart. We can notify that the RDT is the fastest among others.

RDT was also tested with different categories of computer system. Fig. 9 shows the result of frame per second (fps) for RDT running on different specification category of computer systems. Category 1 is the highest specification which can run the virtual heritage application in highest fps. This result also shows that RDT can improve the application speed although it was running on the lower specification like in category 3. The specification category of computer systems used for testing in this research as below:

Category 1:
Intel Pentium 4 3.2 GHz (HT), 2046 MB RAM, nVidia

GeForce FX5950U with 256MB VRAM.

Category 2:
Intel Pentium 4 3.0 GHz, 1024 MB RAM, nVidia GeForce FX5200 with 128MB VRAM.

Category 3:
Intel Pentium 4 1.9 GHz, 512 MB RAM, ATI Radeon 7500 VRAM.

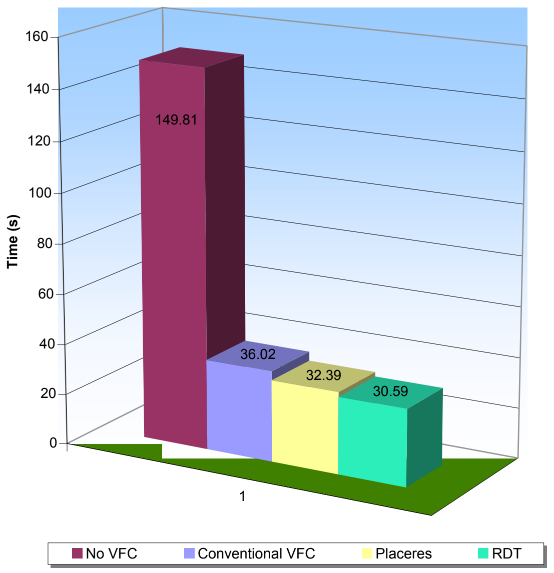


Fig. 8. This figure shows the advantage of RDT is the time taken to finish one round of the testing path is shorter than normal VFC.

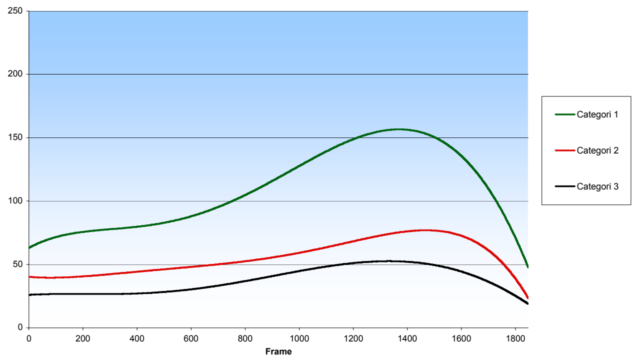


Fig. 9. This figure shows the result of frame per second for RDT running on different system category.

VI. CONCLUSION

We present a method that can improve view frustum culling and effectively eliminates the unseen objects in Ancient Malacca Virtual Walkthrough in this paper. The main contribution of this paper is the implementation, testing and comparison of conventional six plane VFC, Placeres's and RDT approach. The result show that we can accelerate the performance of walkthrough system by using six plane checking method in VFC approach. Placeres had proposed an outstanding alternative approach in determining view frustum. We put an added value on the testing method which slightly improves the virtual walkthrough performance. In future, we would like to incorporate with crowd of characters. In another word the technique will also to be tested to improve dynamic virtual environment.

ACKNOWLEDGEMENT

We would like express our appreciation to Malaysian Ministry of Science, Technology and Innovation under eScienceFund grant (01-01-06-SF0472) for financial support of this research. We also like to thanks Creative Application Development Centre (CADC), Multimedia Development Corporation, Cyberjaya, Malaysia for allowing us to use the 3D model taken from Ancient Malacca project and permit us to explore the potential improvement of the project.

REFERENCES

[1] K. Chiu and P. Shirley. Rendering, Complexity, and Perception, in proceedings of the *5th Eurographics Rendering Workshop (Darmstadt), Darmstadt*, pp. 19-34, June 1994.
[2] P. Dzwig. Complex Scene Generation, in *IEE Colloquium on Practical Applications of Parallel Signal Processing. London, UK: IEE*, pp. 1-7, November 1988.
[3] M. Ramasubramanian, S. N. Pattanaik and D. P. Greenberg. A Perceptually Based Physical Error Metric for Realistic Image Synthesis, in *SIGGRAPH '99: proceedings of the 26th annual conference on Computer graphics and interactive techniques. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.*, pp. 73-82, 1999.
[4] J. M. Airey, J. H. Rohlf and F. P. Brooks. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments, vol. 24, no. 2, pp. 41-50, March 1990.
[5] A. Angelidis and G. Fouquier. Visualization Issues in Virtual Environments: from Computer Graphics Techniques to Intentional Visualization, in *WSCG'2001*, pp. 90-98, February 2001.
[6] D. P. Luebke. A Developer's Survey of Polygonal Simplification Algorithms, *IEEE Computer Graphics Application*, vol. 21, no. 3, pp. 24-35, 2001.
[7] T. Möller and E. Haines. Real-time Rendering, Natick, MA, USA: A. K. Peters, Ltd., 2002.
[8] H. Zhang and K. E. Hoff. Fast Backface Culling Using Normal Masks, in *Symposium on Interactive 3D Graphics*, pp. 103-106, 1997.
[9] D. Bartz, M. Meißner and T. Huttner. OpenGL-assisted Occlusion Culling for Large Polygonal Models, *Computers and Graphics*, vol. 23, no. 5, pp. 667-679, 1999.
[10] T. Aila. Efficient Algorithms for Occlusion Culling and Shadows, Ph. D. dissertation, Helsinki University of Technology, Helsinki, Finland, February 2005.
[11] H. Zhang. Effective Occlusion Culling for the Interactive Display of Arbitrary Models, Ph. D. dissertation, University of North Carolina, Chapel Hill, 1998.
[12] F. P. Placeres. Improved Frustum Culling, in *Game Programming Gems 5*, K. Pallister, Ed. Charles River Media, February 2005.
[13] A. S. Ahmad. Sulatus Sulatin (Sejarah Melayu), Dewan Bahasa dan Pustaka, 1979.
[14] J. H. Clark. Hierarchical Geometric Models for Visible Surface Algorithms, *Communications of the ACM*, vol. 19, no. 10, pp. 547-554, 1976.
[15] D. Cohen-Or, Y. Chrysanthou, C. Silva and F. Durand. A Survey of Visibility for Walkthrough Applications, *Course 30, SIGGRAPH Course Notes*, 2003.
[16] L. Bishop, D. Eberly, T. Whitted, M. Finch and M. Shantz. Designing a PC Game Engine, *Computer Graphics in Entertainment*, pp. 46-53, January/February 1998.
[17] K. E. Hoff. A "fast" Method for Culling of Orientedbounding Boxes (obbs) Against a Perspective Viewing Frustum in Large "walkthrough" Models. [Online]. Available: http://www.cs.unc.edu/hoff/research/vfculler/viewcull.html, May 1996.

[18] N. Greene, Detecting Intersection of a Rectangular Solid and a Convex Polyhedron, San Diego, CA, USA: Academic Press Professional, Inc., 1994, pp. 74-82.
[19] M. Slater and Y. Chrysanthou. View Volume Culling Using a Probabilistic Caching Scheme, in *VRST '97: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 1997, ACM Press.
[20] J. Bittner, V. Havran and P. Slavik. Hierarchical Visibility Culling with Occlusion trees, in proceedings of *Computer Graphics International*, pp. 207-219, June 1998.
[21] U. Assarsson. View Frustum Culling and Animated Ray Tracing: Improvements and Methodological Considerations, Tesis Ph.D. Chalmers University of Technology, Goteborg, Sweeden. 2001.
[22] U. Assarsson and T. Möller. Optimized View Frustum Culling Algorithms, *Chalmers University of Technology*, Tech. Rep. 99-3, March 1999.
[23] F. P. Placeres. Improved Frustum Culling, in Pallister, K. (Ed.). Game Programming Gems 5, pp. 65-77. Masssachusetts: Charles River Media. 2005.

Mohd Shahrizal Sunar received the BSc degree in Computer Science majoring in Computer Graphics (1999) from Universiti Teknologi Malaysia and MSc in Computer Graphics and Virtual Environment (2001) from The University of Hull, UK. In 2008, he obtained his PhD from National University of Malaysia. His major field of study is real-time and interactive computer graphics and virtual reality. He is a faculty member at Department of Computer Graphics and Multimedia, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia since 1999. He had published numerous articles in international as well as national journals, conference proceedings and technical papers including article in magazines. Dr. Shahrizal is an active profesional member of ACM SIGGRAPH KL-Chapter. He is also a member Malaysian Society of Mathematic and Science.

Abdullah Mohd Zin is a full professor at the Department of Industrial Computing, Faculty of Information Science and Technology, National University of Malaysia. He received his BSc in Computer Science from Southampton, UK and MSc from University of Wales. He obtained his PhD (Computer Science) from the Nottingham University UK. He has published numerous articles in international peer-reviewed journals. His research interest includes foundation of programming, collaborative programming learning tools, software management and computer graphics. Currently, Prof Abdullah is the deputy dean as well as the head of programming research group at the faculty.

Tengku Mohd Tengku Sembok received his BSc in Computer Science (1977) from Brighton, UK, MS degree in Computer Science (1981) from University of Iowa, USA and PhD in Computer Science University of Glasgow (1989). He is a full professor of Computer Science at National University of Malaysia. He joined National University of Malaysia since 1978 and holds many important positions to the faculty as well as university. His research interest includes intelligent information system, information retrieval, multimedia courseware, and knowledge management. He had published more than 100 papers on international journals, national journals and international conferences. Prof. Tengku Mohd has over twenty years of experience in various fields of Information Technology and has managed numerous R&D and consultancy projects successfully. He is currently holding a chair of professor in Computer Science in Universiti Kebangsaan Malaysia and has held several academic posts; among them are Head of Computer Science Department and the Dean of Faculty of Information Science and Technology. He involves in numerous profesional board including IEEE and Malaysia Artificial Intelligent Society. He sits in various national committees on IT such as Terengganu State Steering Committee on IT, IRPA Steering Committee in Services Sector (MOSTE), and Curriculum Development Committee for Computing Subject (Malaysia Examination Council).