REAL-TIME SHADOW USING A COMBINATION OF STENCIL AND THE Z-BUFFER

Hoshang Kolivand[1*], Mohd Shahrizal Sunar[2], Normal Mat Jusoh[3],
Folorunso Olufemi A.[4]

[1,2,3,4]UTM ViCubelab, Department of Computer Graphics and Multimedia, Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia,81310 Skudai Johor Malaysia
[1]Islamic Azad University, Firozkoh Branch, Iran
[1]shahinkey@yahoo.com
[2]shah@cs.utm.my
[3]normal@utm.my
[4]femi_folorunso@yahoo.com

ABSTRACT

In this paper we describe a real-time shadow generation with volume shadow algorithm in virtual environment that is illuminated by light sources with possibility to move separately. This algorithm uses the combination of stencil and Z-buffers to generate shadow volume. It is simple to understand and implement. We have significantly improved and implemented recent techniques that are used in shadow volume algorithms using stencil buffers especially in order to recognize silhouette, reduce the number of shadow polygons and also redundant length of each triangles that makes the volume shadow. This work may be applied in commercial games or other virtual reality systems.

KEYWORD

Real-time Shadow, Volume Shadow, Stencil Buffer, Depth-Buffer.

1. INTRODUCTION

Recent advances in games are geared towards realism but the introduction of shadows plays very important role in this direction. Although shadows makes game more realistic it is often very difficult to implement in the virtual environments, specifically in real-time games. In computer games, shadows give the gamers the feelings that they are playing in realistic world and so they can enjoy the games as much as possible. A game without shadow effect cannot be so interesting and enjoyable, especially in this century that gamers' imagination request more realistic situation when they are watching cartoons or playing games.

There are some methods to create shadow and some tools that are needed to be introduced before starting to make real-time shadows. Z-buffer is most important equipment that programming languages provide to create real-time shadow for virtual environments. The Z-buffer visible surface algorithm, first used by Catmull [1], He propose the first algorithm to create shaded images of bicubic surface patches. His algorithm is not entirely private and was too difficult to implement as it requires huge amount of memory.

The other buffer that is used in this paper is the Stencil buffer. This kind of buffer is used to limit the area that we need to have shadow. In the other words, stencil buffer can help us limit the rendering scene. One of the advanced usages of the stencil buffer is the time that we need to use stencil buffer with Z-buffer. For example, stencil values can be automatically increased or decreased for every pixel whose depth test either fail or pass that we will explain latter in this paper.

In computer graphics, shadows re made up of many polygons patched together. Each polygon is a converted face of the original model, alongside the rays that started from light source onto the receiver. This technique is only reasonable just for flat surfaces (like wall or on a table) but it is useless when we want to have shadow on non-flat surface such as cube, torus or sphere. The other limitation is that this technique doesn't have a self-shadow (shadow of one part of object onto the other part). Volume shadow using stencil buffer is therefore a good method to detect the other object behind the occluder form the light source.

Combination of depth test and stencil buffer makes a substantial number of effects possible such as real-time shadows that often require several rendering passes and, as a result of that, it can put a heavy load on the graphics hardware. But in this paper we are going to discuss some calculations in the hardware and software to improve the combined algorithm of stencilling and depth buffering. The other aim of this paper is to reduce the number of steps required to implement volume shadow as a combination of stencil and depth buffers.

2. PREVIOUS WORK

In 1997 Frank Crow [7] introduced the main idea of shadow volume and published his ray-casting paper based on shadow volume algorithm. His method explicitly clips shadow geometry to the view frustum. In 1991 Heidmann published a paper base on volume shadow using stencil buffer which is the main idea behind shadow volume algorithm [2]. Figure 1 illustrates the algorithm of stencil shadow volume which he proposed.

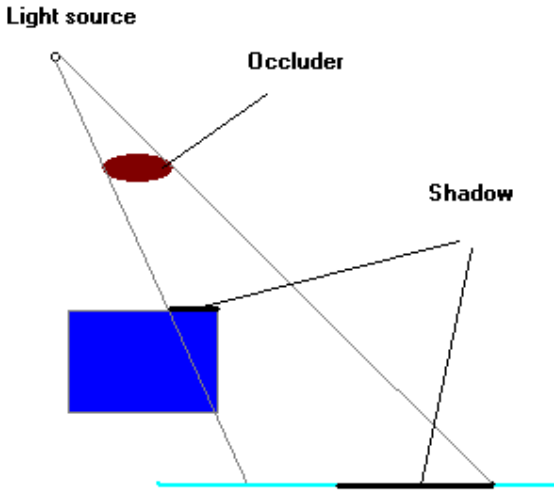

Figure 1: Stencil shadow

The other algorithm that Carmack suggested in year 2000 was a bit different from the previous algorithms it includes rays that are traced from infinity onto the eye and stops when it hits the pixel that is nearest to the eye [19]. Shadows of self-shadowing use in some situation such as hair, fur and smoke [21]. Shadow Maps suggested by [27] which are an extension to the traditional

shadow mapping techniques. In year 2002 Lengyel propose a hybrid algorithm that uses Z-fail rendering when point of view is in shadow and Z-pass rendering that was a bit faster when it is in shadow [5].

Shadow volume and shadow mapping are the classical real-time hard shadow techniques for shadow generation on non-flat surface. In shadow mapping method, two algorithms have been proposed: the conventional shadow mapping presented by Williams [8] and the forward shadow mapping presented by Zhang [9]. In volume shadows, method that was introduced by Crow could be implemented using the stencil buffer on commodity graphics hardware by Heidmann in 1991[2].

There is another recently published and interesting technique to create real-time shadow by McCool. McCool's technique introduced a combination of shadow mapping and shadow volume [4]. In this technique, the main idea was based on observation that at first we generate a shadow map, and then with the information of shadow map such as depth, observation of scene and light sources, we define volume to recognize which objects are in shadow or which of them are out of the shadow. To get some improvement of the shadow volume method, it is good to refer to many article that are published before [see 3,4 &6].

3. METHOD

To create shadows, especially volume shadows, there are some appropriate methods which are needed to be introduced:

3.1 Z-Buffer

In computer graphics, Z-buffering is one of the best tools to maintain three-dimensional (3D) environment and management of image depth coordinates in memory that is usually done in the hardware or the CPU, although some parts of that are in the software or the GPU, this largely depends on the respective algorithms. Z-buffer is also called depth buffer. Williams was the first who used Z-buffer in his algorithm to create shadows. The Z-buffer is used to determine which objects or part of objects are in the viewpoint of light source and which ones are behind the other objects that they will be in the shadow. The Z axis value of scene is stored in this buffer which is located in the main memory. In each scene there are some objects that they have the same X and Y- coordinates, in some part but with different Z-coordinates. These objects create shadow on each other. Z-buffer is a convenient buffer to store depth value and recognize these kinds of objects. It is mentionable here that a good algorithm for hiding objects behind the other objects is called Z-sorting.

3.2. Silhouettes

To implement shadow volume, it is required to detect silhouettes of occluder. A silhouette edge of polygon is the edges that belongs to two neighbourhood plates that is normal vector of one of them is towards the light and normal vector of the other plate is away from the light. If we want to have volume shadow, point by point,the program becomes too difficult to implement . It requires a lot of calculations and as a result takes substantial time to render. To improve this technique, we should recognize the contour edges or Silhouettes of the object and implement the algorithm just for Silhouettes. When we use silhouette edges of the occluder to generate a volume shadow, it is better because in this case we can reduce the amount of memory and therefore, rendering will be done faster. It is mentionable that the silhouette should be recalculated when position of the light source changes or the occluder moves.

There are so many methods to recognize the silhouette of the occluder. In stencil shadow algorithm we need to divide silhouette face of occluder to triangle meshes. This means that each

edge should be shared by just two triangles. To determine which edge is silhouette, we need to know each edge shared between faces toward the light source and faces away from the light source. One of the ways to implement this is through the CPU cycle hungry.

Now we are going to introduce a formula to find a silhouette:

$$S = \sum_{p=1}^{NP} \sum_{q=1}^{SN} iff(V1_{pq} \in S[V2_{pq}], Delete\ V1_{pq}, Add\ V2_{pq})$$

Where:
p is a polygon
q is a edge of polygon
NP is number of all polygons
SN is number edges of polygon of P
V1 is first vertex of edge q
V2 is second vertex of edge q
S is an array of vertices

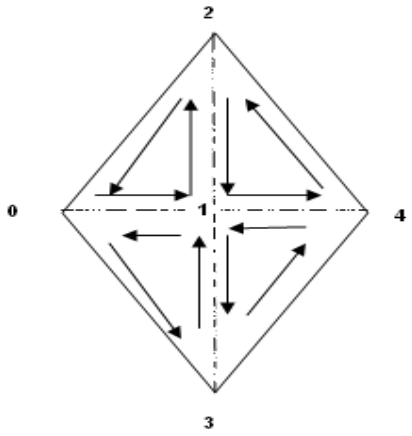

Figure 2: Silhouette determination

Figure 2 illustrates how to divide one side of a box which consist four triangles. To determine silhouette we need just the outline of the box.

In stencil shadow volume one of the most expensive phases is silhouette determination. Although we should not forget cost of update the volume rendering passes.

3.3. Shadow Buffer

As we discussed in the previous section, Williams used Z-buffer in his algorithm that is called the shadow buffer. His algorithm has three phases: firstly, it renders the whole scene from the light source as a viewpoint to calculate the depth value. Secondly, it renders the whole scene again but using the Z-buffer algorithm. Finally, if each point is visible from the viewpoint, it transforms the coordinates also to the light source coordinate system. If the Z-component is greater than Z-component that is stored in Z-buffer, the point is in shadow and if it is not greater than the Z-buffer the point is lit. Watt in 1993 improved on this algorithm which has been applied to solve some problems like aliasing [29].

3.4. Stencil Buffer

Stencil buffer is a part of memory in 3D accelerator that can control the rendering of each pixel. In other word, the stencil buffer is some bits located in the frame buffer that is used for drawing of each pixel of object in rendering window. In 2002, the stencil test was proposed by Everitt [22].

To generate shadows it is important to know which objects or parts of objects are in shadow and which parts are in lit. We should do this test for each pixel of scene in each render. For this, we should count the time of entering and leaving of the shadows. The stencil buffer is the best tools to keep this counting for volume shadows.

There is a close relationship between stencil buffer and Z-buffer, both are of neighbourhood and located in the graphics hardware memory. Z- Buffer needs to control a selected pixel of stencil value that is increased or decreased when we want to count the time of entering and leaving in the shadow volume. Fortunately, the Z-buffer tests are after the stencil buffer tests. The stencil buffer will increase when eye's ray enter the shadow volume by passing from front face of shadow volume and it will decrease when ray leave the shadow volume by passing of back face of that. This has two phases:

(1) At first render the front faces.
    If (depth test passes)
        Stencil Buffer (INCrease)
(2) In the second render
    If (depth test passes)
        Stencil Buffer (DECrease)

The figure 3 illustrates the above algorithm in 2D.

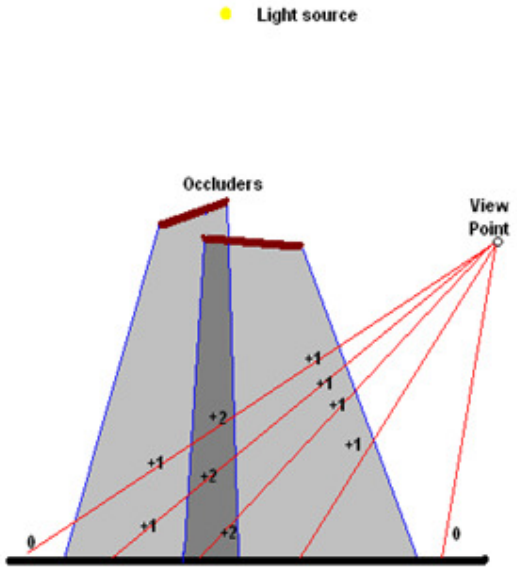

Figure 3: The stencil buffer value

Nowadays, Opengl and DiretX support stencil buffer, so we used depth buffer and stencil buffer together. One of the uses of stencil buffer is the limitation of area.

4. PROJECTED PLANAR SHADOWS

There are so many shadow techniques, like drawing a dark shape similar the occluder under the occluder on the plane, although this is not precise, but it is used frequently especially in older computer game(Half loaf is better than none) . The other simple method to create real-time shadow is projective shadow algorithm [10] that is still widely used in game programming. In this method we can have shadow just in the flat surface such as wall or on the ground but not both of them together. In this method we should draw projection of each pixel of occluder on the shadow receiver with Y=0 along ray that started from the light source until the surface [12].

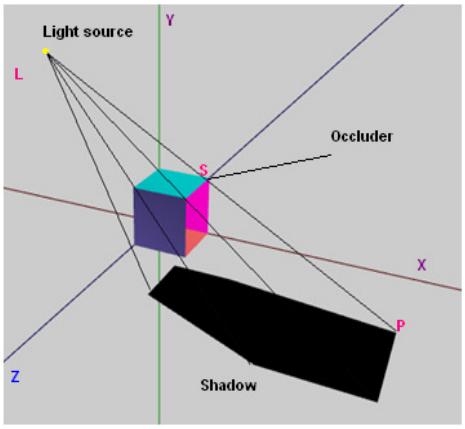

Figure 4: Projection shadow on flat surface

L is light source; P is a pixel of occluder that S is projection on the floor with Cartesian system. According to the basic formula for shadow on the flat surface, there is a shadow matrix that provides facility for moving light source or changing the position of occluder[28].

N: Normal vector of ground.
C: an arbitrary point on ground.
S=L + α (P-L)

(P-C) N = 0 =>$P = L + \frac{E}{NP-D}(P - L)$

Shadow matrix=

$$\begin{pmatrix} LxNx + E & LxNy & LxNz & -ELx - DLx \\ LyNx & LyNy + E & LyNz & -ELy - DLy \\ LzNx & LzNy & LzNz + E & -ELz - DLz \\ Nx & Ny & Nz & -D \end{pmatrix}$$

In modeview transform, if we put this matrix in the current matrix (In Opengl you can use MultMatrix()), it affects on the first object that we want to draw. This matrix, projects pixel by pixel of object that you want to have shadow for, on the plane with N normal vector related to light source. Multiplying of this matrix by each point makes shadow of that pixel.

One of the advantages of this kind of shadow is the fact that shadow is an abstracted object. In other words, it can be manipulated as another object. It is possible to change its colour, simply. The other advantage is that it is easy to compute. On the contrary, one of the disadvantages of this shadow is the fact that it is convenient for flat surface and it is so difficult to have shadow onto the other object and it requires graphic hardware with ability of clipping to clip boundary of each polygon. The other disadvantage of projection shadow is dark shadow without illumination.

It is mentionable here that there are some other problems in projection shadow like on the unrestrained areas, but it can be solved with stencil buffers. Therefore, first we should render the shadow receiver and transfer this to the stencil buffer. Then, draw the scene where shadow receiver was drawn and finally render all scenes except the part that has been rendered already. Opaque shadow in projection shadow is another limitation. To have semi-transparent shadow requires much more work especially for concave object that we have to split to some convex parts. And finally, the shadow should be rendered in each frame, even when shadow does not have a change.

In addition, although the quality of this shadow is not convenient but the cost of producing is low and we use it in some part of our projects. There are some ways to improve the projection shadows. One of the best ways to have projection shadow on other object is using stencil shadow and double blending.

5. SHADOW VOLUME USING STENCIL BUFFER AND DEPTH BUFFER

Before we discuss further on shadow volume it is good to know about illumination models that Phong proposed. In this model, intensity is related to three components: ambient, diffuse and secular.

$I = I_a + I_d + I_s$

$I_a=K_aI_a$

$I_d=K_d\ I_l\ (N.L)$

$I_s=I_lK_s(N.H)^n$

$I = I_aK_a+ K_d\ I_l\ (N.L)+ I_lK_s(N.H)^n$

K: surface attributes
L: light vector
N: surface normal vector
H: vector between lighting vector and the viewing vector.
n: constant for roughness

In 1977 Crow[10] could not implement his algorithm because of lack of enough hardware at that time. Heidmann [2], in 1991, completed the shadow volume algorithm of Crow and implemented it. He created shadow on arbitrary objects. This is illustrated as follows:

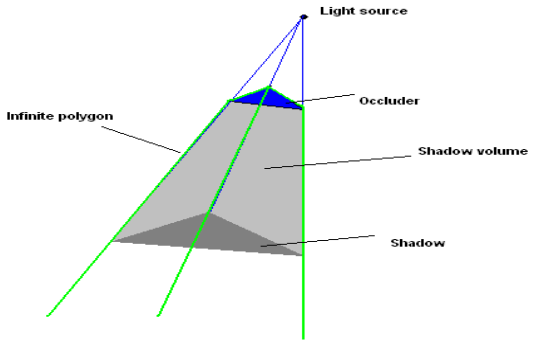

Figure 5: Shadow volume

We should draw a line form point of light source to the each vertex and these rays continue infinitely. After recognizing each pair of neighbourhood rays, we draw a polygon from the edge of occluder onto the shadow receiver. This method is also called shadow volume boundary polygon or simply shadow polygon. These polygons make an infinite truncated pyramid that the part of object that is located below the occluder is the shadow volume, that is, semi infinite quadrilateral with two finite vertices and two other vertices that located in infinity that is showed with green lines in Figure 5.

To speed up our algorithm we will clip each infinite quadrilateral to the finite quadrilateral between occluder and shadow receiver. To find a minimum length of volume in each pixel we propose new pixel:

$$(x_{new}, y_{new}, z_{new}) = \begin{pmatrix} Lx + \frac{E(Fx-Lx)}{NP \quad D} \\ Ly + \frac{Ly+E(Fy-Ly)}{NP-D} \\ Lz \mid \frac{E(Fz-Lz)}{NP-D} \end{pmatrix}$$

Now, if each object or part of object is located inside of this truncated pyramid it is in shadow and should be dark but each object or part of that is located out of this truncated pyramid is in lit and it does not need any changes.

Now we are going to generate volume shadow using stencil buffer and depth buffer together. First, in simple word we can say this algorithm in summarize:

(1) Turn off the light and render all scene just with ambient and emission lighting.
With this the color buffer will be full for all pint of object in shadow and also Z-buffer will be full with depth value.
(2) After disable Z-buffer and Color buffer to write, render all scenes with lighting.
(3) Subtract of this two depth value provides shadow volume.

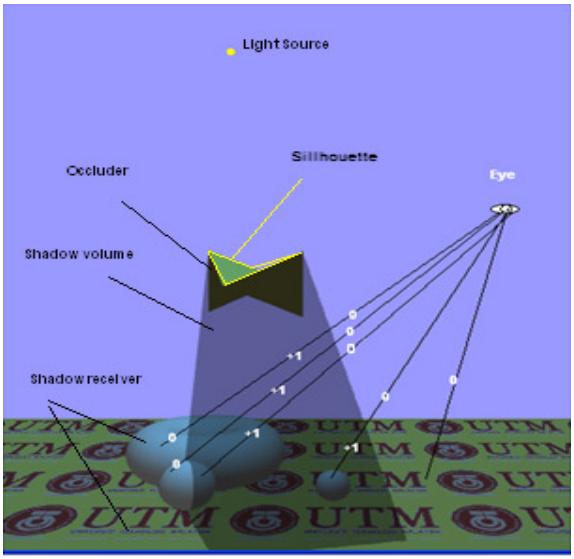

Figure 6: Shadow volume using stencil buffer and depth buffer

After generating the volume shadow, we should pass the ray from eye to the each point of object on the shadow receiver. As we said earlier (in stencil buffer), if the ray passes the front face of volume shadow, we increase the stencil buffer and when the ray pass the back face of shadow volume, we decrease the stencil shadow. Finally, if the number of stencil buffer is not zero it is therefore in the shadow. We present the algorithm as follows:

(1) Render the all scene without lighting

(2) Enable stencil buffer.

(3) Disable depth buffer to write and prevent to write in color buffer.

$$(4)\ \sum_{i=1}^{NP} Iff(P_i, R, \sum_{i=1}^{NP'} Iff(Ztest, Stencil++, Nil), Nil)$$

$$(5)\ \sum_{i=1}^{NP} Iff(\sim P, R, \sum_{i=1}^{NP'} Iff(Ztest, Stencil--, Nil), Nil)$$

(6) Render the all scene again with lighting.

(7) Enable to write in color buffer.

(8) If (~Stencil%2)
Keep Stencil

We use Z-pass algorithm when eyes are out of shadow. If eyes are in shadow we should use Z-fail algorithm that we are introduce in following:

As a matter of fact, the whole the algorithm is like Z-pass algorithm except the step 4 and 5:

$$(4)\ \sum_{i=1}^{NP} Iff(\sim P, R, \sum_{i=1}^{NP'} Iff(\sim Ztest, Stencil\ |\ |, Nil), Nil)$$

$$(5)\ \sum_{i=1}^{NP} Iff(P_i, R, \sum_{i=1}^{NP'} Iff(\sim Ztest, Stencil--, Nil), Nil)$$

6. RESULTS

We implemented this method to generate shadow on 3-D object on none-flat surface like torus and sphere. In this implementation the number of frame per second is different when we use this algorithm without shadow and with shadow. The wonder result is related to the one light source; in this case number of frame per second is not much different with the case of that without shadow in scene. It may be applied in current commercial games or other virtual reality systems. Figure 7 shows the result of this algorithm for more number of light sources and effect of them on number of frame per second:

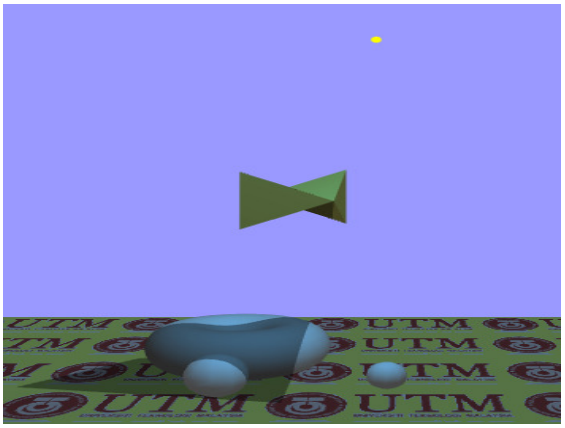

Figure 7: Result of volume shadow

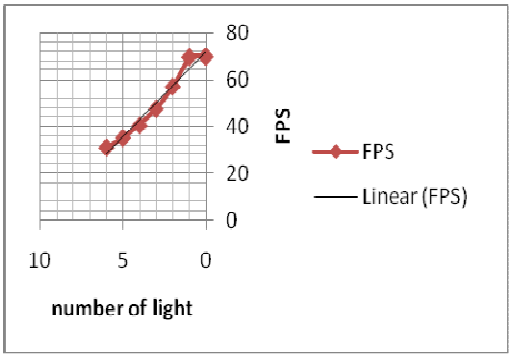

Figure 8: Frame per second

7. CONCLUSION

In this paper we have presented two kind of method to create real-time shadow. Projection shadow is simple to implement and shadow volume that was difficult to understand and implement. Stencil buffer is so appropriate to implement volume shadow especially when it will be combining with Z- buffer. Combination of these two buffers and using propose formula to recognize silhouette can improve current algorithm of volume shadow. This paper has tried to convert current volume shadow algorithm and outline detection to mathematics modelling to understand easily. We have proposed the one of simplest ways to implement shadow volume using stencil buffer and depth buffer. Using stencil buffer can reduce some part of calculation. Although shadow volume is geometry computation and it requires supporting of CPU but we share it in CPU and GPU.

REFERENCES

[1] Catmull, E ,(1974), "A SubdivisionAlgorithm for Computer Display of Curved Surfaces," University of Utah. 1974 - Cited by 784 - Related articles

[2] Tim Heidmann, (1991)"Real Shadows Real Time", IRIS Universe,Number 18pp. 28-31.Silicon Graphics Inc., *1991*. 24.

[3] David Blythe, Tom McReynolds, et.al.,(1996) “Shadow Volumes,” , Program with OpenGL: Advanced Rendering, SIGGRAPH course notes.

[4] Michael D. McCool,(2001) “Shadow Volume Reconstruction from,Depth Maps,” ACM Transactions on Graphics, Canada N2L 3G1, http://www.cgl.uwaterloo.ca/~mmccool/. To appear in ACM Transactions on Graphics, 2000. Jan, pp.1-25.

[5] Eric Lengyel,(2002) "Mathematics for 3D Game Programming & Computer Graphics", Charles River Media.

[6] Nan LIU, and Ming-Yong PANG, (2009) A Survey of Shadow Rendering Algorithms: Projection Shadows and Shadow Volumes , 2009 Second International Workshop on Computer Science and Engineering, 978-0-7695-3881-5/09 -2009 IEEEDOI 10.1109/WCSE.2009.107 age(s): 488 – 492.

[7] Jim Blinn,(1988) “Me and My (Fake) Shadow,” IEEE Computer Graphics and Applications, vol. 8, no. 1, pp. 82-86.

[8] H.Zhang, (1998) “Forward Shadow Mapping”, In Proceedings of the 9th Eurographics Workshop on Rendering, pp.131-138.

[9] L.Williams,(1978) “Casting Curved Shadows on Curved Surfaces”, SIGGRAPH ’78, Vol.12, No.3, pp.270-274.

[10] Crow, F. (1977) “Shadow Algorithms for Computer Graphics”, Computer Graphics 11(2) 242-247.

[11] Nan Liu; Ming Yong Pang (2009), Shadow Mapping Algorithms: A Complete Survey ,Computer Network and Multimedia Technology,CNMT Page(s): 1 – 5.

[12] Yang Xiao; Jin Yicheng; Yin Yong; Wu Zhuoyu(2006),"GPU based real time shadow research", 10.1109/CGIV..48 Page(s): 372 – 377.

[13] Modern OpenGL(2008) ACM SIGGRAPH ASIA Courses, SIGGRAPH Asia'08 , art. no. 48.

[14] Decoro,C,Cole,F(2007),"Styize Shadows",NPAR Symposium on Non photo realistic , pp 77-83.

[15] Mustafa, S. Fawad Wang, Wencheng.Key. Generation of Stencil Shadow Volumes,

[16] Yang Xiao; Jin Yicheng; Yin Yong; Wu Zhuoyu (2006); GPU Based Real-time Shadow Research,CGIV, Page(s): 372 - 377 .

[17] Woo, A. Poulin, P.; Fournier, A.(1990); A survey of shadow algorithms Computer Graphics and Applications, IEEE , Page(s): 13-32. [18] REN Hong-xiang, JIN Yi-cheng, YIN Yong ICIG(2007),"real-time shadow rendering in scene system of marine simulator" proceeding of the Image and graphic pp 1010-1014.

[19] Carmack, J.(2000). CarmackOnShadowVolumes (Personal Communication between Carmack and Kilgard). Referenced: 10.4.2002. Available: http://developer.nvidia.com/view.asp?IO=robust_shadow_volumes

[20] Franklin C. Crow,University of Texas at Austin,Austin, Texas,Shadow algorithm computer graphic,

[21] Lokovic & Veach, (2000), Lokovic Tom, & Veach Eric. August 2000. Deep Shadow Maps. SIGGRAPH 2000 Proceedings.

[22] Cass Everitt and Mark J. Kilgard ,(2002). Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering .

[23] Fawad, M.S.; Wang, W.; Wu, E.(2005) Atika, M.; Generation of Stencil Shadow Volume INMIC. 334437 , Page(s): 1 – 11.

[24] Peter Grogono. Getting Started with OpenGL,Supplementary Course Notes for COMP 471 and COMP 6761

[25] Liu Hui; Jin Hanjun; Yan Jianlin;(2008). The Research of Real-Time Shadow Rendering , algorithm of Virtual Scenes ETT and GRS, Page(s): 159 – 162.

[26] Pierre(HR) Rautenbach, (2008).An empirically devived system for high speed shadow rendering .

[27] Fernado, R., Fernadez, S.(2001), BALA, K., And Greenberh, D. P. Adaptive shadow maps. In Proceedings of ACM SIGGRAPH 2001, ACM Press / ACM SIGGRAPH, Computer.387–390. ISBN 1-58113-292-1.

[28] Tomas M¨ oller and Eric Haines, A.K. Peters, Excerpted and updated from the book ”Real-Time Rendering”, ISBN 1568811012. See http://www.realtimerendering.com/].

[29] Watt, A. (1993). 3D Computer Graphics. Second Edition. Addison Wesley. 500 p.

[30] Rua Paese(1999) .Real-Time Shadow Generation Using BSP Trees and Stencil Buffers, Page(s): 93 - 102

Authors

Hoshang Kolivand received the B.S degree in Computer Science & Mathematic from Islamic Azad University, Iran in 1997, M.S degree in Application Mathematic and computer from Amirkabir University, Iran in 1999. He is now pursuing Ph.D in UTM ViCunelab under the guidance of Dr Mohd Shahrizal Bin Sunar. His research interests include Computer Graphics. Previously he is a lecturerin Shahid Beheshti University Iran. He has published some articles in international journals and conference and also he had published 3 books in object oriented programming and one in mathematics before start to study in Phd.

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 the head of computer graphics and multimedia department, 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 professional member of ACM SIGGRAPH. He is also a member Malaysian Society of Mathematics and Science.

Datin Normal Binti Mat Jusoh received the B.Sc. in BEng(Hons) Manufacturing System Engineering and M.Sc. degrees in Multimedia Information System from the University Of Portsmouth , United Kingdom, and currently pursuing PhD in Computer Games Entertainment Industry : Games Ontology at the Universiti Teknologi Malaysia. She has published articles in international and local journals. She is a lecturer in Universiti Teknologi Malaysia.

Olufemi Folorunso received the B.Sc. and M.Sc. degrees in Mathematics and Computer Science from the Obafemi Awolowo University, Ile-Ife, and the University of Lagos, Nigeria in 1992 and 1997 respectively. He is a Senior Lecturer at the Yaba College of Technology, Lagos, Nigeria and just completing a doctorate degree Ph.D (Computer Science) at the Universiti Teknologi Malaysia. His research interests include signal processing, augmented reality and scientific visualization. He has published articles in international and local journals. He is a member of the Nigerian Computer society, the Computer Professional Registration Council of Nigeria and a member of vizNET, UK.