International Journal of Computer Graphics
Vol. 2, No. 1, May, 2011

To Combine Silhouette Detection and Stencil Buffer for Generating Real-Time Shadow

[1]Hoshang Kolivand, [2]Mohd Shahrizal Sunar
[1,2]Faculty of Computer Science and Information Systems, Department of Computer Graphics and Multimedia, University Teknology Malaysia – [1]Islamic Azad University, Firozkoh Branch, Iran
[1]E-mail:shahinkey@yahoo.com
[2]E-mail: shah@fsksm.utm.my

Abstract

This paper describes one of the best methods to generate real-time shadow that called volume shadow. Volume shadow algorithm is possible to implement for virtual environment with moveable illuminated light source. We improve to find silhouette and implementation of the last techniques used in shadow volume algorithms, and limitation of length of volume of each semi-polygon that makes between silhouette's edge and shadow receiver. We rewrite the last volume shadow algorithm using stencil buffer and propose a pseudo code in OpenGl. The best tools that we have used in this algorithm are stencil buffer and Z-buffer.It makes realistic commercial games or it can be use in virtual reality systems.

Keywords: *silhouette detection; volume shadow; stencil buffer; depth-buffer*.

1. Introduction

In commercial games or virtual environment shadows are more attractive effect to make them realism and visual quality of images, but they are difficult to implement in display environment especially in real-time games. In real time environment such as computer games, shadows give the user feeling that they imagine they play in realistic world and they can enjoy as much as possible. A game with lack of shadow effect cannot be attractive nowadays users expect more realistic games and they are not satisfy of each kind of games. To generate a realistic game shadow is substantial important.

There are a lot of methods to create shadow but the shadow volume technique is powerful to have a self shadowing and cast the other objects. Although there are some improvements in the shadow research that have been did before, but they did not solve all of them and there are still many problems related with the shadow rendering specially in soft shadows. Now, computer graphics researcher tries more finding new algorithm or to improve the algorithms that exist before.

The last shadow volume algorithm is difficult to understand and to implement and also it is expensive to implement. But by propose this algorithm we reduce the expense of reorganization of silhouette.

2. Previous Work

The idea of shadow volume was introduced in 1977 at first time, when Frank Crow [4] published his ray-casting based 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 even today the main shadow volume algorithm [1]. Stencil shadows belong to the group of volumetric shadow algorithms, as

the shadowed volume in the scene is explicit in the algorithm. Let's take a look at how the original stencil shadow volume technique works.

The other algorithm that Carmack suggested in year 2000 was a bit different with the previous algorithm which include rays are traced from infinity towards the eye. Which is discussed between them [10]. shadows of self-shadowing primitives such as hair, fur and smoke [11]. Shadow Maps suggested by Fernando et al [12] which are an extension to the traditional shadow mapping technique. In year 2002 Lengyel propose a hybrid algorithm that uses faster Z-pass rendering [3].

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 [5] and the forward shadow mapping presented by Zhang [6]. In volume shadow method that was introduced by Crow in 1977 can be implemented using the stencil buffer on commodity graphics hardware by Heidmann in 1991[1].

3. Silhouettes

Silhouettes have a most important role to recognize and project shape onto shadow receiver. Because to create shadow, projection of silhouette of occluder is enough to generate shadow of whole object and as a result the cost of projection will be low. The most expensive part of shadow volume algorithm is identification silhouette of occluder.

A silhouette edge of polygon is the edges of that where is belong to two neighborhood plates that normal vector of one of them is toward the light and normal vector of the other plate is away from the light. If we want to have volume shadow, point by point, it is too difficult to execute program . It needs a lot of calculation and as a result it takes a substantial time to rendering. To improve this technique we should recognize the contour edges or Silhouettes of 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 , render will be done faster. It is mentionable that the silhouette should be recalculated when position of the light source changes or occluder moves.

There are so many method to do this but they are expensive but we introduce a algorithm that is not expensive like past algorithm. In stencil shadow algorithm we need to divide silhouette face of occluder to triangle meshes. It means that each edge should shared by just two triangles. To determine which edge is silhouette, we need to know each aged shared between faces toward the light source and faces away to the light source. Now we are going to introduce a technique to find a silhouette:

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

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=3

V1 is first vertex of edge q

V2 is second vertex of edge q

S is a array of vertices

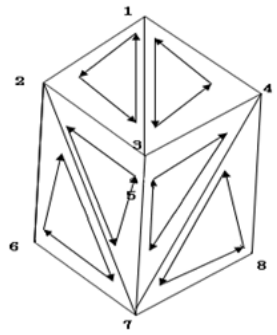

Fig 1: Silhouette Determination

Fig 1 illustrates how to divide one side of a box which is consist of four triangles. To determine silhouette we need just outline of the box not all of the edges. This algorithm as a flowchart is like:

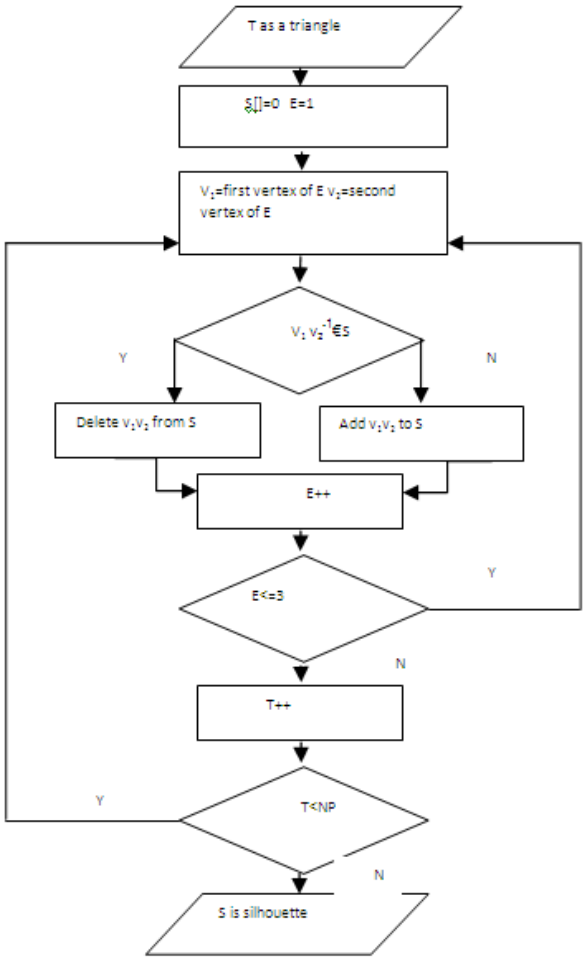

Fig 2: Silhouette Determination Flowchart

It is important to note that silhouette determination is one of the two most expensive operations in stencil shadow volume implementation. The other is the shadow volume rendering passes to update the stencil buffer. These two steps are the phase that we can work on them in future.

4. Stencil Buffer

Stencil buffer is a part of memory in 3-D accelerator that can control the rendering of each pixel. In other word the stencil buffer is a number bits in the frame buffer that use for enable and disable drawing of each pixel of object in rendering and also it can be use to mask color buffer.

There is a close relationship between stencil buffer and Z-buffer. Because both these buffers are neighborhood 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 is 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)

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

5. Shadow Volume Using Stencil Buffer and Depth Buffer

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

$$I = I_a + I_d + I_s$$

$$I = I_a K_a + K_d I_l (N.L) + I_l K_s (N.H)^n$$

K: surface attributes

L: light vector

N: surface normal vector

H: vector halfway between L and the viewing vector.

n: constant simulating roughness

Now if each object or part of object that is inside of truncated pyramid is in shadow and it should be dark but each object or part of object that is out of truncated pyramid is in lit and it should not be dark. Now we are going to generate volume shadow using stencil buffer and depth buffer together. First, in simple word we can say the algorithm in brief :

1-Render the all scene just with ambient.

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 scene with lighting.

3-Subtract of this two depth value provides shadow volume.

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

$$(x_{new}, y_{new}, z_{new}) = \begin{pmatrix} Lx + \frac{E(Px-Lx)}{NP-D} \\ Ly + \frac{Ly+E(Py-Ly)}{NP-D} \\ Lz + \frac{E(Pz-Lz)}{NP-D} \end{pmatrix}^T$$

After generate volume shadow, we should pass the ray from eye to the each point of object on the shadow receiver. As we told before (in stencil buffer) ,if the ray pass the front face of volume shadow, increase the stencil buffer and when the ray pass the back face of shadow volume , decrease the stencil shadow. Finally, if the number of stencil buffer is not zero it is in shadow.

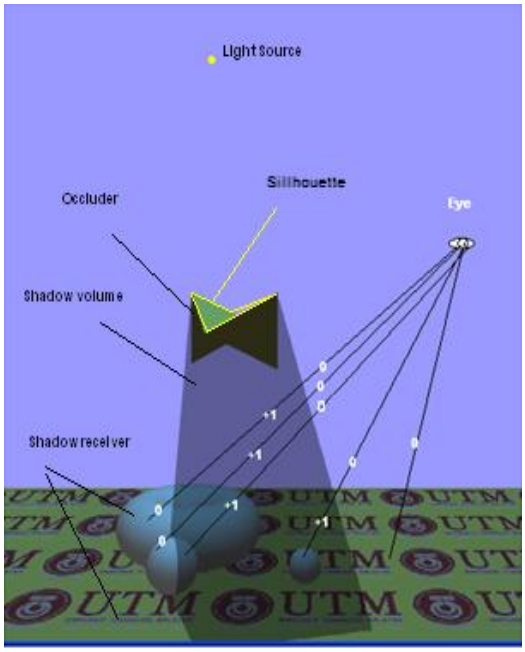

Fig 3: Shadow Volume Using Stencil Buffer and Z-buffer

We are going to say this as a algorithm or pseudo code in Opengl:

1-Provide the equipment to have stencil buffer.

```
glutInitDisplayMode(GLUT_STENCIL|GLUT_DEPTH|GLUT_SINGLE|
GLUT_RGBA);
```

2-Render the all scene without lighting

```
glDisable(GL_LIGHTING);
renderScene();
```

3-Enable stencil buffer.

```
glEnable(GL_STENCIL_TEST);
glDepthFunc(GL_LEQUAL);
```

4 -Disable depth buffer to write and prevent to write in color buffer.

```
glDepthMask(False);
```

```
        glColorMask(0, 0, 0, 0);
        glStencilFunc(GL_ALWAYS, 1, 0xffffffff);
    5-For (i=0; i<number of Planes;i++){
            If (Inner product(planes[i],Ray)>0)
          For (j=pixel; on the polygon;j++){
                If (Z-test pass){
                        glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
                }}}
    6-For (i=0; i<number of Planes;i++){
        If (Inner product(planes[i],Ray)<0)
          For (j=pixel; on the polygon;j++){
                If (Z-test pass){
                        glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);
          }}}
    7-Render the all scene again with lighting.
    8-Enable to write in color buffer.
        glColorMask(1, 1, 1, 1);
        glStencilFunc(GL_EQUAL, 1, 1);
    9-If (the stencil buffer %2==0 )
        glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
```

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 face whole the algorithm is like Z-pass algorithm except the step 5 and step 6:

```
    5-For (i=0; i<number of Planes;i++){
        If (Inner product(planes[i],Ray)<0)
          For (j=pixel; on the polygon;j++){
                If (Z-test fail){
                        glStencilOp(GL_KEEP, GL_INCR, GL_KEEP);
          }}}
    6-For (i=0; i<number of Planes;i++){
        If (Inner product(planes[i],Ray)>0)
          For (j=pixel; on the polygon;j++){
                If (Z-test fail){
                        glStencilOp(GL_KEEP, GL_DECR, GL_KEEP);
          }}}
```

6. Results of my Project

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. This below table shows shadow volume by this algorithm is not so expensive and we can use it in new games and virtual environments:

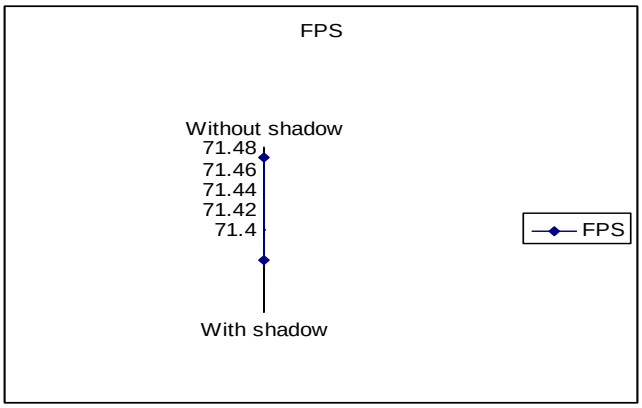

Table 1: Compare FPS

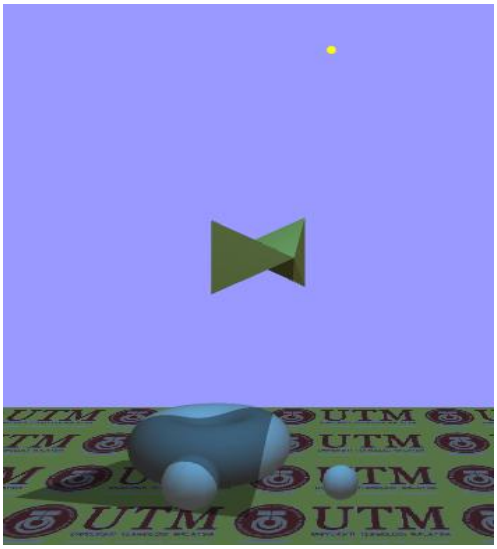

Fig 4: Shadow Volume

7. Conclusion

In this paper we have present a method to recognize silhouette of occluder to have volume shadow on arbitrary objects . Shadow volume that was difficult to understand and implement we improve it and make it simple to implement and fast to identify silhouette. We have propose the one of simplest ways to implement shadow volume using stencil buffer and depth buffer. Introduce algorithm using one programming languages means pseudo code and it is simple to understand. To have a real-time shadow on arbitrary objects, shadow volume is convenient method. Shadow volume is geometry computation and it requires supporting of CPU.

References

[1] Tim Heidmann, 1991,"Real Shadows Real Time", IRIS Universe,Number 18pp. 28-31.

[2] David Blythe, Tom McReynolds, et.al., 1996, "Shadow Volumes," , Program with OpenGL: Advanced Rendering, SIGGRAPH course notes.

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

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

[5] H.Zhang,,1998, "Forward Shadow Mapping", In Proceedings of the 9th Eurographics Workshop on Rendering, pp.131-138.

[6] L.Williams, 1978, "Casting Curved Shadows on Curved Surfaces", SIGGRAPH '78, Vol.12, No.3, pp.270-274.

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

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

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

[10] 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

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

[12] 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.

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

Authors

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 Mathematics and Science.

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, and registered for Ph.D. in Computer Science University Teknology Malaysia under the guidance of Dr Mohd Shahrizal Bin Sunar. His research interests include Computer Graphics. He has published 3 books in object oriented programming and one in mathematics.