INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH
Beyn=lxalq Elmi Araşdırmalar Jurnalı
Print: **ISSN: 2075-4124** **Volume 4, Number 3, pp. 209-218, 2012**

EXTENDED EDGE SILHOUETTE DETECTION ALGORITHM TO CREATE REAL-TIME SHADOW VOLUME

Hoshang Kolivand[1*], Mohd Shahrizal Sunar[2] , Amjad Rehman[3]
[1,2] UTM ViCubelab, Department of Computer Graphics and Multimedia, Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia,81310 Skudai Johor (Malaysia)
[1]E-mail:shahinkey@yahoo.com, [2]E-mail: shah@cs.utm.my,[3]rkamjad@gmail.com

ABSTRACT

A new algorithm to recognize occluder silhouette to generate shadow volume with some improvement of the traditional shadow volume algorithm is proposed. We named the new algorithm as Extended Edge Silhouette Detection (EESD). Silhouette detection is one of the expensive processes of shadow volumes. The triangular algorithm and visible-non-visible algorithm that are famous algorithms to detect the outline of occluder are renewed. An accurate comparison between these algorithms and a new algorithm has been done. The algorithm is to detect silhouette and decreases the cost of implementation of shadow volume. The obtained results confirm superiority of the proposed algorithm in terms of processing time compared to the previously used traditional algorithms. The last shadow volume algorithm using stencil buffer is rewritten and an algorithm for shadow volume with silhouette detection together is proposed.

Keywords: silhouette detection, shadow volume, stencil buffer, real-time

1.Introduction

To have a realistic environment, shadow is the most important effect that reveals more information about the distance between objects in the scene. It is the major factor of 3-D graphics for virtual environment but unfortunately is difficult to be implemented in display environment especially in real-time games. In computer games, shadows give the gamers feelings that could trigger an imagination that they are playing in the real world hence provides maximum pleasure. A game with lack of shadow cannot be attractive especially in this century that gamers' imagination requests more realistic situation when they are watching cartoons or playing games.

There are many algorithms to create shadow but shadow volumes have a great achievements in game makers. Although volumes shadow is considered as established in gaming industry, they have two expensive phases. One of them is update of volume rendering passes and the other one is silhouette detection.

To have a shadow volume it is needed to recognize boundary of occluder and other aspects are not important. Outline detection is required in each frame and as a result taking the CPU time of CPU. It is important to reduce this time and increase frame per second (FPS) of movies.

Recognizing the outline of object can increase the speed of algorithm. To find out outline of object, silhouette detection is essential because it can reduce the cost of implementation and it is the main item to improve an algorithm. Silhouettes are the special part of outline that can contribute to make objective. Silhouette detection is an important phase in all visual effects. It is mentionable that the bases of silhouettes are visual and view point. To generate shadow, silhouette detection plays a crucial role to detect the boundary of occluder. Just the silhouette contributes in shadow and the other outline of occluder is not important.

Silhouette detection is a function like f: $R^3 \rightarrow R^2$. In 1987 Richards, Koenderink, and Hoffman [1] were the first researchers proposing a method to interpret silhouette of arbitrary objects. In 1990, Saito and Takahashi [2] used G-buffer to recognize silhouette. G-buffer is the base of Z-buffer produced by adding some geometric data.

Hertzmann proposed an algorithm for irregular surface with lack of continuity and silhouette based on combination of normal map and depth mapping. This hybrid algorithm changes the face of object using Z-buffer [3]. Raskar and Cohen [4] introduced a way to recognize silhouette by putting front face of polygon on the back face of it. In 1999, Gooch et al.[5] used a similar method to compute silhouette.

In 2000, Northup and Markosion proposed an algorithm, although it is very simple and appropriate, it is difficult to find starting points of each coherence. In their algorithm the spatial coherence is so impressive.

In 2003, Isenberg[7] classified reorganization of silhouette in three groups, which are image space algorithms that focus on image processing , object space algorithms that are divided in two groups, free form and polygon mesh and the second group is hybrid algorithms. The object space algorithms are better than the others due to its advantages in finding accuracy of silhouette and appropriate stylization [9,7].

In 2004, Soon Ki Jun et al.[10] introduced an algorithm that uses the spatial coherence and frame coherence together.

An easy and simple way to find silhouette is to check all edges. At first, programmer used this algorithm to recognize silhouette for polygon models. For this computation information about neighboring edges is needed and

it requires recalculation when the light source or viewer's position changed. But it is wasting time; hence they have discontinued the study and started to research about appropriate algorithm [5, 18].

Benichou and Elber[8] used a method to compute parallel silhouette edges of polyhedral object based on Gaussian sphere, for the normal vectors which are located on the view direction mapped onto a Gaussian sphere.

Matt Olson and Hao Zhang [16], in 2006 work on tangent-space and they focused on tangential distance of objects to be used in polygon mesh silhouette detection. In 2010, Matt Olson[11], designed a site that lists all related papers.

In EESD algorithm the average of operation is less than the others and as a result it is faster. In the traditional shadow volume which uses triangular silhouette, the number of operation is high thus resulting in fewer frames per second.

After introducing the silhouette and two famous techniques to recognize occluder's outline, the number of multiplication, addition, and comparison are calculated. Finally an accurate comparison between them has been done.

2.Silhouette Definition

Silhouettes have the most important role to recognize and project shape onto the shadow receiver. To create shadow, projection of silhouette of occluder is enough to generate a shadow of the whole object and as a result the cost of projection will be low. A silhouette edge of polygon is the edges that belong to two neighborhood planes where normal vector of one of them is towards the light and normal vector of the other plane is away from the light [6]. If the shadow volume is desired point by point, it is too difficult to execute the program. It needs a lot of calculation and then it takes a substantial time of CPU for rendering. To improve this technique we should recognize the contour edges or silhouettes of object and implement the algorithm just for silhouettes. Using silhouette edges of the occluder to generate a shadow volume could optimized the process because the amount of memory is decreased, therefore, rendering will be done faster. The silhouette should be recalculated when position of the light source changes or the occluder moves [21].

Regarding the techniques used to recognize silhouette, there are five categories:

- **Silhouette edges;** include all visible edges which connect the back faces to the front faces.
- **Ridge edges or sharp edges;** these kind of edges are the edges which have one visible plane face and one invisible plane face.
- **Valley edges;** these kinds of edges have an angle greater than or equal to the threshold angle between two plane faces that have made the edge.
- **Border edges;** which is also called boundary edges.
- **Combination of shadow outline and texture boundary outline**.

Assume that:

v is point of view

f_i is a face $\in F$

f_v is a vertex of *f*

$\vec{n}_f$ is normal vector of *f*

$e \in E$ *is a edge of occluder*

e_{f_i} $\forall$ *i=1,2* are faces inculde *e*

p_f is a plane of *f*

The distance between v and f is:

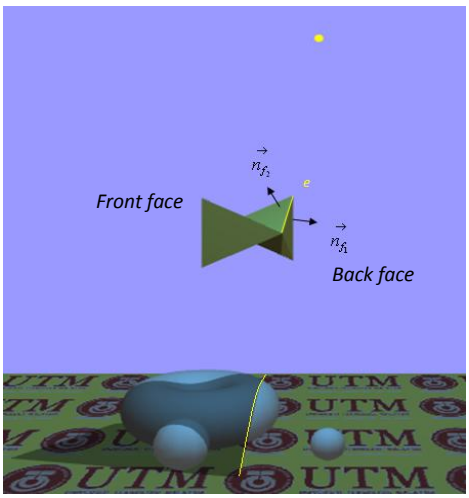

***Fig.1**. Effect of silhouette detection on shadow*

$||v, f|| = (\vec{v} - \vec{f}_v).\ \vec{n}_f$

Now with the sign of $||v,f||$ available, the face could be recognized as back face or front face.

If $||v, f||>0$ then
  f is a front face
Else
  f is a back face
End if

If $||v, e_{f_1}|| \ ||v, e_{f_2}|| <0$ then
  e is a silhouette
End if

For smooth faces this can be normalized to recognize silhouette edge.

$$D_i = \frac{(\vec{f}_{v_i} - v)\,\vec{n}_{f_i}}{\| n_{f_i} \| \| f_{v_i} - v \|}$$

If $D_i>0$ then
  f_i is front face
else if $D_i<0$
  f_i is back face
else if $D_i=0$ then
  v_i is silhouette
end if

3. Silhouette Detection Algorithms

Although, there are many silhouette detection algorithms, we wish to compare EESD algorithm with the most popular ones. First, it is necessary to list popular and useful algorithms as this could facilitate easier understanding.

3.1. Triangular Algorithm

In this algorithm proposed by Harlen and Ilaim[12] in 1999, it is needed to divide each face of occluder into triangle meshes. It means that each edge should be shared by just two triangles. To determine which edge is silhouette, it is needed to know which edge is shared between faces towards the light source and faces away of the light source [22].

First lets have an array of all edges (S)

For p=1 to Number Of Faces
If p.visible=True then
  For q=1 to Number Of Side
    If $v_p v_q^{-1}$ is belong to S then
      Delete $v_p v_q$ from S
    Else
      Add $v_p v_q$ to S
    End if
  Next
End if
Next
S is a list of silhouette

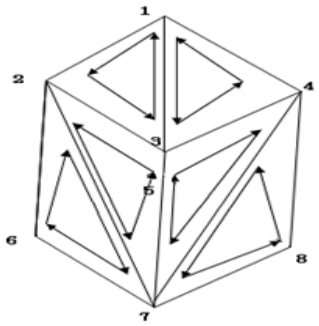

***Fig.2**. Triangular mesh*

Before starting this algorithm, all faces of occluder should be divided into some triangles and all of these edges of triangles should be put in an array (S). Then, from the first of array ,if the inverse of each edge belongs to the S, it is not silhouette and it should be omitted from the array. On the other hand, if it does not belong to the array it is silhouette and should be put in the array, until the end of array.

3.2. Visible-Non visible Algorithm

In this algorithm proposed by Soon Ki Jung in 2004[10], all edges which have just one visible face are silhouette. These edges which have exactly one visible and one invisible face are silhouette, but on the contrary, each edge with two visible faces or two invisible faces is not regarded as a silhouette.

```
For E=1 to Number Of Edges
    For F=1 to Number Of Faces
        If F.visible =True then
            E_ counter =E_counter+1
        End if
    Next
    If  e_counter=1 then
        Add E to S

Next
```

S is a list of silhouette

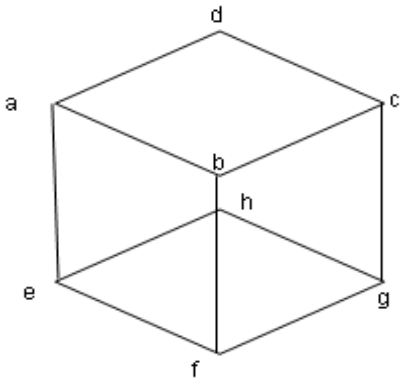

***Fig.3**. Silhouette detection*

This algorithm is very simple. Assume that the light source is located in the viewer eyes. The edge ***ad*** in Figure 3 belongs to one visible face and one invisible face, and then it is silhouette while the edge ***ab*** belongs to two visible faces then it is not silhouette. Edge ***hg*** belongs to two invisible faces and it is also not a silhouette.

3.3. EESD Algorithm

Here we propose a new algorithm which is called EESD Algorithm that will be described in the following. Assume that the light source point and view point are located in one place and they are in the eyes of viewer. For each edge v1v2, extend v1 to v1+Δ_{v1} where $\Delta_{v1} \rightarrow 0$. Then make a ray (R) from light source to the v1'=v1+Δ_{v1} and extend to infinity. If it has an intersection with one of the faces of occluder it is not a silhouette but on the contrary, if it doesn't have any intersection with occluder it is a silhouette.

```
For e1=1(v1v2) to Number Of Edges
        v1'=v1+ Δv1
        R=ray of light source to v1'
        for f=1 to Number of Faces
            if Intersection(R,f)=false then
                Add v1v2 to S
        End if
    Next
Next
```

If the light source is assumed to be located at the view point and with the help of Figure 4, imagine a ray R from point of view to the v1' point and continue the ray to the infinity. If it has an intersection with two faces of occluder, it is not a silhouette. On the contrary, if it has no intersection with the occluder (green edge) it is a silhouette.

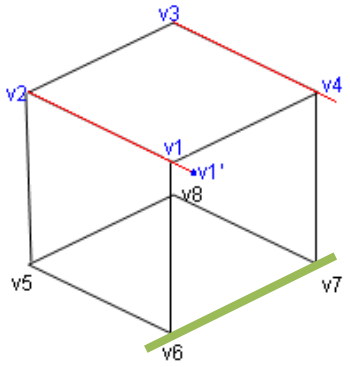

Fig.4. *Silhouette detection*

4. Shadow Volume Using EESD Algorithm

In 1977, Crow [13] could not implement his algorithm because of lack of enough hardware in that time. Heidmann [14], in 1991, complete the shadow volume algorithm of Crow and implemented it. He could then create shadow on arbitrary object. The algorithm in simple word is as a follows:

A line from the point of light source to each vertex should be drawn and continued to infinity. After recognizing each pair of neighboring rays a polygon from the edge of occluder should be drawn onto the shadow receiver. This method is also called shadow volume boundary polygon or simply shadow polygon. These polygons make an infinite truncated pyramid where the part located below the occluder is shadow volume that is semi infinite quadrilateral with two finite vertices and two other vertices located in infinity shown as green lines in Fig.5.

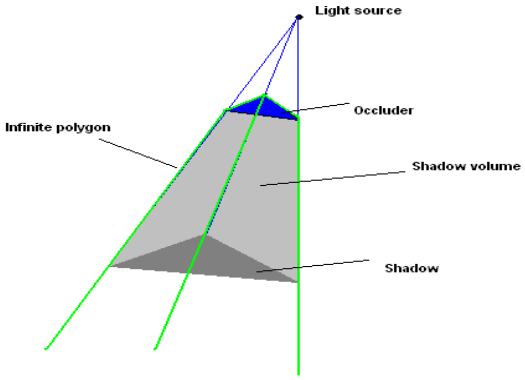

Fig.5. *Shadow volume*

To speed up the proposed algorithm each infinite quadrilateral has clipped to be finite quadrilateral between occluder and shadow receiver. To find a minimum length of volume in each pixel, new pixel is proposed:

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

Now, if an object or part of object is located inside this truncated pyramid, it is in shadow and should be dark but object or part of object located out of this truncated pyramid is in lit and does not need any changes[19]. Then, shadow volume could be generated using stencil buffer and depth buffer together. This algorithm in summary is:

1-Turn off the light and render scene just with ambient and emission lighting.
With this the color buffer will be full for all point of object in shadow and also Z-buffer will be full with depth value.
2-After disabling Z-buffer and Color buffer to write, render scene with lighting.
3-Subtraction of these two depth values provides shadow volume.

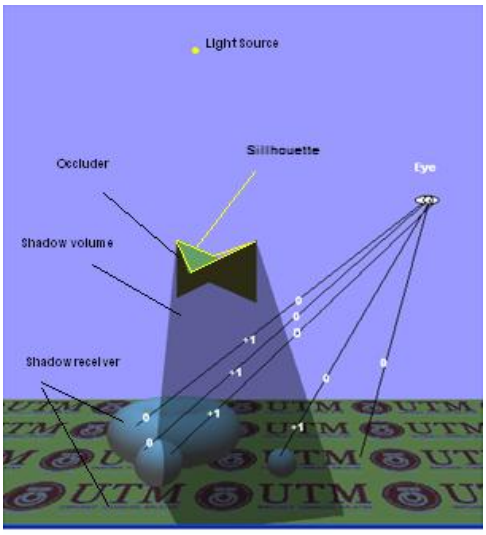

Fig.6. *Shadow volume using stencil buffer and silhouette detection*

After generating shadow volume, we should pass the ray from the eye to each point of object on the shadow receiver. As it mentioned before (in stencil buffer) ,if the ray pass the front face of shadow volume, increase the stencil buffer and when the ray pass the back face of shadow volume, decrease the stencil shadow[20]. Finally, if the number of stencil buffer is not zero it is in shadow. We are going to say this as an algorithm with silhouette detection:

1-Render scene without lighting
2-Enable stencil buffer
3 -Disable depth buffer to write and prevent writing in color buffer.

$$4-\sum_{e=1}^{NE}\sum_{f=1}^{NF} \mathrm{IIf}(\sim Intersection(R_{e_v},\ f),\ \mathrm{Add\ to\ S},\ \mathrm{Nil})$$

$$5-\sum_{i=1}^{NP} IIf(P_i,R_v,\sum_{j=1}^{NP} IIf(ZTest,Stencil++,Nil),Nil)$$

$$6-\sum_{i=1}^{NP} IIf(\sim P_i,R_v,\sum_{j=1}^{NP} IIf(ZTest,Stencil--,Nil),Nil)$$

7-Render scene again with lighting
8-Enable to write in color buffer
9-If (~Stencil % 2)
Keep Stencil

NE is number of edges.
NF is the number of faces.
*N*P is the number of polygon.
S is array of edges.
P_i is polygon.
R_v is a ray from light source to the face with edge *v*.

Z-pass algorithm is used when the eyes are out of shadow. If the eyes are in shadow Z-fail algorithm should be used as follows:
As a matter of fact the whole algorithm is like Z-pass algorithm except step 5 and 6:

$$5-\sum_{i=1}^{NP} IIf(\sim P_i,R_v,\sum_{j=1}^{NP} IIf(\sim ZTest,Stencil++,Nil),Nil)$$

$$6-\sum_{i=1}^{NP} IIf(P_i,R_v,\sum_{j=1}^{NP} IIf(\sim ZTest,Stencil--,Nil),Nil)$$

The step 4 is EESD algorithm. The step 5 is the Z-pass algorithm where ray pass into the volume and as a result stencil buffer is increased. The step 6 is where the ray pass out of volume, then the stencil buffer is decreased.

5. **Performance Estimates**

Assume that there are ***n*** vertices, f_n faces and e_n edges, the number of vertices in each plane is ***m***. Some information is the same for all the three listed algorithms such as calculation of equation of all planes. In this section these three algorithms are compared. It is needed to know the following information about the number of operation:

A dot product needs 3 Multiplies and 2 Adds. A vector or ray addition requires 3 Adds. If there is ***n*** faces, although the number of vertices and edges is not exactly obvious a reasonable number of vertices will be ***2n*** and for this number of faces, the number of edges will be ***6n***[15].

This ratio is based on Chris L. Gorman's [17] idea that has suggested that ratio of vertices to face is 2 and also the ratio of the number of edges to number of vertices is 3.

5.1. Estimate Calculation of Triangular Algorithm

To calculate the visibility of each face, the normal face of plane is needed to be calculated and a ray of light source into the plane is necessary and the sign of dot product should be calculated. To determine if an edge belongs to the array e_n comparison should be done.

For p=1 to Number Of Faces → f_n
If p.visible=True then →*3*Adds+*3*Muls+*4*Muls+*3*Adds+3Mults+
2Adds+*1*Compare+*8*Read
For q=1 to Number Of Side →*m*
If $v_pv_q^{-1}$ is belong to S then →e_n Compare +*1*Inverse +e_n Read

Delete v_pv_q from S →*1* read +*2* write
Else
Add v_pv_q to S →*1* write
End if
Next
} *1* read+*2* write
End if
Next

The number of faces is f_n. To determine whether a plane *p* is visible or not the normal vector of faces should be calculated and it needs 3Adds and 3Mults. To obtain a ray from light source to the plane, it needs 4Mults and 3Adds and finally the dot product needs 3Mults and 2Adds. To determine if the dot product is positive or negative, one comparison is required.

The average number of sides for each plane is considered as *m*. To determine whether $v_pv_q^{-1}$ belongs to *S* or not, one Inverse or 1Mults should be calculated and follow that search in the array which needs e_n Comps and all of them should also be read.

Sum of the all calculations are:

$f_n(8A+10M+1C+8R+(m(e_nC+e_n\ R+1M+1R+2W)))$
$=n(8A+10M+1C+8R+6nmC+6nmR+1mM+1mR+2mW)$
$=6n^2mC+(6n^2m+\ mn+8n)R+(m+10)nM+8nA+2mnW$
$Complexity=O(n^2)$

5.2. Estimate Calculation of Visible-Non visible Algorithm

Visibility of planes is important in this algorithm. To calculate the visibility of each face, the normal face of plane is needed to calculate and also a ray of light source into the plane is necessary and sign of dot product should be calculated.

For E=1 to Number Of Edges →e_n
For F=1 to Number Of Faces →f_n
If F.visible =True then →*3*Adds+*3*Muls+*4*Muls+*3*Adds+3Mults+
2Adds+*1*Compare+*8*Read
E_ counter =E_counter+1 →*1* Add
End if
Next
If e_counter=1 then →*1* Compare + *1* Read
Add E to S → *1* Write
Next

Sum of the all calculations is:

$e*(f*(10M+8A+1C+8R+1A)+1C+1R+1W)$
$= 6n*(n*(10M+9A+1C+8R)+1C+1R+1W)$

$=60n^2M+54n^2A+(6n^2+6n)C+(48n^2+6n)R+6nW$

$Complexity=O(n^2)$

5.3. Estimate Calculation of Extended Edge Algorithm

To extend each edge three addition should be calculated. To determine, if a ray has an intersection with a plane or not, requires putting the ray in the plane equation and compare with zero.

For e1=1(v1v2) to Number Of Edges → e_n
  v1'=v1+ Δ_{v1} → 3Add+1Read+1W
  R=ray of light source to v1' → 3Add+3Multiple+1Read+1W
  for f=1 to Number of Faces → f_n
    if Intersection(R,f)=false then → 3Multiple+3Add+1Compare
      Add v1v2 to S → 1Add
  End if
  Next
Next

A ray is $x=x_0+\alpha n$. It needs 3Mults and 3Adds. The number of faces is f_n and to calculate status of intersection the ray R should hold on to plane f. if the equation of plane is $ax+by+cz+d=0$ then if $aR_x+bR_y+cR_z+d=0$ then the ray intersected the plane.

Sum of calculation is:

$e(6A+3M+2R+2W+f(3M+3A+1C)$
$=6n(6A+3M+2R+2W+n(3M+3A+1C))$
$=(18n^2+18n)M+(18n^2+36n)A+6n^2C+12nR+12nW$
$Complexity=O(n^2)$

6. Results and Discussion

To compare these algorithms, assume that the number of vertices on the corner of occluder is 10 and the number of vertices in each plane is 5. With these date as inputs, the number of operation will be as information in following:

Table 1. *Complexity with n=10,m=5*

n=10,m=5	Multiple	Add	Compare	Read	Write
Triangle Algorithm	150	80	3000	3110	100
Visible-non visible Algorithm	6000	5400	660	4860	60
Extended Edge Algorithm	1980	2160	600	120	120

With increasing of the n and m the result will be more obvious:

Table 2. *Complexity with n=100,m=10*

n=100,m=10	Multiple	Add	Compare	Read	Write
Triangle Algorithm	2000	800	600000	601800	2000
Visible-non visible Algorithm	600000	540000	60600	48600	600
Extended Edge Algorithm	181800	183600	60000	1200	1200

The following charts illustrate the comparison between EESD algorithm with the other algorithms with ***n*** vertices , f_n faces, e_n edges and ***m*** vertices in each face.

The two charts illustrate that with an increase of edges, time consumed by the new algorithm is lesser than the other algorithms and as a result the speed of the newly proposed algorithm is better than the others.

By consideration of cycling in CPU, Mults 4 cycles, Adds 2 cycles, Compares 3 cycles, Reads 1 cycle and Writes 2 cycles:

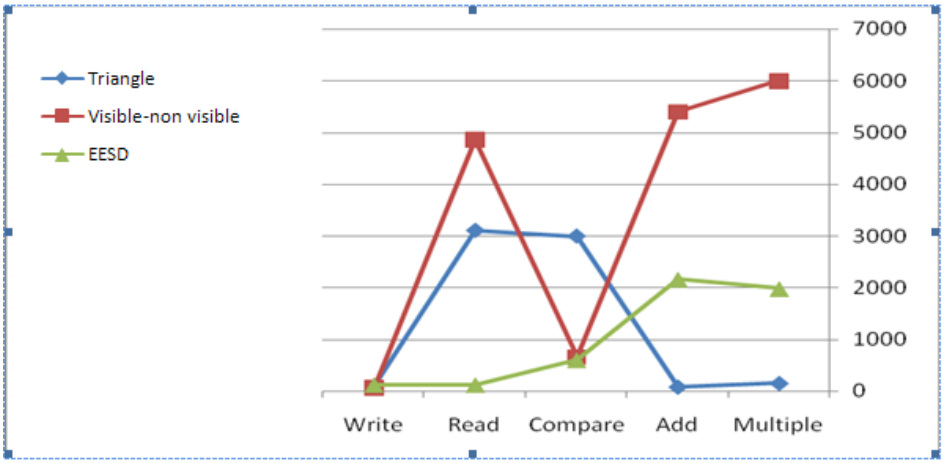

Fig.7. *Complexity with n=10,m=5*

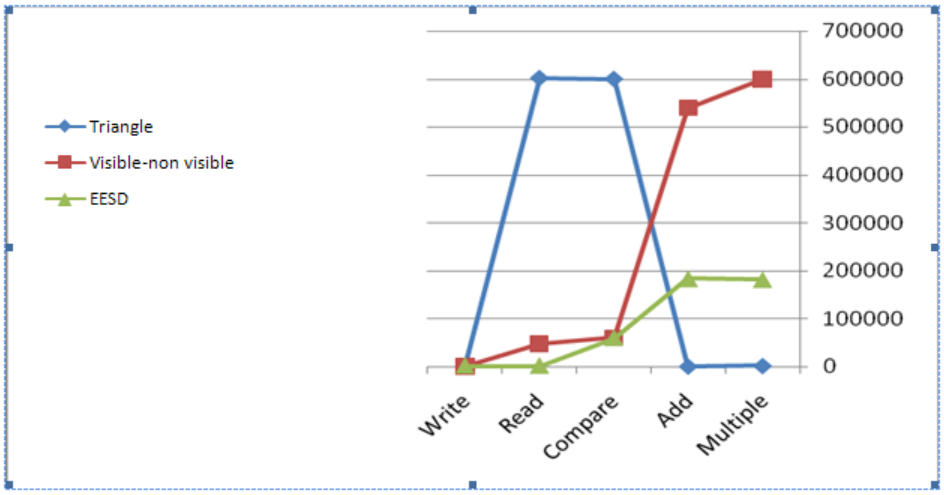

Fig.8. Complexity with n=100,m=10

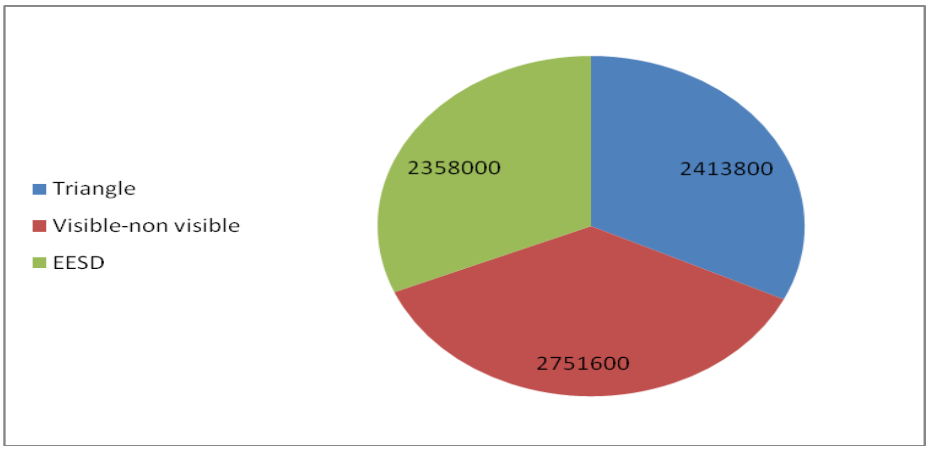

Fig.9. Final comparison between three algorithms

After implement thee algorithms FPS can prove our claim as well as possible. Figure 10 (left) is implementation of shadow volume using Triangular algorithm with average 69.93 FPS with one light source. Figure 10(right) illustrates FPS is 65.51 for shadow volume using Visible-Non Visible algorithm. Finally, Figure 11 shows FPS is 73.16 for shadow volume using EESD algorithm. All algorithms implemented in C++ OpenGL with same version and ran in same computer after turn on computer directly.

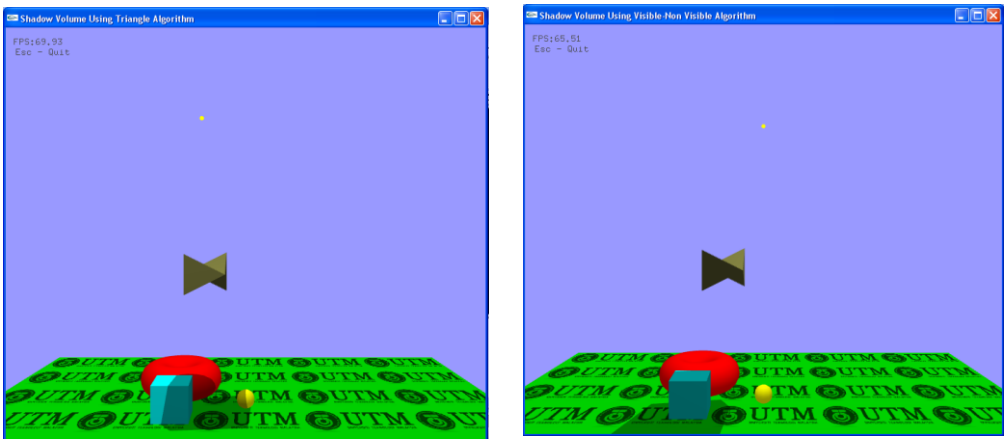

Fig.10. Result shadow volume using Triangle Algorithm and shadow volume using Visible-Non Visible Algorithm

Fig.11.: Result shadow volume using EESD Algorithm

7. Conclusion

Two popular algorithms, Triangular and Visible-non visible algorithm are described. The EESD algorithm is proposed to recognize object's silhouette. The EESD algorithm is used to create real-time shadow volume. In comparison with the other algorithms, it is faster. The base of this algorithm is related to mathematics and visualization. To use this algorithm in shadow volume, stencil buffer and Z- buffer are convenient tools. The benefit of the EESD algorithm is appropriate when the number of occluder's edge increase.

Two methods are used to demonstrate EESD algorithm is faster than the others. Mathematical proving with Big-Oh notation and FPS are used to demonstrate the claim.This algorithm is also convenient for complex scene. The result is good and it is suitable to be used in commercial game and real-time virtual environment.

REFERENCES

[1] W.A. Richards, J.J. Koenderink, D.D. Hoffman, Infer-ring three-dimensional shapes from two-dimensional silhouettes, Journal of the Optical Society of America, 4:1168-1175 (1987).

[2] T. Saito, T. Takahashi, Comprehensible rendering of 3-D shapes, ACM SIGGRAPH Computer Graphics (ACM Press) 24:4, 197-206, (1990)

[3] D. Scherzer, M. Wimmer, W. Purgathofer, A Survey of Real-Time Hard Shadow Mapping Methods, Computer Graphics Forum, 30: 169-186 (2011).

[4] R. Raskar, M. Cohen, Image precision silhouette edges, In: Spencer SN, editor. Proceedings of the conference on the 1999 symposium on interactive 3D graphics, New York: ACM Press, 135-40 (1999).

[5] B. Gooch, P.P.J. Sloan, A. Gooch, P. Shirley, R. Riesenfeld, Interactive Technical Illustration, In Proceedings of the 1999 Symposium on Interactive 3D Graphics, Atlanta, Georgia, United States, 31-38 (1999).
[6] M.C.P. Matos, M.A.M. Ferreira, M. Andrade, Code Form Game, International Journal of Academic Research, 2:1, 135-141 (2010)
[7] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, T. Strothotte, A developer's guide to silhouette algorithms for polygonal models, IEEE CG & A, 28-37 (2003).
[8] F. Benichou, G. Elber, Output sensitive extraction of silhouettes from polygonal geometry, In Proc. of Pacific Graphics 1999, 60-69 (1999).
[9] J. Buchanan, M. Sousa, The edge buffer: A data structure for easy silhouette rendering, In Proc. of NPAR 2000, 39-42 (2000).
[10] S.K. Jung, S.I Kwon, K.J. Kim, Real-time Silhouette Extraction Using Hierarchical Face Clusters, 1-8 (2004).
[11] M. Olson, provide a site about all papers in silhouette entitle A Survey of Silhouette Papers, http://www.cs.sfu.ca/~matto/personal/sil_papers.html , (2010).
[12] H.C. Btagelo, I.C. Junior, Real-Time Shadow Generation using BSP Trees and Stencil Buffers, XII Brazilian Symposium on Computer Graphics and Image Processing, 93-102 (1999).
[13] F. Crow, Shadow Algorithms for Computer Graphics, Computer Graphics, 11(2): 242-247 (1977).
[14] T. Heidmann, Real Shadows Real Time, IRIS Universe, 18: 28-31 (1991).
[15] D.A. Hostetler, Silhouette Edge Detection Algorithms for use with 3D Models, Graphics Algorithms and 3D Technologies (G3D) Intel Architecture Labs (IAL), (2002).
[16] M. Olson, H. Zhang, Silhouette Extraction in Hough Space, Proceedings of Eurographics, 273-282 (2006).
[17] http://www.linkedin.com/in/clgorman, (2011).
[18] Mustafa, S. Fawad, Member, IACSIT, LOD based Real Time Shadow Generation for Subdivison Surfaces, International Journal of Computer and Electrical Engineering, 2:1, 170-174 (2010).
[19] A.B.M. Azahar, M.S. Sunar, D. Daman, A. Bade, Survey on real-time crowds simulation, Technologies For E-Learning and Digital Entertainment, Proceedings , 5093: 573-580 (2008).
[20] H. Kolivand, M.S. Sunar., A. Amirshakarami, Z Ranjbari, Real-Time Volume Shadow using Visible-Non Visible Algorithm, Journal of Computer Science, 7:7, 980-985 (2011).
[21] H. Kolivand, M.S. Sunar, Silhouette Detection for Real-Time Shadow in Virtual Environments, The International Journal of Virtual Reality, 10(3): 45-52 (2011).
[22] H. Kolivand, M.S. Sunar, To Combine Silhouette Detection and Stencil Buffer for Generating Real-Time Shadow, International Journal of Computer Graphics, 2(1): 1-8 (2011).