Shadow Casting with Stencil Buffer for Real-Time Rendering

Lee Weng, Daut Daman, and Mohd Rahim
Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia

Abstract: *We present a new method for real-time rendering of soft shadows in dynamic scenes. Our approach is based on shadow volume algorithm which provides fast, accurate and high quality shadows. The shadow volume algorithm is used to generate hard shadows before adding fake soft shadows onto it. Although the generated soft shadows are physically inaccurate, this method provides soft shadows that are smooth and perceptually convincing. This proposed hybrid method adds more realism to a dynamic scene which is an important factor in computer graphics.*

Keywords: *Shadow volume, silhouette detection, rendering, depth-pass, soft shadow*.

Received November 14, 2006; accepted May 31, 2007

1. Introduction

Shadows are essential elements to realistic and visually appealing images, but are difficult to compute in most display environments especially in computer games. Similar to lighting, there are increasing levels of realism possible, paid for with decreasing levels of rendering performance. Since the introduction of shadow volume [7], shadow map [14] and fake shadows [5], there have been numerous developments done to improve shadow algorithm in real-time graphic application. Among current issues concerns real-time dynamic soft shadows and hardware improvement that improvise real-time shadow generation.

The important element in shadows is the dynamic and accuracy of the hard shadow as it provides information and spatial cue while the soft shadow determines the type of light source. Thus in this research, we attempt to create an accurate real-time dynamic fake soft shadow, where the hard shadow will be accurate and dynamic and the soft shadow will be fake. Stencil shadow volume algorithm will be combined with plateaus soft shadow in order to create the dynamic fake soft shadow.

2. Related Work

Early work on shadows bring us back to 1977 where Frank Crow first published his paper on shadow algorithm for computer graphics [7], in which his method explicitly clip shadows geometry to the view frustums, generating perfect caps where the volume crosses a clipping plane. [11] suggested the use of stencil buffer in implementing crow's original algorithm which gave the algorithm the name by which it is best known today. Stencil shadows belong to the group of volumetric shadow algorithms as the shadowed volume in the scene is explicit in the algorithm. In 2000, Carmack suggested a slightly different approach which entails that the view rays are traced from infinity towards the eye, stopping when encountering the pixel on the geometry that is closest to the eye [6]. This reversal of the view rays' direction has given the algorithm the name Carmack's Reverse. [13] created a hybrid algorithm that uses a faster z-pass rendering when the viewport is not shadowed and reverts to robust z-fail rendering when the viewport is shadowed. Following this, several other shadow volume improvements have also been suggested; papers [1, 2, 3] for example described how to create soft shadows using penumbra wedges rendered from shadow volume. [9] developed a technique for highly efficient coverage calculation for spherical light sources. The technique can avoid clipping operations in the pixel shader and let the texture sampler do the clipping for free. The only setback is the technique is limited to spherical shaped light source only.

3. Algorithm

Our proposed method combines the existing stencil shadow volume algorithm with the concepts of Heckbert and Herf's soft shadow technique [10] which was originally for shadow map. This research was divided into two important steps, as shown in Figure 1.

3.1. Step 1: Creating Hard Shadow - Shadow Volume

A shadow volume for an object and light is the volume of space that is shadowed. That is, all points in the volume are in the shadow for that light. In

generating shadow volume, the first step is to create the shadow volume using the silhouette edges of shadowing object/ occluder as seen by the light source. The edges are then extruded away from the light as shown in Figure 2.

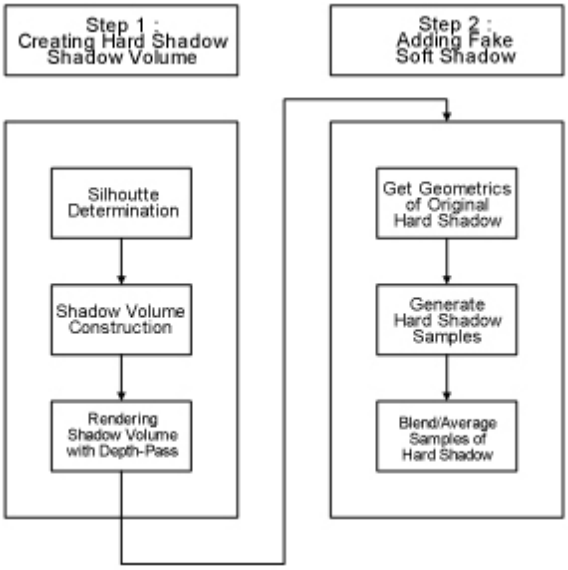

Figure 1. Research methodology.

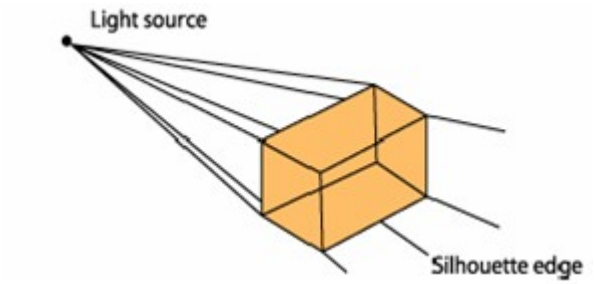

Figure 2. Silhouette edge.

Next, as shown in Figure 3, the shadow volume is clipped to the view/ camera volume, and forms the polygons that bound the shadow volume. The final result is a set of shadow volume boundary polygons with all points within the shadow volume are in the shadow. Along a ray from the eye, we can track the shadow state by looking at intersections with shadow volume boundaries following the rules below (assume the eye is not in shadow):

- Each time the ray crosses a front facing shadow polygon, add one to a counter.
- Each time the ray crosses a back-facing shadow polygon, subtract one from a counter.
- Places where the counter is zero are lit, others are shadowed.

The algorithm to implement stencil shadow volumes is summarized as the following [12]:

A. Render all the objects using only ambient lighting and any other surface-shading attribute. Rendering should not depend on any particular light source. Make sure depth buffer is written.
B. Starting with a light source, clear the stencil buffer and calculate the silhouette of all the occluders with respect to the light source.
C. Extrude the silhouette away from the light source to an infinite distance to form the shadow.
D. Render the shadow volumes using the depth-pass.
E. Using the updated stencil buffer, do a lighting pass to shade (make it a tone darker) the fragments that corresponds to non-zero stencil values.
F. Repeat step 2 to 5 for all the lights in the scene.

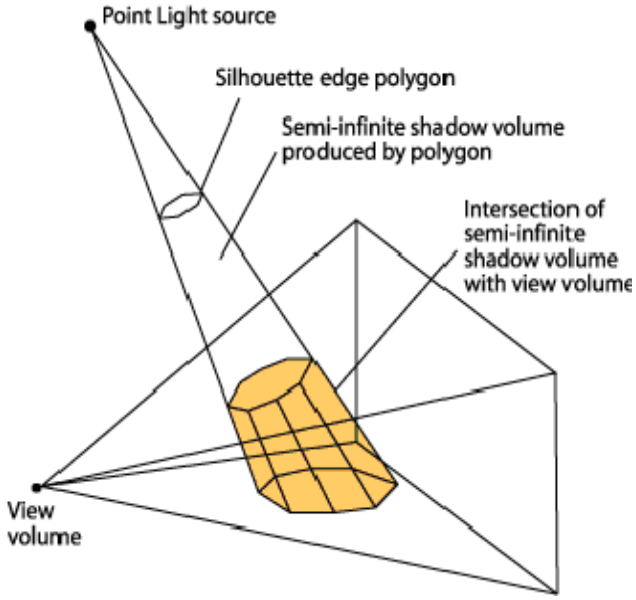

Figure 3. Shadow volume clipping with view volume.

From the above list of steps, it should be quite obvious that having more lights leads to having more passes, which can increase frame rate intensity. Thus, we have to be very selective when deciding which lights should be used for casting shadows.

3.1.1. Silhouette Determination

As stated before, the very first step to construct a shadow volume is to determine the silhouette of the occluder. The stencil shadow algorithm requires that the occluders be closed triangle meshes. This means that every edge in the model must only be shared by 2 triangles thus disallowing any holes that would expose the interior of the model. There are many ways to calculate the silhouette edges and every single one of these methods are CPU cycles hungry.

Edge connectivity information must be pre-computed so that we can determine a mesh's silhouette for shadow volume rendering. The method used here can be explained by using an array of N vertices V_1, V_2 ..,V_N and an array of M triangle faces F_1, F_2,.., F_M. Each triangle faces simply indicate which three vertices it uses by storing three integer indexes i_1, i_2 and i_3. An index i_p precedes an index i_q if the number p immediately precedes the number q in

the cyclic chain 1→2→3→1. The indexes i_1, i_2 and i_3 are ordered such that the positions of the vertices ${}^{V}i_1$, ${}^{V}i_2$ and ${}^{V}i_3$ to which they refer are wound counter-clockwise about the triangles normal vector. Suppose that two triangles share an edge with endpoints of vertices V_a and V_b. The consistent winding rule enforces the property that for one of the triangle faces, the index referring to V_a precedes the index referring to V_b and that for the other triangle and the index referring to V_b precedes the index referring to V_a.

With this the edges of a triangle mesh can be identified by making a single pass through the triangle face list. For any triangle having vertex indexes i_1, i_2 and i_3, create an edge record for every instance in which $i_1 \rightarrow i_2$, $i_2 \rightarrow i_3$, and $i_3 \rightarrow i_1$ and store the index of the current triangle face in the edge record. Once all the edges are identified, make a second pass through the triangle face list to find the second triangle that shares each edge. This is done by locating triangles for which $i_1 \rightarrow i_2$, $i_2 \rightarrow i_3$, or $i_3 \rightarrow i_1$ and matching it to an edge having the same vertex indexes that has not yet been supplied with a second triangle index. The general concept of the explanation above can be summarized in the following pseudo code:

A. for each triangle face (A) in the object/ model
B. for each edge in A
C. if we don't know this edges triangle face (neighbors) yet
D. for each triangle face (B) in the object/ model except A
E. for each edge in B
F. if A's edge is the same as B's edge, then they are neighboring each other on that edge, set the neighbor property for each triangle face A and B, then move onto next edge in A.

With the edge list of a triangle mesh/ face, the silhouette is determined by substituting the light position with the plane equation. A triangle face that is visible to light source will have a value of plane equation>0 otherwise the triangle face is not visible from light source. The silhouette is equal to the set of edges shared by one triangle face that is visible by the light and one triangle face that is not visible by the light. This is done by going through all the triangle faces and if it is visible, the edges are checked. If at the edge, there is no neighboring triangle face or the neighboring triangle face is not visible to light source then this edge is a silhouette and it cast shadow.

It is important to note that silhouette determination is one of the two most expensive operations in stencil shadow volume implementation. The other operation is the shadow volume rendering passes to update the stencil buffer. These two areas are the prime candidates for aggressive optimizations.

3.1.2. Shadow Volume Construction

Once the set of an object's silhouette edges is determined with respect to a light source, each edge must be extruded away from the light source's position to form the object's shadow volume. For a point light source, which was implemented in this research prototype, the extrusion of the silhouette edges consist a set of quads (which can also be substituted with triangle strips). The quads are constructed from the two vertices belonging to an edge and two additional vertices corresponding to the extrusion of the same edge to "infinity" (a large value) by using homogeneous coordinates. Shadow volume is extruded to "infinity" in order to avoid light source being too close to an occluder. If this happens, the situation shown in Figure 4 will occur where a finite shadow volume extrusion fails to cover all the shadow receivers in a scene. However, it is not compulsory to extrude the silhouette edges to infinity if it is ensured that the situation in Figure 4 can be avoided. In practical cases, a large value would normally be more than adequate.

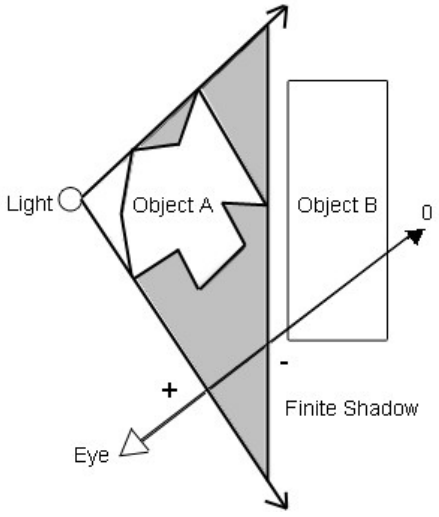

Figure 4. Finite shadow volume fails to shadow other objects.

The extrusion distance is the distance the vertices of the bottom cap of the shadow volume are extruded away from the light. The approach implemented in this research prototype is brute force approach that simply draws the extrusion polygon to "infinity" (a large value) and the shadow volume is just clipped against all the polygon it encounters (refer Figure 5 for illustration). The procedure of silhouette determination and getting the triangle faces that is visible to light source will provide a triangle faces that are situated at the edge of the silhouette, a triangle face with no neighboring triangle face or the neighboring triangle face is not visible to light source (refer Figure 6) which will be called silhouette triangle face from this point on.

With a single silhouette triangle face (marked with black and white lines in Figure 6), edge test are done

so that the edge of the silhouette (the black line edge) can be obtained. That single edge provides two vertices which will be used to extrude another two vertices that will be generated after this. After obtaining the two vertices, the extrusion to "infinity" will be done using homogeneous coordinates. Here a basic scaling transformation is used to extrude the shadow volume by using the two vertices of the silhouette edge. This will produce two new vertices which are projected along the vector between the light source and the first silhouette edge. It is scaled to INFINITY (the variable used in the implemented prototype for "infinity" value) which has been set to a very large value (refer Figure 7).

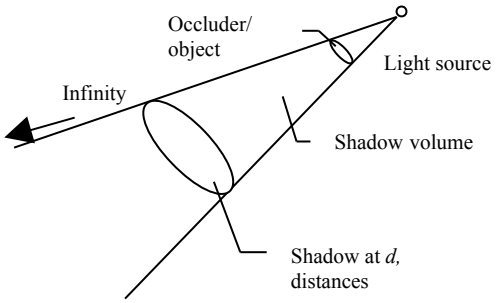

Figure 5. Extrusion option.

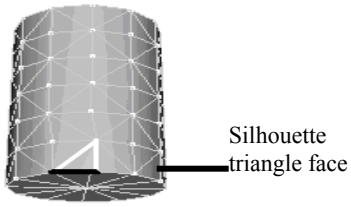

Figure 6. The edge for casting shadow volume.

The scaling transformations used to produce the new vertices are shown below;

$$x' = x_f + (x - x_f)s_x \quad (1)$$

$$y' = y_f + (y - y_f)s_y \quad (2)$$

$$z' = z_f + (z - z_f)s_z \quad (3)$$

Vertices $P'(x', y', z')$ is a new vertex, $L(x, y, z)$ is the light source location and $O(x, y, z)$ is the vertex from the silhouette edge. With two vertices from the silhouette edge, using the above equation, two new vertices are produced and this form the quadrilateral (in research prototype implementation triangle strips are used) needed to create shadow volume. After creating the shadow volume the next process is to render it so that the hard shadow is visible, which will be explained in the next section.

3.1.3. Rendering Shadow Volume with Depth Pass

Depth-pass is also commonly known as z-pass. Let's assume that the objects had already been rendered onto the frame buffer prior to the above stenciling operations. This means that the depth buffer would have been set with the correct values for depth testing or z-testing. Referring to Figure 8, the two leftmost ray originating from the eye position does not hit any part of the shadow volume (in grey), hence the resultant stencil values is 0, which means that the fragment represented by this two rays are not in shadow. The 3rd ray from the left, when the front face of the shadow volume is rendered, the depth test would pass and the stencil value would be incremented to 1. However when the back face of the shadow volume is rendered, the depth test would fail since the back face of the shadow volume is behind the occluder. Thus the stencil value for the fragment represented by this ray remains at 1. This means that the fragment is in shadow since its stencil value is non-zero.

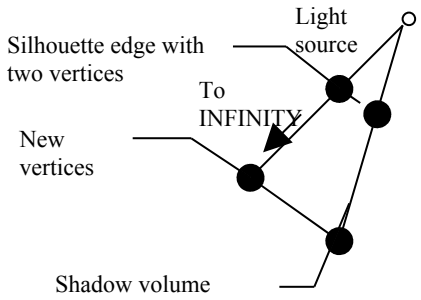

Figure 7. Extruding to INFINITY by producing two new vertices.

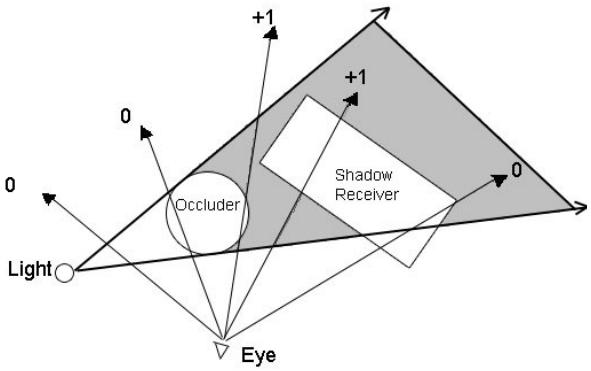

Figure 8. Depth-pass.

After determining the object's silhouette with respect to a light source and constructing a shadow volume by extruding the silhouette edges away from the light source, finally the shadow volume is ready to be rendered into stencil buffer using depth-pass technique. The frame buffer is cleared and an ambient rendering pass is performed to initialize the depth buffer. Lighting is disabled because there will be no rendering to the colour buffer except the stencil buffer.

Then the stencil buffer is cleared, initialized and configured so that it always passes (which is why it is called depth-pass technique). The depth test is configured to always pass only when fragment depth values are less or equal than those already in the depth

buffer. The drawing will only be done into the stencil buffer, write to colour buffer and depth buffer are disabled so that shadow volume don't appear as solid objects in the depth buffer.

Shadow volume faces which were constructed as described earlier are rendered using different stencil operations depending on whether they are facing towards or away from the camera. It is rendered in two passes, first pass increase the stencil buffer with front faces (casting shadow) and the second pass decrease the stencil buffer with back faces ("turning off" the shadow between the object and any other surfaces).

Once shadow volumes have been rendered for all objects that could potentially cast shadows into the visible region of the scene, it will cause all the areas that are in the shadow volume to have a non-zero stencil value while all those areas in the light area remain zero. Lighting pass is performed to illuminate surfaces where the stencil value remain zero, re-enable write to the colour buffer, change the depth test to pass only when fragment depth values are equal or less to those in the depth buffer and configure the stencil test to pass when the value in stencil buffer is not equal to zero. Finally, a blend drawn over the whole screen will cast a shadow which is how the hard shadow is produced. This fulfils the first objective of generating accurate hard shadow. The technique is known as depth-pass technique since it manipulates the stencil values only when the depth test passes, and below is the general overview of the algorithm;

A. Render front face of shadow volume. If depth test passes, increment stencil value, else does nothing. Disable draw to frame and depth buffer.
B. Render back face of shadow volume. If depth test passes, decrement stencil value, else does nothing. Disable draw to frame and depth buffer.

3.2. Step 2: Adding Fake Soft Shadows

Adding fake soft shadows to existing shadows generated by shadow volume will increase the realism of the shadow in a 3D scene. Our proposed method in implementing soft shadows in shadow volume was developed originally from the concept from [10] along with the well-known stencil shadow volume. The algorithm by [10] had been previously implemented in shadow map algorithm where the result proved to be quite convincing even though it is not a conventional method. The algorithm was implemented at interactive rates by exploiting graphics workstation hardware. Since hardware have become increasingly faster and more affordable, nowadays this technique is feasible to be implemented on desktop computers with an adequate graphic card.

Our method was developed by methodically addressing the fundamental limitations of the conventional stenciled shadow volume approach towards soft shadow and also the approach taken by previous researchers. The work flow is shown in Figure 9 followed by brief description of each step.

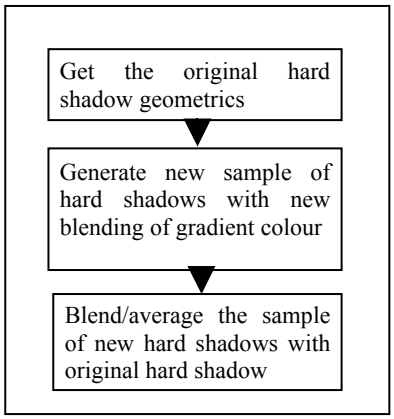

Figure 9. Work flow for adding soft shadows.

3.2.1. Get the Original Hard Shadow Geometrics

The very first step in generating the soft shadows is to obtain the geometrics of the hard shadows generated earlier. This were done by saving all the generated coordinates of the vertices while rendering shadow volume in depth-pass technique so that no re-calculation is needed, saving ample computation cost.

3.2.2. Generate Samples of Hard Shadows

Next is to generate samples of hard shadows with the blending of gradient colours. This can be done by drawing the same shadow volume geometrics obtained earlier only that the size of the shadow volume polygon is scaled to be slightly bigger than the original shadow volume, as shown in Figure 10. The number of samples that needs to be generated depends on how high the required quality of the soft shadows before blending it with gradient of reduced shadow colours.

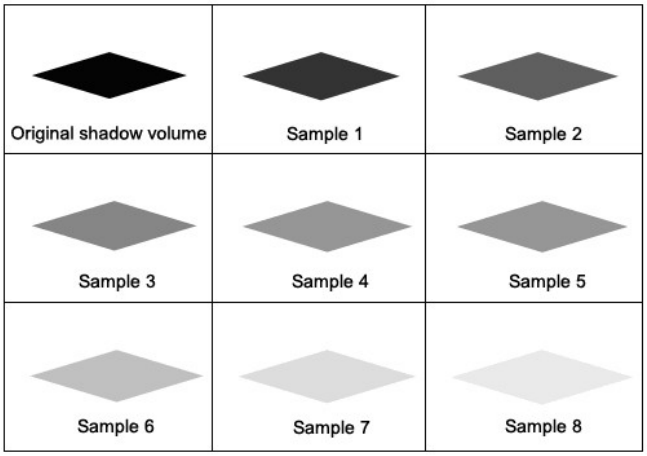

Figure 10. Samples of new hard shadows generated.

While generating these samples doesn't affect much of the processing time but rendering it again with depth-pass and stencil buffer test does involve a

lot of processing. Thus the number of samples generated must be taken into consideration so that it won't slow down the frame rate. Although the more samples generated will produce a higher quality of soft shadows, this will also in turn increase CPU consumption.

3.2.3. Blending Samples of Hard Shadows

After the samples were generated, the last step is to blend or average out the samples together with the original shadow volume. This process is done by stacking the samples on top of one another, starting with the lightest sample to the darkest sample available and finally to the original shadow volume. The illustration is shown in Figure 11.

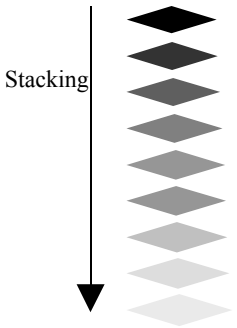

Figure 11. Stacking of the sample and original shadow volume.

The order of the stacking must be done from the less dark to the darkest so that the produced soft shadow will create a penumbrae effect to the edges of the original shadow. As shown in Figure 12, the difference can be seen when compared this with the original shadow volume with only the hard shadow.

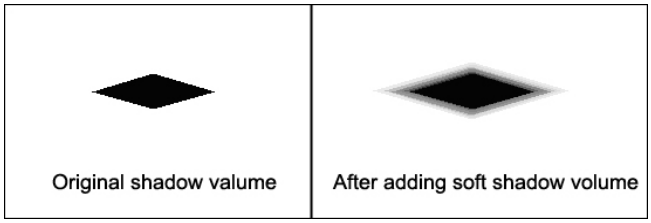

Figure 12. Comparison of original shadow volume and new soft shadow volume.

The creation of soft shadow using this technique accepts two parameters that can differentiate the quality of the soft shadow produced. The first parameter is the length factor which determines how far the penumbrae or the soft shadow will be extended to and the second parameter is the gap factor which determines the gap between the samples of the hard shadows produced (refer Figure 13). The number of samples the hard shadow produced depends on the length and gap factor which is equal to length divided by gap. The produced new soft shadow will have a more realistic and convincing effect compared to the original algorithm, thus fulfilling the second objective of this research.

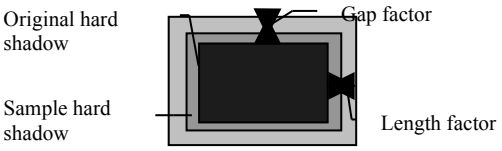

Figure 13. Length and gap factor.

4. Results

As one of the main goals for this research was to test the applicability of soft shadow in a true 3D environment, the prototype implementation of our method and testing stage were done using complex, high polygon models. The tests performed are the accuracy or resemblance test, followed by quality test and lastly real-time test. Side by side visual comparisons with reference examples were used to determine the accuracy of the produced shadow on how far it resembles real life shadow. This comparison approach was also applied to measure the quality of the produced fake soft shadows. Speed test were performed by observing the frame rate and comparing it to reference examples. The running time of the algorithms depends on factors such as number of polygons, desired quality, screen resolution, graphics hardware and also the speed of the CPU.

4.1. Accuracy or Resemblance Test

The shadows generated by the prototype are guaranteed true, real and accurate as it uses the model or shadow caster geometric to produce the shadow. Moreover there is no model simplification done to optimize rendering. By providing the graphic images of the implemented shadow below, it is shown that the shadow in this research prototype is accurate and furthermore resembles the model/ shadow caster, as shown in Figure 14.

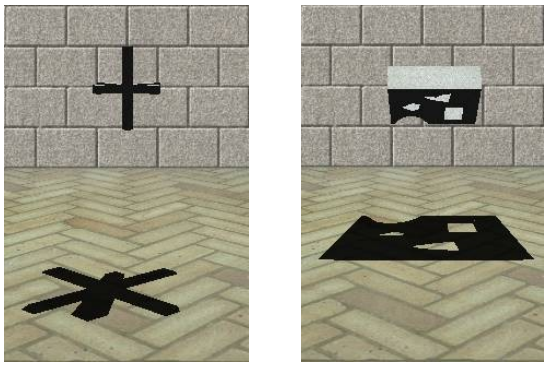

Figure 14. Resemblance test.

4.2. Quality Test

The test involved rendering a cube with 8 vertices and 12 faces of triangle polygons using different values of length and gap factors as parameters. Images from test results were captured and the number of polygon

triangles produced was recorded so that comparison and analysis can be done to evaluate the quality of the soft shadow produced. Figure 15 illustrate the result.

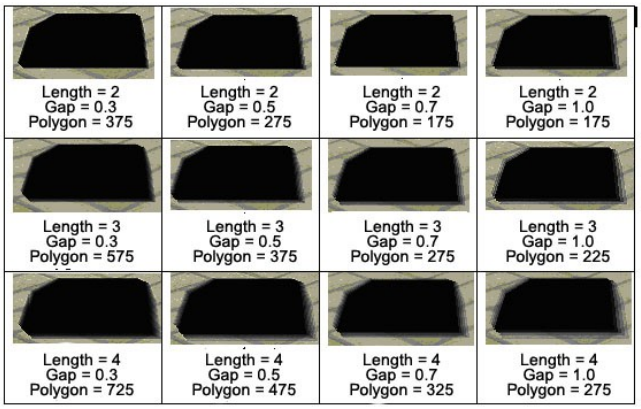

Figure 15. Test results using cube with different length and gap.

The test results achieved a frame per second rate of 60, which can be deemed as a good run time rate. The captured images also show that with increased number of rendered polygon triangles, along with smaller gap and greater length parameters will produce a better quality of soft shadow. Although these parameters provided a good blending of the shadow with the environment, some drawbacks were also found. The bigger the gap is the more aliasing effect will be visible to the image, as shown in the images with 1.0 gap factor in Figure 15. Greater length also causes more changes to the geometric shape of the shadow.

Thus it is concluded from this test that for this model or shadow caster the appropriate value for parameter length should be between 2 to 3, as any different value will cause dramatic changes in geometric shape. Moreover, the optimized gap factor value for this cube should be anywhere between 0.3 to 0.5, as the produced soft shadow will not be visible to the human eyes if any different values were used, beside causing aliasing effect. This also applies to model or shadow caster other than cubes which was not shown here. We also conduct tests by visually comparing the shadow images rendered by our research prototype in comparison to other existing algorithm. We choose the shadow volume algorithm [1], known as "Approximate Soft Shadow on Arbitrary Surfaces Using Penumbra Wedge" to be our reference example. The chosen model or shadow caster was a cube and sphere; this was done without any other model in the scene except the wall and floor.

The result showed that our prototype were able to render at about 60 FPS which was about 6 time faster than Ulf and Tomas algorithm although the quality of our produced soft shadow is undeniably lesser. However, the soft shadows produced were still convincingly realistic on top of being able to render at real-time speed. The comparison is shown in Figure 16.

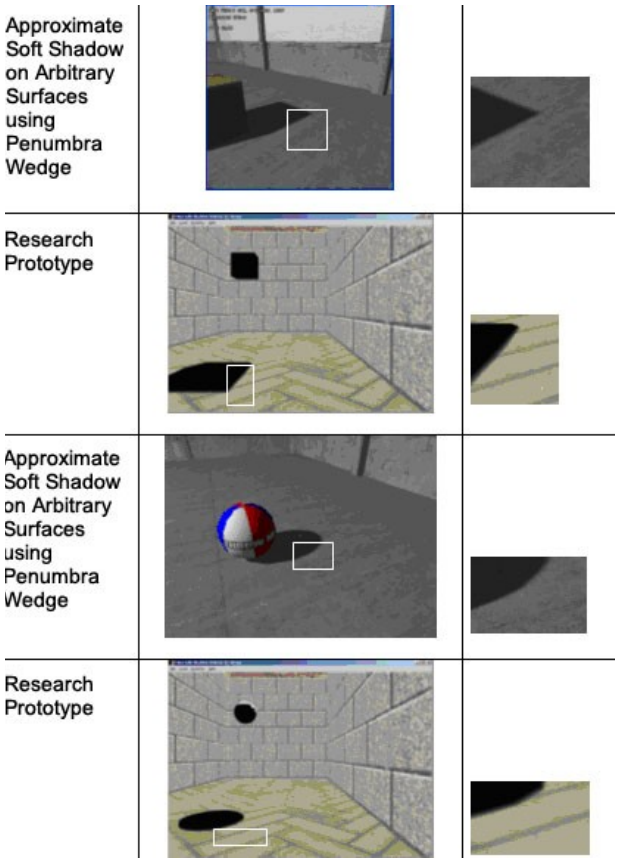

Figure 16. Comparison of soft shadow generated.

4.3. **Real-Time Test**

One way to obtain a fast soft shadow algorithm is to utilize graphics hardware. According to Moore's Law, the speed of graphics hardware has so far more than doubled every 12 months, compare to CPUs which double approximately every 18 months. As shadow volume requires a lot of computations especially in silhouette edge determination and two-pass rendering, it is very high consuming for both CPU and GPU processor. Here, the shadow volume algorithm developed by Ulf and Tomas will again be used as a reference example to compare its speed with our produced algorithm.

Both algorithms will be tested against three different hardware systems to determine the best system available to the public. Our proposed method was tested using the same model or shadow caster that was used in Ulf and Tomas implementation. The quality of the soft shadow produced by both algorithms was identically leveled with parameters set to 3.0 for length and 0.3 for gap. Identical resolution at 800x600 with 32 bit colours was also set as the environment for both algorithms. This is to ensure that the same amounts of polygon are used to render the soft shadow so that a precise comparison can be done.

Two graphs were produced at the end of the test shown here as Figure 17 and 18. Figure 17 shows the result of the test using our proposed method where system 2 is shown to be the best system, followed by system 3 and system 1.

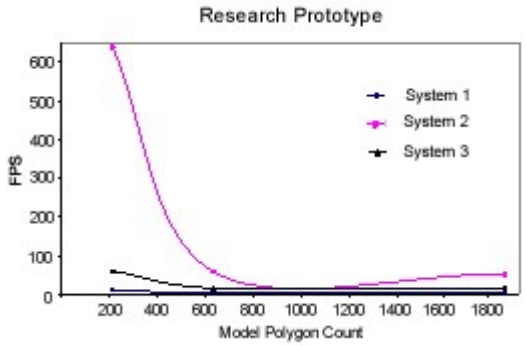

Figure 17. FPS vs. model polygon count for 3 different systems using fake soft shadow volume with stencil buffer.

From here it could be concluded that system 1 requires a much better graphics hardware and system as it can only achieve 10 FPS with 216 polygons and will dropped even further to 3 FPS if the numbers of polygon exceed 500. System 2 only have 1.5 GHz CPU and 768 MB RAM, much lower than system 3 which have a 2.4 GHz CPU and 1 GB RAM but yet the graphic card used in system 2 was way too powerful and was able to render soft shadow at almost 640 FPS for 200 polygons.

However, it also gradually goes down to 50-60 FPS after the number of polygon exceeds 500 as the object geometric become complex. System 3 is a mid range system which is able to render soft shadow at 60 FPS for polygons below 250. After 500 of triangle polygons the FPS remain static at 15 FPS even though the number of polygons reached 1800.

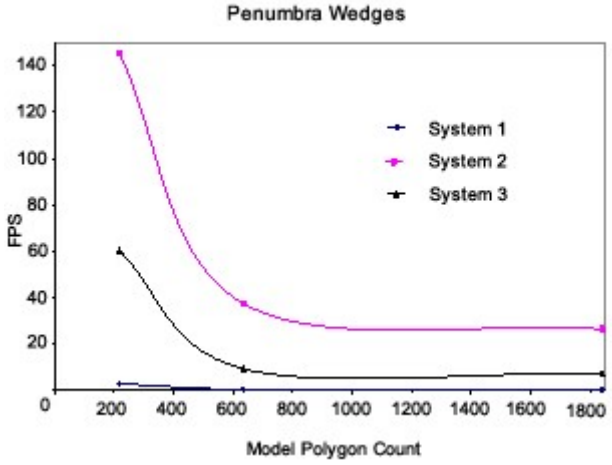

Figure 18. FPS vs. model polygon count for 3 different systems with approximate soft shadows on arbitrary surfaces using penumbra wedges algorithm.

Figure 18 above is the result of the test using Ulf and Tomas soft shadow algorithm where in comparison to the graph in Figure 17, shows a much slower rate of FPS. From the graph it is shown that the maximum FPS is not very high but as the number of triangle polygons increase, the decreasing of FPS is not as low as the statistic shown in Figure 17. The graph had shown that it only decreased almost two to three times in FPS as the number of triangle polygon count increased three times. However, in terms of speed this test has shown that our proposed method performed much better than Ulf and Tomas soft shadow algorithm as it has at least matches the FPS or even faster, which at one point it was eleven times faster.

5. Conclusions and Future Work

In this paper we have shown a method to generate an accurate hard shadow volume and adding fake soft shadows onto it. Among clear benefits is that the method produces high quality soft shadows at high speed. This method can be further improved and be implemented using programmable graphics hardware to achieve real-time performance.

An important aspect yet to be incorporated is how to extend the effect of shadows onto other surfaces and objects in the scene other than planar/ surface. Thus this area is something we would like to explore in our future work in attempts to further improve efficiency. Problems concerning shadow volumes that intersect at near or far clipping plane were not discussed in this paper as the solution were already presented in [8].

References

[1] Akenine-Moller T. and Assarsson U., "Approximate Soft Shadows on Arbitrary Surfaces Using Penumbra Wedges," *in Proceedings of the 13th Eurographics Workshop on Rendering*, Eurographics Association, pp. 297-306, 2002.

[2] Assarsson U. and Akenine-Moller T., "A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 511-520, 2003.

[3] Assarsson U., Dougherty M., Mounier M., and Akenine-Möller T. , "An Optimized Soft Shadow Volume Algorithm with Real-Time Performance," *in Proceedings of ACM SIGGRAPH/ EUROGRAPHICS Conference on Graphics Hardware*, Eurographics Association, pp. 33-40, 2003.

[4] Blinn J. and Corner J., *A Trip Down the Graphics Pipeline*, Morgan Kaufmann Publishers, San Francisco, 1996.

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

[6] Carmack J., "John carmack on shadow volumes," http://developer.nvidia.com/ object/ robustshadow - volumes.html, 2000.

[7] Crow F., "Shadow Algorithms for Computer Graphics," *ACM SIGGRAPH Computer*

Graphics (SIGGRAPH'77), vol. 11, no. 3, pp. 242-248, 1977.

[8] Everitt C. and Killgard M., "Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering," *Technical Report*, NVIDIA Cooperation, 2002.

[9] Fauerby K. and Kjaer C., "Real-Time Soft Shadows in a Game Engine," *Master's Thesis*, 2003.

[10] Heckbert P. and Herf M., "Simulating Soft Shadows with Graphics Hardware," *Technical Report CMU-CS-97-104*, Carnegie Mellon University, January 1997.

[11] Heidmann T., "Real Shadows, Real Time," *Iris Universe*, vol. 18, pp. 23-31, Silicon Graphics Inc., 1991

[12] Kwoon H., "The Theory of Stencil Shadow Volumes," http://www.gamedev.net. 2002.

[13] Lengyel E., "The Mechanics of Robust Stencil Shadows," Gamasutra, http://www.gamasutra.com/features/20021011/lengyel_01.htm, 2002.

[14] Williams L., "Casting Curved Shadows on Curved Surfaces," *Computer Graphics*, vol. 12, no. 3, 1978.

Lee Weng received his BSc form Universiti Teknologi Malaysia in computer science with majoring computer graphics. He also receives MSc from University Teknologi, Malaysia, in computer science with research project in computer games. He is currently a senior computer graphics engineer at international company which is Seagate Penang, Malaysia.

Daut Daman received his BSc form University Sains Malaysia and MSc from University of Cranfield, United Kingdom. He is currently an associate professor in the Faculty of Computer Science and Information System of Universiti Teknologi Malaysia. He has over 27 years of experience in the field of computer graphics and visualization. He has also been actively involved in many research projects related to computer graphics and visualization and has published more than 120 publications.

Mohd Rahim received his Diploma in computer science 1997, BSc computer science 1999 form University Teknologi Malaysia and MSc in GIS and visualization in 2002 from University Teknologi Malaysia. Now he is going to complete the PhD (spatial modelling) in University Putra Malaysia. He also is lecturer in Department of Computer Graphic and Multimedia, Faculty of Computer Science and Information System of University Teknologi Malaysia. He has over 8 years of experience in the field of computer graphics and visualization and GIS data model. He has also been actively involved in many research projects related to GIS and visualization, GIS data management, computer graphics and has published more than 70 publications. Currently his research was on going in spatial, temporal and spatiotemporal data management applications, 3D data visualization, mobile computing for computer graphics application.