Stencil Shadow Volume Algorithms:

An Analysis and Enhancement

Pierre Rautenbach[1], Vreda Pieterse[2], Derrick G Kourie[3]

Department of Computer Science
School of Engineering
University of Pretoria
South Africa
[1]prautenbach@cs.up.ac.za [2]vpieterse@cs.up.ac.za [3]dkourie@cs.up.ac.za

ABSTRACT. A critical analysis and improvement of shadow volume algorithms are presented. This analysis allows for the isolation of key algorithmic weaknesses and possible bottleneck areas. Focusing on these bottleneck areas, we investigate several possibilities of improving the performance and quality of shadow rendering; both on a hardware and software level. Primary performance benefits are seen through effective culling, clipping, the use of hardware extensions and by managing the polygonal complexity and silhouette detection of shadow casting meshes. Additional performance gains are achieved by combining the depth-fail stencil shadow volume algorithm with dynamic spatial subdivision.

RÉSUMÉ. Une analyse critique et une amélioration des algorithmes de volume d'ombre sont présentés. Cette analyse permet à l'isolement des principales faiblesses algorithmique et des domaines d'étranglement possible. Mettant l'accent sur ces domaines, nous étudions plusieurs possibilités d'améliorer la performance et la qualité de l'ombre d'équarrissage; aux niveaux hardware et logiciel. Les benefice principaux de performance sont réaliser par faire d'abattage et de détourage, de l'utilisation des extensions de hardware et par la gestion de la complexité polygonale et la de detection des grillage éclairant de l'ombre. Gains de performance supplémentaires sont obtenus par la combinaison de l'algorithme du profondeur-arrêt stencil volumes de l'ombre avec la subdivision dynamique spatial.

KEYWORDS : Stencil Shadow Volumes, Real-time shadows, Shadow algorithms, Spatial subdivision, Octtrees.

MOTS-CLÉS : Algorithmes de volume d'ombre, Temps réel des ombres, Subdivision dynamique spatial, Octtrees.

1. Introduction

Real-time shadow generation contributes heavily towards the realism and ambience of any scene being rendered. Research regarding the calculation of shadows has been conducted since the late 60's and has picked up great momentum with the evolution of high-end dedicated graphics hardware.

Shadows are produced by opaque or semi-opaque objects obstructing light from reaching other objects or surfaces. A shadow is a two-dimensional projection of at least one object onto another object or surface. The size of a shadow is dependent on the angle between the light vector and light-blocking object [3]. The intensity of a shadow is in turn influenced by the opacity of the light-blocking object [3]. An opaque object is completely impenetrable to light and will thus cast a darker shadow than a semi-opaque object. The number of light sources will also affect the number of shadows in a scene, with the darkness of a shadow intensifying when multiple shadows overlap.

The drive towards realism has led to the development of many shadowing algorithms, with the success of an algorithm heavily dependent on the balance between speed and realism [4]. Hardware has also become particularly amenable to the support of a number of shadow rendering techniques - thus freeing the CPU of a significant processing burden and making the real-time rendering of shadows feasible [6].

Section 2 describes previous work that forms the background to this current research. In section 3, criteria used for the evaluation of stencil shadow volume rendering are proposed. Section 4 then presents the results obtained through several performance and quality tests based on these criteria. Section 5 proposes the use of dynamic spatial subdivision to improve performance. The empirical results of these proposals are given in section 6, followed by a number of conclusions in section 7.

2. Previous Work

The idea of creating shadows by computing shadow volumes was first proposed by Frank Crow [3]. A shadow volume designates a 3D-space that is obscured from one or more light sources by occluding object(s). When the obscuring objects are represented by polygon meshes, then the boundaries (or surfaces) of the associated shadow volumes can also be represented by polygons.

Crowe's ideas were later extended by Bergeron [2] to include non-planar polygons and objects. There have been several attempts at improving the performance of shadow representations based on shadow volume computations, the most successful approaches making efficient use of graphics hardware. For example, Heidmann [4] relies on the hardware's stencil buffer to implement Crow's core ideas, resulting in the so-called depth-pass stencil shadow volume algorithm, or simply, the depth-pass algorithm.

There are two variations to the depth-pass algorithm, namely the depth-fail algorithm and the exclusive-or algorithm. These three algorithms (only the first two of which are discussed below) each involve the computation of shadow volumes, including the determination of front- and back-facing boundary surfaces - front and back being defined in relation to the viewer. In general, each such surface constitutes a polygon.

The algorithms differ principally in the way they utilise these back- and front-facing polygon surfaces to compute a mask (stored in the hardware's stencil buffer). This stencil mask is used to indicate the pixels in shadow. In outline, these algorithms, referred to generically as stencil shadow volume algorithms, work as follows: *1) For each rasterised fragment, render the fragments using ambient lighting, updating the Z-buffer after each fragment has been rendered. 2) Determine the silhouette edges of shadow casting objects. By protruding light rays from the relevant light source, past each vertex of such an edge, the required shadow volume polygons (shadow surfaces) can be found. These shadow volume polygons and their associated front- and back-facing attributes (relative to the viewer) are required for each shadow casting object. 3) Next, compute the regions that are in shadow, updating the stencil buffer so that it collectively constitutes a mask, indicating the pixels that are in shadow and those that are not. Fragments with non-zero stencil values are*

considered to be in shadow. 4) Consider each rasterised fragment, and render the fragment as partially lit if in shadow.

In the case of the depth-pass algorithm, values in the stencil buffer are determined by means of the following stencil operations: *1) At each pixel determine if the corresponding depth of a front facing shadow surface is less than the matching value in the Z-buffer. If it is, then that pixel has 'passed' the depth test, and the corresponding stencil buffer value is incremented by 1. 2) The foregoing step is now repeated, but with back-facing shadow surfaces. Thus, whenever a pixel passes the depth test, then the corresponding stencil buffer value is decremented by 1.*

If the stencil buffer indicates that, at a given pixel, there are more front-facing shadow surfaces than back-facing ones; then a shadow should be projected onto the object whose depth is recorded in the Z-buffer, and vice-versa. However, this algorithm breaks down whenever a viewer is spatially located within a shadow volume. The depth-fail algorithm was devised to correct this so-called stencil counting inversion problem.

The depth-fail algorithm works as a duel of the depth-pass algorithm in the following manner: *1) At each pixel determine if the corresponding depth of a back facing shadow surface is greater than the matching value in the Z-buffer. If it is, then that pixel has 'failed' the depth test, and the corresponding stencil buffer value is incremented by 1. 2) The foregoing step is now repeated, but with front-facing shadow surfaces. Similarly, the corresponding stencil buffer value is decremented by 1 whenever a pixel fails the depth test.*

Although the depth-fail method effectively avoids the stencil counting inversion issue, it requires the additional back-capping of shadow volumes. This necessitates extra rasterisation time which can lead to considerable performance slowdowns under certain conditions.

In some cases it is therefore more advantageous to use the depth-pass method provided the point of view is not located within a shadow volume. It is also often possible to increase the performance of a stencil shadow volume implementation by utilising hardware extensions such as NVIDIA's depth bounds test, which provides for the culling of shadow volume sections that do not affect the visible area.

Interesting to mention though, work done by Kolic, Mihajlovic, and Budin [7] presents a shadowing technique purely developed to utilise current GPU (Graphics Processing Unit) advances. When Crow [3] implemented and defined the original shadow volume model back in 1977 he simply did not have access to any of these modern hardware acceleration aids and hence did not develop this now commonly used algorithm with these advances in mind.

Thakur, Cheng and Miura [9] also developed a discrete algorithm for improving the Heidmann [4] original. Their algorithm was primarily based on the elimination of various testing phases which resulted in some performance gains under certain conditions. For the purpose of the presented study, the depth-fail algorithm was empirically evaluated on various scenes, as described in section 4. However, prior to such evaluation, it was important to lay down appropriate evaluation criteria to be followed. These are discussed in the next section.

3. Evaluation Criteria

We present a set of criteria that was used to evaluate the depth-fail algorithm for shadow generation. The given evaluation criteria were selected with the aim of assessing the relationship between shadow rendering quality and performance - in turn allowing us to isolate key algorithmic weaknesses and possible bottleneck areas. Table 1 lists the proposed evaluation criteria in the first column, indicating in parenthesis whether its focus is on quality, performance or both. The second column provides the motivation for inclusion of the associated criterion.

Criteria	Motivation for Inclusion	Criteria	Motivation for Inclusion
Scalability (performance)	Evaluating the performance of an algorithm based on the intensifying complexity of the rendered scene enables the identification of algorithmic limits and the maximum threshold for scene and model complexity. (Analyse the overall performance impact due to an increase in the number of light sources and the shadow casting model's polygonal complexity).	Construction Complexity (performance)	Assessing the total number of clock cycles and the number of clock cycles for every routine of the shadow rendering algorithm allows us to identify not only the processor intensive operations, but also the areas where the algorithmic improvements can lead to the biggest overall performance gain.
Rendering Accuracy and Detail (quality)	Determining whether a shadow is cropped and/or skewed properly, and accurately projected onto other models and surfaces, allows for the evaluation of shadow rendering quality.	Instruction Set Utilisation (performance/ quality)	Comparing a standard C/Direct3D 9.0c implementation to the utilisation of Intel's SSE and AMD's 3DNow! instruction set allows for the evaluation of maximized parallelism (offered by these instruction sets) versus conventionally executed routines.

Table 1. *Evaluation criteria*

4. Results

To gather the necessary results, we implemented the depth-fail stencil shadow volume algorithm for a number of scenes. In each case, the scene was a relatively simple cubic environment featuring a single movable 3D model and a variable number of light sources. The 3D models utilised are those provided as samples by the Microsoft DirectX SDK. The test system had the following configuration: NVIDIA GeForce^{TM}6200TC 256MB (Video Card), AMD Athlon^{TM}3000+ (Processor), 1.5GB (Memory), 1280x1024 (Screen Resolution). The video card was selected due to it being entry level (variations in the rendering frame rates being highly visible) while at the same time supporting a vast array of current hardware extensions.

4.1. Scalability

In order to accurately benchmark the scalability of the shadow volume algorithm, we render a somewhat simple scene consisting of a relatively high polygon model (599 faces) and one light source. Figures 1(a) to (d) illustrate the shadows and constructed shadow volumes for the various models. Increasing the number of light sources will lead to additional shadows as shown in Figure 1(e).

Following this initial rendering, we systematically increase the number of light sources while varying the model complexity. Figure 2(a) summarises the resulting performance data. By analysing these results, it is clear that both the number of light sources and mesh complexity have an influence on the performance of the stencil shadow volume algorithm as illustrated in Figure 2(a). The primary

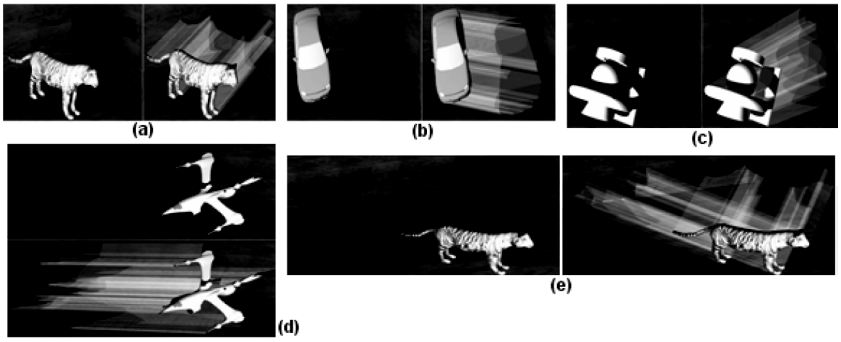

Figure 1. *(a) Rendering a shadow by means of a stencil shadow volume (1 light source) - cropped and skewed to fit surrounding area, (b) Shadow generated by the 'car' model (1 light source), (c) Shadow generated by the 'shapes' model (1 light source), (d) Shadow generated by the 'battleship' model (1 light source), (e) Shadow generated by the 'tiger' model (2 light sources)*

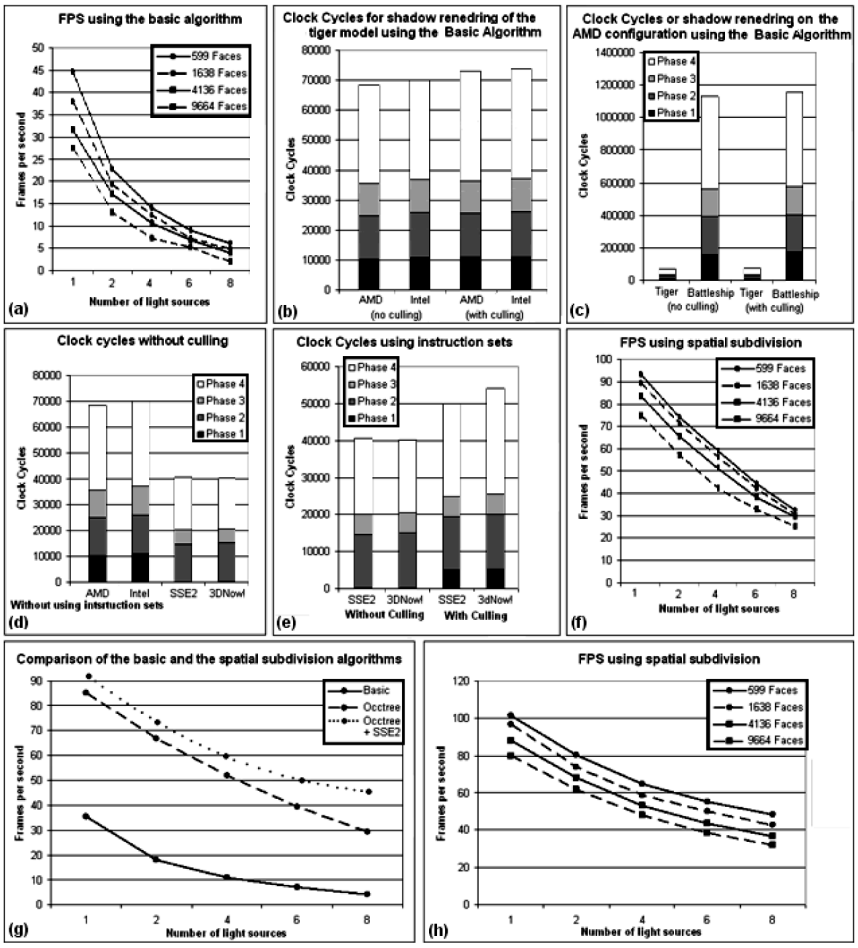

Figure 2. *(a) Number of light sources vs. FPS (basic algorithm), (b) Clock cycles with and without culling for shadow volume construction (599 faces 'tiger'), (c) Clock cycles for shadow volume construction of the tiger and battleship (9664 faces) with and without culling respectively, (d) Clock cycle gains when using instruction sets, (e) SSE2 vs. 3DNow! with and without the use of culling, (f) Number of light sources vs. FPS (our approach), (g) Our techniques vs the traditional approach, (h) Number of light sources vs. FPS (extended approach)*

bottleneck is in fact the shadow volume construction process. Increasing the number of light sources results in an additional shadow volume being created for each light source. Also, the greater the number of faces, the greater the number of silhouette edges considered during the shadow volume construction process.

4.2. Construction Complexity

Assessing the total number of clock cycles and the number of clock cycles for every routine of the shadow volume construction process allows us to isolate the processor intensive operations and possible areas of improvement. The results shown in Figure 2(b) compare the processor dependence of our 599-face 'tiger' mesh. These tests were executed on the same system as the scalability testing. We also employed an Intel Pentium 4 'Northwood' 2.8 GHz - based test system (with all its hardware components corresponding to that of our AMD test machine). The data obtained from this machine is also shown in Figure 2(b). These results will be used when comparing the standard C/Direct3D implementation to the utilisation of Intel's SSE2 and AMD's 3DNow! instruction sets.

We investigated the number of clock cycle iterations for each of the shadow volume construction phases. The first phase in this process is to calculate the number of light source facing triangles. The second phase involves construction of the silhouette polygons (also referred to as the silhouette plane polygons); with the third phase responsible for the construction of the shadow volume's capping triangles. The fourth phase is the actual construction of the shadow volume. We can also modify this phase to construct shadow volumes while at the same time culling polygons located outside of the volume. We can similarly modify the first shadow volume construction phase to calculate the number of light source facing triangles with the culling of polygons outside of the volume. As can be seen in Figure 2(b), the difference between the two hardware configurations is minimal. The culling of polygons located outside of the volume involves more work initially, thus resulting in some performance loss.

We repeated this same experiment on the AMD machine using models with more faces. As can be seen in Figure 2(c), the total number of clock cycles increases drastically when the number of faces increases, but the relation between the work done in the different phases does not differ significantly. The increasing clock cycle measurements are mostly due to the numerous conditional branches executed during the shadow volume creation process. Translating and/or rotating a mesh also impacts heavily on the clock cycle count due to the recalculation of the light source facing polygons and subsequently the shadow volume. Calculation of the number of light source facing polygons is the primary performance bottleneck and the only foreseeable performance enhancement would be to execute all the current sequential-conditional code in parallel - the section on `Instruction Set Utilisation` deals with this in detail.

4.3. Rendering Accuracy and Detail

The rendering accuracy and detail of projected shadows can be determined by way of visual observation. A model is translated and rotated within a scene and the detail of the projected shadow is investigated. Stencil shadow volumes, being per-pixel based, are by default of high quality. Figure 3(a) shows a properly cropped and skewed shadow.

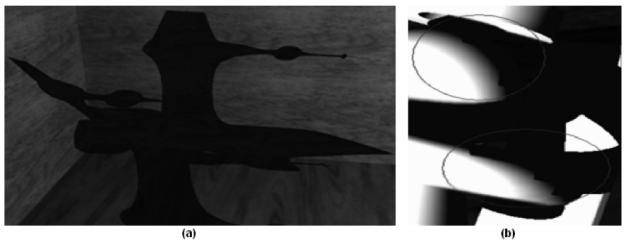

Figure 3. *(a) Properly cropped/skewed shadow, (b) Encircled areas: artefacts near silhouette edges*

Although the generated shadow is of high quality, the shadow volume algorithm is deficient in some contexts - specifically when a mesh casts a shadow onto itself, as shown in Figure 3(b). These well publicised artefacts are the result of the shadow volume's lighting computation phase. An object's faces are rendered as shadowed or lit depending on whether a face normal points towards the light source or not. Thus, an entire face will be rendered as shadowed although only part of it is actually in shadow (something the algorithm doesn't consider). Shadow mapping doesn't suffer from this and these artefacts can only be reduced by increasing the mesh complexity - thus resulting in less noticeable artefacts near the silhouette edges.

4.4. Instruction Set Utilisation

Comparing a standard C/Direct3D implementation to the utilisation of Intel's SSE2 and AMD's 3DNow! instruction sets allows for the evaluation of maximised parallelism (as offered by these

instruction sets) versus conventionally sequentially executed routines. The 3DNow! instruction set is a multimedia extension created by AMD for the improvement of vector processing and floating-point-intensive tasks as required by graphic-intensive and multimedia applications [1]. Intel's SSE is similar in nature and stands for Streaming Single Instruction, Multiple Data Extensions and is based on the principle of carrying out multiple computations using a single instruction in parallel [5]. The SEE instruction set (specifically SSE2 found on the Pentium 4 architecture) also adds 64-bit floating point and 8/16/32-bit integer support. As can be seen in Figure 2(d), there is significant gain especially in the first phase. It is also noticeable that the second phase (i.e. the construction of the silhouette polygons) did not lend itself to any gain. The difference between the utilised hardware offerings is negligible. Figure 2(e) illustrates the comparison between pure SSE2 vs. 3DNow! performance data and the results obtained by combining these instruction sets with culling.

To understand the concept of maximised parallelism, consider the following for-loop used to count the number of facing polygons:

```
for(int n = 0; n < numberOfTriangles; n++){
   if(triangleIsFacing(n)) {facing_triangles = facing_trangles + 1;}}
```

The given loop can be subdivided and parallelised via either SSE or 3DNow!, with each individual loop processing a portion of the triangle faces. This parallelised face counting results in a performance increase from 10841 clock cycles to 149 clock cycles for the SSE2 implementation and 10224 to 157 clock cycles for the 3DNow! implementation.

All the other shadow volume construction phases use similar loops with either the culling or the filling of triangle indices added. Partitioning and parallelising these loops via SSE2 or 3DNow! lead to substantial performance increases. The slight architectural differences between the Intel and AMD instruction sets lead to slightly varying results. The AMD architecture only slightly outperforms the Intel architecture when surfaces outside of the light volume aren't culled. One reason for this could be that the AMD processor's clock speed - the Athlon64 3000+ operates at a substantially lower rate than its Intel counterpart (2.8GHz). The AMD processor rating system (3000+ in this case) indicates that the processor is comparable to a 3.0GHz processor although operating at roughly 1.8GHz. This rating system is the result of AMD processors executing a greater number of instructions during each clock cycle than Intel's offerings [8]. It is, however; patently clear that usage of either SSE or 3DNow! results in considerable performance gains.

5. Performance Results Obtained by using Spatial Subdivision

We now present the results obtained from combining the depth-fail stencil shadow volume algorithm with spatial subdivision. This unification results in real-time frame rates for rather complex scenes. We primarily deal with static polygonal environments and, although an apt shadowing model and improvement over the traditional Heidmann [4] algorithm, our approach is mainly amenable to the rendering of static scenes - falling back on the original depth-pass/fail algorithm when lights sources are added, moved, or removed from a scene.

Our algorithm enhances the current depth-fail and depth-pass stencil shadow volume algorithms by enabling more efficient silhouette detection, thus reducing the number of unnecessary surplus shadow polygons. Our approach also includes a technique for the efficient capping of polygons, thus effectively handling situations where shadow volumes are being clipped by the point-of-view near clipping plane. Crucial to this implementation is the Octtree data structure. Relying on this datastructure, an Octtree algorithm sorts the collections of polygons that make up the shadow volumes into a specific visibility order. This order is pre-determined by the viewpoint.

Our approach basically uses the resulting Octtree to calculate the shadow volume unification by traversing the tree in a `front-to-back` order, thus in effect subdividing the surface (endpoint) polygons for each element/object. This arrangement results in performance gains where the rendered scene contains several concealed shadow volumes. However, the Octtree structure does require processing time to build and the non-Octtree enhanced shadow volume algorithm performs much better

where there are few concealed shadow volumes or in situations where light sources are dynamically added or removed. Figure 2(f) shows the frame rates obtained when using our technique with a varying number of light sources and models of differing complexity. Similar to the basic algorithm, both the number of faces and number of light sources have a negative impact on the overall frame rate.

In Figure 2(g) the frame rates achieved (for the 1638 faces model) via the implementation of special subdivision is compared to that obtained using the Heidmann algorithm. It is clear from the data that our approach results in significantly better performance than the original stencil shadow volume algorithm. This is especially visible where the number of light sources is less.

The next step was to extend this spatial subdivision approach by combining it with the utilisation of the SSE2 instruction set (the data obtained from the use of 3DNow! is nearly identical). Figure 2(h) shows the results obtained from this hybrid technique. The improvement is in the order of about 10%. In Figure 2(g), the frame rates attained from using special subdivision combined with the utilisation of the SSE2 instruction set is compared to that obtained from using the Heidmann algorithm.

6. Conclusion

The computer graphics industry has developed immensely during the past decade. Looking at the area of computer games one can easily see technological leaps being made on a yearly basis. However, most of the currently implemented stencil shadow volumes are based on the "vanilla" depth-fail/depth-pass algorithm. This conventional algorithm is based on a series of processor intensive conditionally executed branches. We replaced these conditionally executed branches with SSE and 3DNow! instructions, forcing their parallel execution during each rendering cycle and achieving a notable performance increase. Considering this improvement, we went even further by combining the depth-fail stencil shadow volume algorithm with spatial subdivision. The implementation of this method resulted in significant performance gains, especially where the rendered scene consisted of several concealed objects.

It is important to note that, despite all the algorithms available for the implementation of shadows, a lot of work remains in the field. Shadowing is immensely processor intensive and can only be successfully implemented on high-end hardware. Only by continuing research on the subject will real-time shadow generation become as common as texture mapping.

7. References

[1] ADVANCED MICRO DEVICES, INC., "AMD 3DNow! Technology Manual", *Published online: www.amd.com, 2000*,

[2] P BERGERON, "Shadow volumes for non-planar polygons", *Proc. Graphics Interface '85, 417-418*,

[3] FC CROW, "Shadow algorithms for computer graphics", *Proc. SIGGRAPH '77, 242-248, 1977*,

[4] T HEIDMANN, "Real shadows real time", *IRIS Universe, 28-31, 1991*,

[5] INTEL CORPORATION, "Getting Started with SSE/SSE2 for the Intel Pentium4", *Published online: www.intel.com, 2002*,

[6] MJ KILGARD AND C EVERITT, "Optimized Stencil Shadow Volumes", *GDC Presentation: http://developer.nvidia.com/docs/IO/8230/GDC2003_ShadowVolumes.pdf*,

[7] I KOLIC AND Z MIHAJLOVIC AND L BUDIN, "Stencil shadow volumes for complex and deformable objects", *Proc. 2004 11th IEEE International Conference, 314-317, 2004*,

[8] S RAU, "AMD PR Rating", *Published online: www.amd.com, 2002*,

[9] K THAKUR AND F CHENG AND KT MIURA, "Shadow generation using discretized shadow volume in angular coordinates", *Proc. Computer Graphics and Applications, 11th Pacific Conference, 224-233, 2003*