Shadow Mapping and Shadow Volumes: Recent Developments in Real-Time Shadow Rendering

University of British Columbia
CS514 Advanced Computer Graphics: Image Based Rendering
Instructor: Wolfgang Heidrich

Project Report

Andrew V. Nealen*
Technische Universität Darmstadt

Abstract

In recent years , both Williams' original Z-buffer shadow mapping algorithm [Williams 1978] and Crow's shadow volumes [Crow 1977] have seen many variations, additions and enhancements, greatly increasing the visual quality and efficiency of renderings using these techniques. Additionally, the fast evolution of commodity graphics hardware allows for a nearly complete mapping of these algorithms to such devices (depending on the GPU's capabilities) which results in real-time display rates as seen in the real-time version of Pixar's *Luxo Jr* and the use of hardware shadow maps therein. In this project report we describe the major contributions since Williams' and Crow's original publications in 1978 and 1977 respectively, briefly present both the shadow mapping (which is computed in image space) and the shadow volume algorithms, present more sophisticated approaches to shadow mapping which are better suited to high quality off-line renderers and describe the aliasing problems inherent in all shadow algorithms which operate in image space (and proposed solutions). Finally, we describe new extensions to the existing algorithms such as *perspective shadow maps* as described by [Stamminger and Drettakis 2002] in the 2002 *SIGGRAPH* conference, and *robust stenciled shadow volumes* by Mark Kilgard [Everitt and Kilgard 2002].

Keywords: shadows, shadow mapping, algorithms, visual realism, graphics hardware

Figure 1: Shadows create drama: a screen-shot from id software's newest creation *Doom 3*. A good example using *stenciled shadow volumes*.

1 Overview

The importance of Lance Williams' original shadow algorithm [Williams 1978] is obvious, as many renderers such as Pixar's RenderMan utilize the technique and the original paper was also elected to be of *seminal* nature, see [Wolfe 1998]. The basic idea is seemingly simple, yet its largest benefit is also its greatest drawback: the discrete nature of the image space computation and thus the introduction of aliasing artifacts (For a more detailed description, see section 4). Since Williams' original publication many enhancements and variations of the algorithm have been proposed and published, yet only recently (in the past few years) have more major contributions been made to the technique, which is not surprising as it has been in these years that commodity graphics hardware evolution has surpassed moore's law. This new computational (graphics)power has made many multi-pass algorithms more attractive to real-time rendering, including Williams' shadow mapping algorithm, Crow's shadow volume algorithm and derivatives thereof. Poulin, Fournier and Woo [Woo et al. 1990] have summed up the efforts up to 1990, including many off-line techniques not described in this report. Due to the age of their survey the paper fails to present the newer developments in the field such as (for shadow mapping): support for hardware acceleration [Kilgard 2001][Brabec and Seidel 2001], projective texture mapping [Segal et al. 1992][Heidrich 1999], newer soft shadow mapping techniques [Brabec and Seidel 2002][Heidrich et al. 2000] and perspective shadow maps [Stamminger and Drettakis 2002]. For the shadow volume algorithm: stenciled shadow volumes [Everitt and Kilgard 2002][Heidmann 1991], soft stenciled shadow volumes [Akenine-Moeller and Assarsson 2002] and hybrid methods between shadow mapping and shadow volumes [McCool 2000].

In this project report we attempt to bridge this gap and present to the reader a comprehensive overview of available recently developed techniques, enhancements and additions to the basic shadow mapping and shadow volume algorithms. To keep within the scope of this report we only briefly describe off-line, highly sophisticated versions of shadow mapping, but list relevant literature for further reading. Also, the description of the shadow mapping and shadow volume techniques presented in this report may also not be as detailed as necessary for direct implementation, therefore all necessary literature is referenced. An invaluable resource for developers is *NVIDIA*'s developer web-site at `http://developer.nvidia.com`. Given this information, the reader should be up to date with the current state of the art in real-

*e-mail: andy@nealen.com

time shadow rendering.

2 Alternatives to (Real-Time) Shadow Mapping and Shadow Volumes

2.1 Projected Planar Shadows

During the research phase, which inspired this report, other alternatives to shadow mapping and shadow volumes were surveyed. Of the possible alternatives, *projected planar shadows* seems to be the most promising algorithm. Blinn [Blinn 1988] invented the original technique, which only allows shadows to be cast on planar surfaces. Haines [Haines 2001] extended the planar shadow projection technique to soft shadows, although with the same limitations of Blinn's basic projection algorithm. Kilgard [Kilgard 1999] shows how to utilize the stencil buffer to improve the visual quality of projected shadows.

The idea: given a plane $\mathbf{P} : \mathbf{n} \cdot \mathbf{x} + d = 0$ and a point light source $\mathbf{l}$, construct a projection matrix $\mathbf{M}$ that projects each vertex $\mathbf{v}$ onto the plane $\mathbf{P}$. The point $\mathbf{p}$, which is the projection of vertex $\mathbf{v}$ onto the plane $\mathbf{P}$ can be described as

$$\mathbf{p} = \mathbf{l} - \frac{d + \mathbf{n} \cdot \mathbf{l}}{\mathbf{n} \cdot (\mathbf{v} - \mathbf{l})} \mathbf{v} - \mathbf{l}$$

This can be converted into a projection matrix which satisfies $\mathbf{Mv} = \mathbf{p}$ [Haines and Moeller 1999]

$$\mathbf{M} = \begin{pmatrix} \mathbf{n} \cdot \mathbf{l} + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & \mathbf{n} \cdot \mathbf{l} + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & \mathbf{n} \cdot \mathbf{l} + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & \mathbf{n} \cdot \mathbf{l} \end{pmatrix}$$

Care must be taken to avoid anti-shadows, e.g. when the light source is between the occluder and the shadow plane. Other subtleties and solutions (such as utilizing the stencil buffer to prevent geometry overlapping) can be found in [Haines and Moeller 1999] and [Kilgard 1999].

2.2 Off-line Shadow Mapping

Stamminger and Drettakis [Stamminger and Drettakis 2002] describe a slew of very sophisticated shadow mapping algorithms in their paper on perspective shadow maps. They note that almost all of these algorithms involve multiple rendering passes (as they generally use more than one shadow map) and more involved data structures and thus do not, or do not easily map to hardware. Nevertheless, for completeness, we list the relevant literature in the references section, specifically for soft shadows [Agrawala et al. 2000], adaptive shadow maps [Fernando et al. 2001], deep shadow maps [Lokovic and Veach 2000] and solar shadows [Tadamura et al. 2001].

3 The Shadow Volume Algorithm

The shadow volume algorithm is a *geometry based* shadow algorithm which requires connectivity information of the polygonal meshes in the scene to efficiently compute the silhouette of each shadow casting object (each *occluder*). It is also a *per pixel* algorithm, which performs an *in shadow* test for each rendered fragment. This operation can be accelerated using graphics hardware (the *stencil buffer*) as we will describe later. In pseudocode, the algorithm reads as follows

```
procedure SHADOWVOLUMERENDERING
for all rasterized fragments do
  draw fragment with ambient and emissive lighting
  update the Z-buffer
end for
COMPUTEFRAGMENTSINSHADOW
for all rasterized fragments do
  if not INSHADOW(fragment) then
    draw fragment with diffuse and specular lighting
  end if
end for
```

As obvious, the key issue here is determining whether a rendered fragment is in shadow or not. This procedure maps easily to any graphics hardware with a *stencil buffer*. After rendering each fragment with emissive and ambient lighting, the following must be inserted

```
procedure COMPUTEFRAGMENTSINSHADOW (Z-pass)
for all shadow casting objects do
  compute potential silhouette edges (PSE) of the polygonal
  model
  compute the shadow volume polygons (shadow quads) from
  the light source(s) and the PSE
end for
for all front facing shadow quads from viewpoint do
  if Z-buffer test passes then
    increment stencil buffer value
  end if
end for
for all back facing shadow quads from viewpoint do
  if Z-buffer test passes then
    decrement stencil buffer value
  end if
end for
```

Now to evaluate the boolean function **INSHADOW** (used in **SHADOWVOLUMERENDERING**) for each fragment we simply query the stencil buffer value. If the the value stored in the stencil buffer after **COMPUTEFRAGMENTSINSHADOW** is greater than zero, the fragment is in shadow and must not be drawn in the second rendering pass.

To understand the procedure geometrically, see figure 2 taken from [Akenine-Moeller and Assarsson 2002].

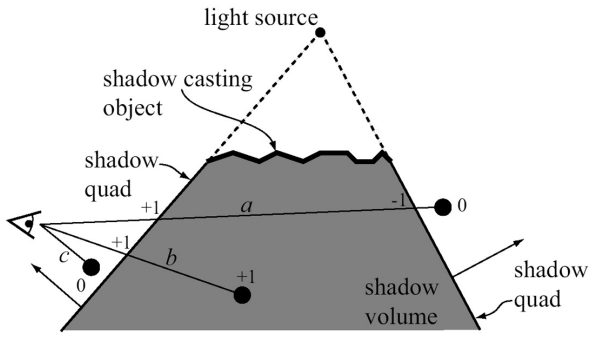

Figure 2: Shadow Volume Rendering: counting entering and exiting rays from the viewpoint to determine whether a visible fragment is in shadow.

When computing the shadow query using the stencil buffer as described above, the algorithm is usually titled *stenciled shadow*

volumes and was first proposed in [Heidmann 1991] and further refined in [Kilgard 1999]. The algorithm as described suffers from an amount of drawbacks which make it impractical for certain situations, if special cases are not treated correctly. First off, the algorithm only works if the viewpoint is outside of shadow, otherwise the stencil counting is inverted. This can be remedied by testing this special case, inverting the shadow test, and also initializing the stencil buffer to 2^{N-1} (where N is the stencil buffer precision), as the stencil buffer holds only unsigned values and decrementing from a zero value would again result in incorrect shadows.

A more elegant solution is proposed in [Everitt and Kilgard 2002] , which was partially inspired by an insight of John Carmack, lead programmer of id software [Carmack 2000]. First off, instead of computing the stencil values by incrementing front facing shadow quads and decrementing back facing shadow quads on Z-buffer pass, the entire process is modified to count from infinity instead of from the viewpoint, the so called *Z-fail* version

procedure **COMPUTEFRAGMENTSINSHADOW**(*Z-fail*)
for all shadow casting objects **do**
  compute potential silhouette edges (PSE) of the polygonal model
  compute the shadow volume polygons (shadow quads) from the light source(s) and the PSE
end for
for all front facing shadow quads from viewpoint **do**
  if Z-buffer test fails **then**
    decrement stencil buffer value
  end if
end for
for all back facing shadow quads from viewpoint **do**
  if Z-buffer test fails **then**
    increment stencil buffer value
  end if
end for

The two representations (*Z-pass* and *Z-fail*) are completely equivalent and compute the same value, yet they do not suffer from the problems mentioned above: the viewer being in shadow is no longer a special case.

Another problem related to stenciled shadow volumes has also been solved in [Everitt and Kilgard 2002], namely the necessity to compute shadow volume caps to prevent clipping errors, as the per pixel computation is carried out in already clipped, post perspective-transform space (clip space). By using the Z-fail method and setting the far clipping plane to infinity, we only need to cap off the shadow volume by adding the occluder polygons facing the light source to the shadow volume, and possibly projecting these onto the near clip plane, should they be *in front* of this plane. By setting the far plane to infinity we can guarantee that the shadow volume will never be clipped away by the far clipping plane. This entire technique is named *Robust Stenciled Shadow Volumes* and fully described in [Everitt and Kilgard 2002]. Some good code and comprehensible presentations can be obtained from `http://developer.nvidia.com`.

Up to this point, the described stenciled shadow volume algorithm produces only hard shadows and, thus, is only suited for ideal point light sources without the *penumbrae* typical for area light sources. One possible variation is to recompute the shadow volumes from a variety of point light samples nearby the original point light source and accumulate the resulting shadows. This is both computationally expensive and also results in visually displeasing results, see [Kilgard 1999]. Möller [Akenine-Moeller and Assarsson 2002] has recently presented a viable solution, computing soft shadows for existing shadow volumes using *penumbra wedges*, see figure 3 (taken from [Akenine-Moeller and Assarsson 2002]).

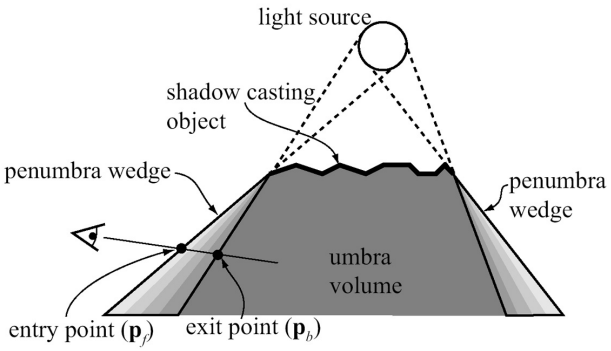

Figure 3: Shadow Volume Rendering using Penumbra Wedges

The procedure is quite intricate and is therefore out of the scope of this report. See the paper for a detailed description.

Overall, the stenciled shadow volume produces visually impressive results at high frame rates, which could be seen at this years Electronic Entertainment Expo in form of the newest *Doom 3* demo (see figure 1).

The drawbacks inherent to the method are due both to the geometric nature and the per pixel operation of the algorithm. Because a potential silhouette must be computed for every shadow casting object in the scene, the scene complexity has direct influence on the performance of the algorithm (a property that shadow maps do not have, as we will see later). Also, rendering the shadow quads to the stencil buffer can consume tremendous amounts of stencil fill rate. Kilgard therefore proposes the use of more effective shadow volume culling schemes [Everitt and Kilgard 2002].

4 The Shadow Mapping Algorithm

Shadow mapping is a completely image-space algorithm, which means that no knowledge of the scene's geometry is required to carry out the necessary computations. As it uses discrete sampling it must deal with various aliasing artifacts, the major drawback of the technique. These drawbacks can be partially overcome as we will describe soon.

Figure 4: Pixar's real-time version of Luxo Jr. uses hardware accelerated shadow mapping

This algorithm, like the shadow volume algorithm, performs a per-pixel (per-fragment) *in shadow* test to determine whether a pixel has a diffuse and/or specular component or not. The two pass algorithm in pseudocode

procedure **SHADOWMAPPING**

Render depth buffer (Z-buffer) from lights point of view, resulting in a *shadow map* or *depth map*
Now, render scene from the eye's point of view
for all rasterized fragments **do**
  Determine fragment's xyz position relative to the light
  That is transform each fragment's xyz into the light's coordinate system
  $A \leftarrow depth\_map(x,y)$
  $B \leftarrow$ z-value of fragment's xyz light postition
  if $A < B$ **then**
    fragment is shadowed
  else
    fragment is lit
  end if
end for

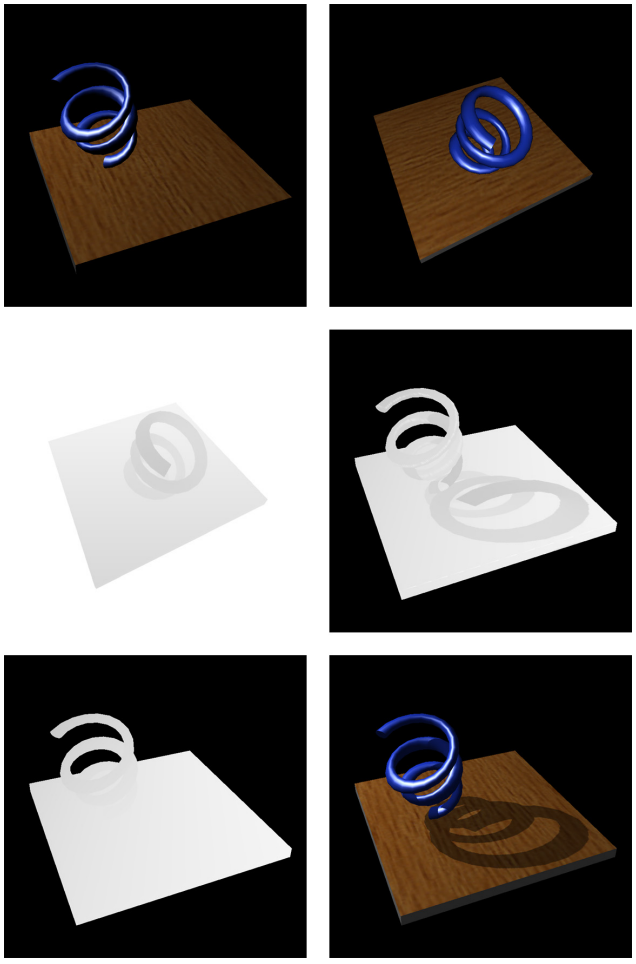

Figure 5: Shadow mapping procedure visualized. Top left (a): the unshadowed scene from the eye's view. Top right (b): The scene as seen from the (point) light source. Mid left (c): the shadow map (depth map) constructed from the light source view. Mid right (d): the shadow map projected onto the eye's view (the A value in SHADOWMAPPING). Bottom left (e): the lights planar distance projected onto the eye's view (the B value in SHADOWMAPPING). Bottom right (f): The shadowed scene after performing the depth test between (d) and (e).

A comprehensive visualization of the algorithm applied to a demo scene (taken from the *NVIDIA* demo) and the accompanying description is shown in figure 5.

The first problem this algorithm must deal with is the case of erroneous self shadowing: When transforming a point from a surface in the eye's point of view into the lights coordinate system, A and B in the above algorithm should ideally be equal. Yet due to Z-buffer quantization, it is very likely that $a \neq b$ and the transformed point will fall above or below the surface [Williams 1978]. To cure this imperfection, a bias value is subtracted to ensure that false self shadowing is removed from the scene. Williams simply subtracts a constant bias from the Z-values points after they have been transformed into light space, which may move the shadow line slightly. Kilgard [Kilgard 2001] uses OpenGL's `glPolygonOffset` to offset the depth value in the shadow map back and compensate for the slope of the polygon (greater slope = more offset). Stamminger and Drettakis [Stamminger and Drettakis 2002] again use a constant bias subtracted from the *perspective shadow map* due to the fact that their depth map is constructed after perspective transform and contains non-uniformly scaled objects. In general, the amount of bias necessary decreases with higher shadow map precision and the tendency should be towards a higher bias setting (figure 6).

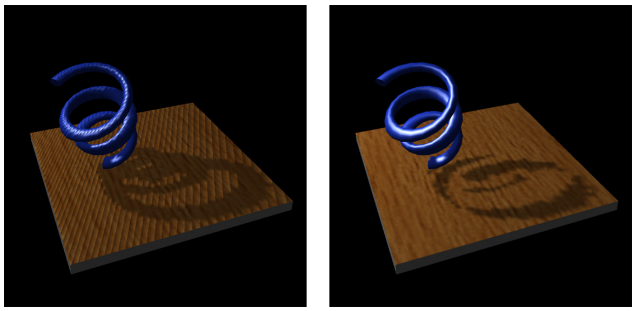

Figure 6: The effect of bias offset. the left image shows too little bias; everything begins to shadow itself, displaying a vivid moiré. The right image uses too much bias, the shadows start too far back.

To perform the depth test, the depth buffer must be read back to memory and accessed with the transformed coordinates of the rasterized fragment. The fragment's light position can be generated using eye-linear texture coordinate generation, which relies on *projective texturing*, see [Heidrich 1999], [Segal et al. 1992] and [Heckbert 1986]. The entire procedure of copying the Z-buffer to a texture, transforming the fragment to the lights xyz coordinates, accessing the depth texture and performing the shadow test can be mapped to hardware using existing OpenGL extensions (and the appropriate hardware supporting these extensions of course). A detailed description is beyond the scope of this report. An excellent treatment on the details can be found in [Kilgard 2001].

As described above, *aliasing* artifacts can reduce shadow quality significantly. These artifacts (figure 8, right image and figure 10, top right image) are due to *shadow map undersampling*, which means that a shadow map texel maps to more then one frame buffer pixel. An elegant formalization reads as follows [Stamminger and Drettakis 2002] (see figure 7): for every pixel of size $d_s \times d_s$ in the shadow map maps to a pixel area of (approximately) size d in the final image

$$d = d_s \frac{r_s}{r_i} \frac{\cos\beta}{\cos\alpha}$$

Undersampling appears when d is larger than the image pixel size d_i. Shadow map aliasing can be split into two independent parts, *perspective aliasing*, dependent on the term $d_s r_s / r_i$ and *projection aliasing*, dependent on $\cos\beta / \cos\alpha$.

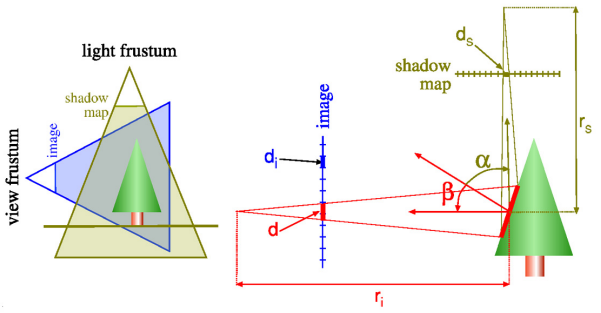

Figure 7: Formalization of shadow map aliasing

One solution to the aliasing problem, *percentage closer filtering*, was proposed by Reeves et. al. [Reeves et al. 1987]. In general, depth map values can not be blended, as this can lead to pixels being wrongly in shadow [Kilgard 2001]. Percentage closer filtering averages over boolean comparison results within the extents of the filter kernel, so if for example we operate with a 3*x*3 kernel around the computed fragment, as shown in figure 9, with the displayed results then the pixel is said to be 55% in shadow (for results see figure 8, left image).

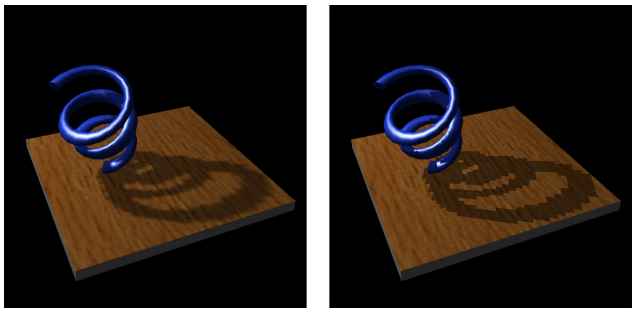

Figure 8: Effects of percentage closer filtering (shadow map resolution 128×128 to heighten artifacts). Left image: with filtering. Right image: no filtering

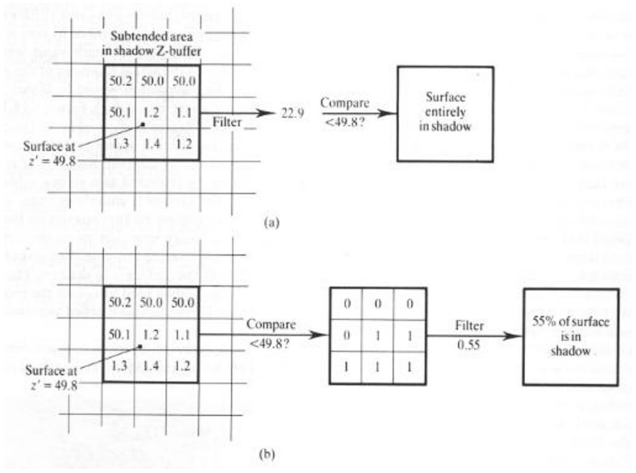

Figure 9: Percentage closer filtering principle

Recently (at *SIGGRAPH* 2002), Stamminger and Drettakis presented another solution to greatly reduce perspective aliasing: *perspective shadow maps* [Stamminger and Drettakis 2002]. The basic idea is to perform the shadow map computation and the shadow test in normalized device coordinate space after perspective transformation (clip space). The goal here is to keep the fraction r_s/r_i close to a constant. In post perspective space the final image is an orthogonal view onto the unit cube, therefore perspective aliasing due to distance to the eye is avoided. For a more in depth analysis and special cases see [Stamminger and Drettakis 2002]. The basic principle and results of the two methods are shown in figure (taken from [Stamminger and Drettakis 2002]). It should be noted though, that the example depicted in figure 10 is a case for which the perspective shadow mapping technique produces excellent results. Cases in which the perspective shadow map converges to the standard uniform shadow map can easily be constructed.

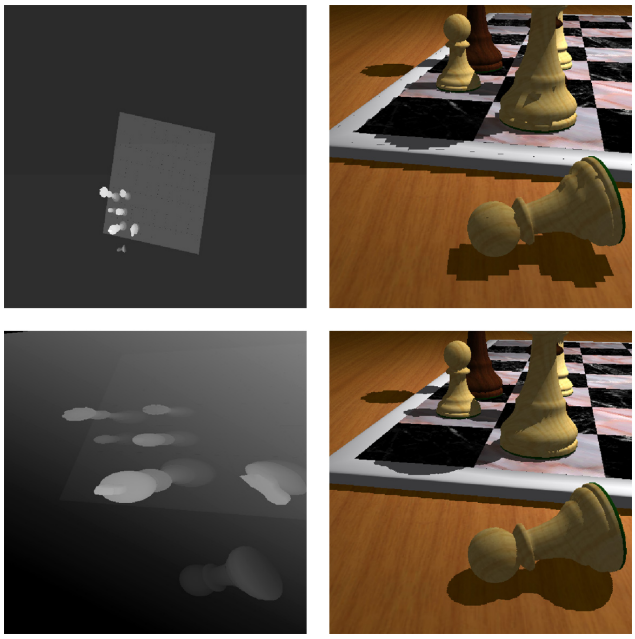

Figure 10: Perspective shadow maps. Top row: using a standard uniform shadow map generated in (world)light space. Bottom row: using a perspective shadow map generated in post-perspective transform space (clip space).

Also, just like for the shadow volume algorithm, some soft shadow extensions to the shadow map algorithm have been suggested, many of which also suited to real-time rendering. These algorithms are not necessarily physically correct, but produce aesthetically pleasing results as pointed out by Brabec and Seidel in their paper [Brabec and Seidel 2002], where single sample soft shadows are generated. Another extension to linear light sources is described in [Heidrich et al. 2000].

5 A Hybrid Approach

One more shadow algorithm which deserves mention in this report is McCool's clever idea *shadow volume reconstruction from depth maps* [McCool 2000]. This algorithm is a hybrid of the shadow map and shadow volume algorithms and does not require a polygonal representation of the scene. Instead of finding the silhouette edges via a dot product per model edge (shadow volumes), a depth map of the scene from the light's point of view is acquired (shadow map) from which the silhouette edges are extracted using computer

vision techniques. From these edges the shadow volumes are constructed. McCool also describes a method which uses only a one-bit stencil buffer and toggling the stencil value during shadow volume rendering. For a detailed description, see [McCool 2000] and figure 11.

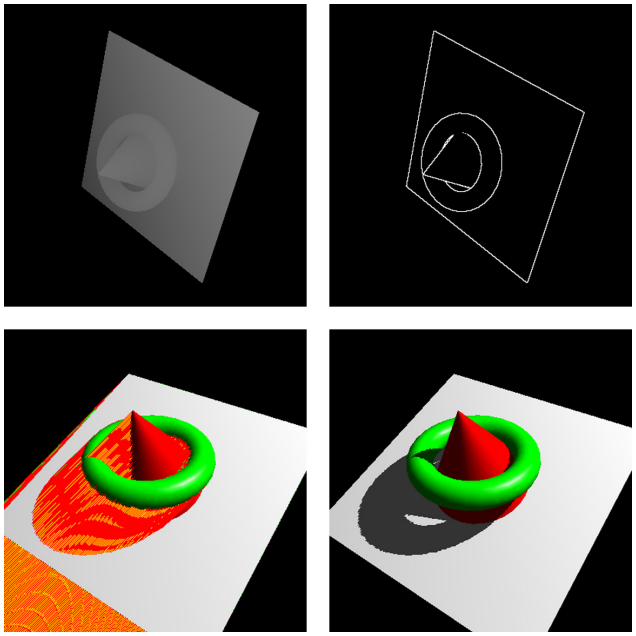

Figure 11: Shadow volume reconstruction from depth maps. Top left (a): the depth map as seen from the light source. Top right (b): the edges detected from (a) using computer vision algorithms. Bottom left (c): the extruded shadow volume quads. Bottom right (d): The final result after rendering using a 1-bit stencil buffer.

6 Discussion

In a recent quote (from August 3rd 2002), John Carmack from id software replied to an email in which he was asked why stenciled shadow volumes were chosen over shadow maps for doom3's rendering architecture, even though stenciled shadow volumes force the application to be *lower*-poly and permits the use of certain special effects such as displacement mapping. Here his reply

> "Shadow buffers make good looking demos with controlled circumstances, but when you start using them for a "real" application, you find that you need absolutely massive resolution to get acceptable results for omnidirectional lights, and a lot of the artifacts need to be tweaked on a per-light basis.
>
> While it is possible to do shadow buffers on GF1/radeon class hardware,without percentage closer filtering they look wretched.
>
> If we were targeting only the newest hardware, shadow buffers would have a better shot, but even then, they have more drawbacks than are commonly appreciated."

Carmack is known to do extensive research before making such decisions, and id software's engines are generally accepted as state-of-the-art in video game technology, so the opinion is a completely valid view on the tradeoffs between shadow maps and shadow volumes. It is not clear if McCool's hybrid approach was ever considered for the doom3 engine.

One of the drawbacks of shadow maps Carmack mentions, is that to achieve good visual quality (referring to the RenderMan architecture) Pixar uses 2k or 4k focussed on a very narrow field of view, assuming that projections outside the shadow map are not in shadow. The problem of the narrow field of view is treated in [Brabec et al. 2002] where Brabec uses Heidrich's omnidirectional dual parabolic parametrization for the shadow map. Depending on the resolution, this can naturally lead to a unwanted growth in aliasing artifacts.

In any case, both algorithms (and the hybrid) come with their fare share of trade-offs and a decision must be made depending on the nature of the application to be rendered. A general statement in favor of a certain approach can not be made at this point, yet it is quite obvious (in the opinion of the author of this report) that an educated decision can be made which leads to very convincing results (figure 12).

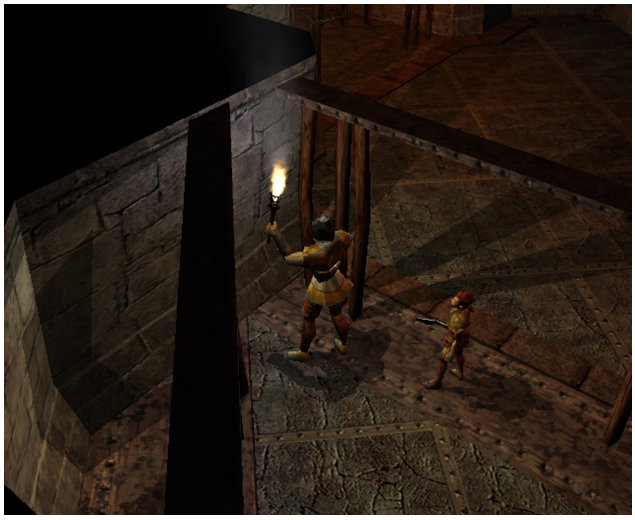

Figure 12: Screenshot from Bioware's Neverwinter Nights (shadow volumes)

7 Conclusions

The fast development of computer graphics hardware in the past few years has helped map the described algorithms almost completely to the hardware accelerators. Shadow volumes would be much faster could the silhouette computation be carried out in hardware as well. Newer GPU's and more flexible vertex and fragment programs as available by DirectX 9 and the newest OpenGL specifications will make such computations possible. Shadow volume extrusion can already be mapped to hardware, a demo of this technique using a vertex shader is available on the *NVIDIA* developer website. It remains to be seen whether the existing techniques will be extended to more realistic shadow display, specifically real-time soft shadows (e.g. [Akenine-Moeller and Assarsson 2002]), or if hardware development will inspire completely original ideas.

8 Acknowledgements and Thanks

I am greatly indebted to Mark Kilgard and Cass Everitt for making available the vast amount of information on shadow algorithms via the *NVIDIA* developer website. Furthermore I would like to thank my supervisor and course instructor Wolfgang Heidrich for his support and direction during my stay at the University of

British Columbia, Vancouver during the academic period Winter 2001/2002 (Sept. 2001 - April 2002). I had many fruitful discussions on various computer science and graphics related topics with Alexander Remann, Hendrik Kueck, Dave Burke, Eddy Boxerman, Eric Brochu, David Pritchard, Frank Hutter, Stefan Pochmann and many more and would also like to thank them all for their ongoing help and inspiration.

References

AGRAWALA, M., RAMAMOORTHI, R., HEIRICH, A., AND MOLL, L. 2000. Efficient image-based methods for rendering soft shadows. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Computer Graphics Proceedings, Annual Conference Series, 375–384. ISBN 1-58113-208-5.

AKENINE-MOELLER, T., AND ASSARSSON, U. 2002. Approximate soft shadows on arbitrary surfaces using penumbra wedges. In *13th Eurographics Workshop on Rendering 2002*.

BLINN, J. F. 1988. Jim blinn's corner: Me and my (fake) shadow. *IEEE Computer Graphics & Applications 8*, 1 (January), 82–86.

BRABEC, S., AND SEIDEL, H.-P. 2001. Hardware-accelerated rendering of antialiased shadows with shadow maps. In *Computer Graphics International 2001*, 209–214. ISBN 0-7695-1007-8.

BRABEC, S., AND SEIDEL, H.-P. 2002. Single Sample Soft Shadows Using Depth Maps. In *Proc. Graphics Interface*, 219–228.

BRABEC, S., ANNEN, T., AND SEIDEL, H.-P. 2002. Shadow mapping for hemispherical and omnidirectional light sources. In *Computer Graphics International (CGI)*.

CARMACK, J. 2000. Carmack on shadow volumes. Personal correspondence between Mark Kilgard and John Carmack.

CROW, F. C. 1977. Shadow algorithms for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 77)*, vol. 11, 242–248.

EVERITT, C., AND KILGARD, M. J. 2002. Practical and robust stenciled shadow volumes for hardware-accelerated rendering. Published online at developer.nvidia.com.

FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. 2001. Adaptive shadow maps. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 387–390. ISBN 1-58113-292-1.

HAINES, E., AND MOELLER, T. 1999. *Real Time Rendering*.

HAINES, E. 2001. Soft planar shadows using plateaus. *Journal of Graphics Tools 6*, 1, 19–27.

HECKBERT, P. 1986. Survey of texture mapping. In *IEEE Computer Graphics and Applications*, 56–67. reprinted in 'Tutorial: Computer Graphics: Image Synthesis', Kenneth Joy, et al., eds., IEEE Computer Society Press, Washington, DC, 1988; earlier version appeared in Proceedings of Graphics Interface '86, Vancouver BC, May 1986.

HEIDMANN, T. 1991. Real shadows real time. *IRIS Universe*, 18, 28–31.

HEIDRICH, W., BRABEC, S., AND SEIDEL, H.-P. 2000. Soft shadow maps for linear lights. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, Eurographics, 269–280. ISBN 3-211-83535-0.

HEIDRICH, W. 1999. *High-quality Shading and Lighting for Hardware-accelerated Rendering*. PhD thesis.

KILGARD, M. J. 1999. Improving shadows and reflections via the stencil buffer. Published online at developer.nvidia.com.

KILGARD, M. J. 2001. Shadow mapping with today's opengl hardware. published online at developer.nvidia.com.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Computer Graphics Proceedings, Annual Conference Series, 385–392. ISBN 1-58113-208-5.

MCCOOL, M. D. 2000. Shadow volume reconstruction from depth maps. *ACM Transactions on Graphics 19*, 1 (January), 1–26. ISSN 0730-0301.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, vol. 21, 283–291.

SEGAL, M., KOROBKIN, C., VAN WIDENFELT, R., FORAN, J., AND HAEBERLI, P. E. 1992. Fast shadows and lighting effects using texture mapping. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, vol. 26, 249–252. ISBN 0-201-51585-7.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *Proceedings of ACM SIGGRAPH 2002*, ACM Press/ ACM SIGGRAPH, J. Hughes, Ed., Annual Conference Series.

TADAMURA, K., QIN, X., JIAO, G., AND NAKAMAE, E. 2001. Rendering optimal solar shadows with plural sunlight depth buffers. *The Visual Computer 17*, 2, 76–90. ISSN 0178-2789.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, vol. 12, 270–274.

WOLFE, R., Ed. 1998. *Seminal Graphics: Pioneering Efforts That Shaped The Field*.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A survey of shadow algorithms. *IEEE Computer Graphics & Applications 10*, 6 (November), 13–32.