Downloaded from orbit.dtu.dk on: Apr 19, 2024

DTU Library

Boundary Correct Real-Time Soft Shadows

Jacobsen, Bjarke; Christensen, Niels Jørgen; Larsen, Bent Dalgaard; Pedersen, Kim S.

Published in:
Computer Graphics International

Link to article, DOI:
10.1109/CGI.2004.1309215

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Jacobsen, B., Christensen, N. J., Larsen, B. D., & Pedersen, K. S. (2004). Boundary Correct Real-Time Soft Shadows. In *Computer Graphics International* IEEE Computer Society Press.
https://doi.org/10.1109/CGI.2004.1309215

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Boundary Correct Real-Time Soft Shadows

Bjarke Jakobsen Niels J. Christensen Bent D. Larsen Kim S. Petersen
Informatics and Mathematical Modelling
Technical University of Denmark
{bj, njc, bdl}@imm.dtu.dk, kim@deadline.dk

Abstract

This paper describes a method to determine correct shadow boundaries from an area light source using umbra and penumbra volumes. The light source is approximated by a circular disk as this gives a fast way to extrude the volumes. The method also gives a crude estimate of the visibility of the area light source as seen from a point in the shadow region. Instead of rendering the volumes to the stencil buffer, we use an extended shadow map - a so-called D-buffer, which among other things stores distances from the center of the light source to the umbra and penumbra volumes. The method is suited for implementation on most programmable hardware. Though some crude approximations are used in the visibility function, the method can be used to produce soft shadows with correct boundaries in real time.

1. Introduction

Soft shadows provide information which is important to give the viewer of a scene an impression of spatial relationships between objects. They are able do so because the softness of a shadow provides information of the relative distances between the light source, the occluding objects and the objects receiving the shadow. When hard shadows are used, this information is not available, which can lead to misinterpretations of the scene. Non-real time global illumination methods, such as radiosity and distributed ray tracing, can handle various types of area light sources and thus render soft shadows by approximating the mathematics of light transportation. Real-time rendering of shadows is dominated by two hard shadow methods - shadow mapping ([23]) and shadow volumes ([9], [15]). The current methods for generating soft shadows in real time can be divided into:

- Image based methods, which use one or more shadow maps.
- Object based methods, where the actual geometry in the scene is used.

1.1. Image Based Methods

In a number of methods the area light source is sampled by a number of point light sources, and regular shadow maps are rendered from each sample. These samples are then combined into either a *Radiance Texture*([14]) or a *Layered Depth Map*([1]). To reduce the number of samples needed, other methods render shadow maps for only the vertices of a linear ([16]) or polygonal ([25]) light source and interpolate these for each receiving pixel. In [22], a convolution filter is used on a traditional shadow map to produce a soft shadow texture. The convolution filter is dependent on the light source and the distances between the light source, the occluder and the receiver. [18] uses both a regular shadow map and a *shadow-width map* and performs shadow calculation by applying a 2D function from a look-up of each map.

1.2. Object Based Methods

Several methods ([12], [6], [24], [8]) add information of object geometry to a shadow map in order to generate object driven penumbrae. [19] uses *Penumbra wedges* as an extension of the shadow volume method, and visibility is approximated using a 16-bit stencil-buffer. The wedges are generated by extruding silhouette edges, which are determined from the center of the light source. In more recent work ([2], [3]), the authors use programmable graphics hardware to project the silhouette edges onto the light source to calculate the visibility.

1.3. Problem statement

In survey [13] a number of general issues concerning soft shadows are presented. It is claimed that physically exact soft shadows are impossible to include in a real time rendering method for general scenes. Where most global illumination methods approximate the visibility using hundreds of samples of the light source, the presented real time methods all try to overcome this sampling through various simplifications. However, in none of the current methods, true soft shadows can be gener-

Proceedings of the Computer Graphics International (CGI'04)
1530-1052/04 $20.00 © 2004 **IEEE**

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.

(a) Point determined silhouette (b) Area determined silhouette

Fig. 1. The shadow regions are not cast correctly if silhouettes are found using a single point c when generating penumbra wedges instead of using the umbra and penumbra volumes from an area light source.

ated in real time for any given scene.

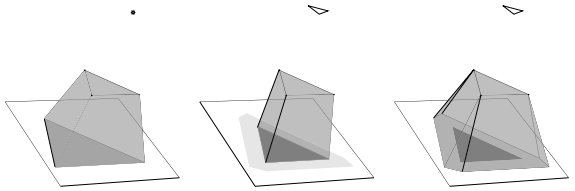
Among the presented methods, the one from [3] is superior to the others, as it produces very convincing soft shadows. The two main approximations used are that the silhouettes are found using a single point of the light source, and that overlapping occluders are not handled correctly in the penumbra region. An example of the consequences of the first approximation is shown in Figure 1.

The case in Figure 1 is quite extreme, since the light source is very large compared to the geometry of the scene. It is noted in [3], that few subdivisions of a large light source will make the artifact sketched in Figure 1 become practically indistinguishable. This is, however, not a general solution, and the problem of generating soft shadows in real time thus still needs work.

The situation in Figure 1(b) could be obtained if *umbra volumes* and *penumbra volumes* were used as an extension to the shadow volume method. These volumes are defined in [20] as respectively the intersection of and the minimum convex hull enclosing all shadow volumes generated from every point on the light source. The different types of volumes are illustrated in Figure 2.

The first task in generating these volumes is to determine the silhouettes of the occluding objects. This step is not shown for the simple occluding triangle in Figure 2, since all edges are silhouette edges. When using more complex occluders, the silhouette determination must be performed prior to the volume extrusion. The determination procedure depends on the shape of the particular light source.

The silhouette extraction for shadow volumes is described in [11]. For closed polygonal, two-manifold meshes, the method marks each polygon as being either front or back facing with regards to the light source. An edge belongs to the silhouette if it connects a front and a back facing polygon. The shadow volume is generated by extruding each silhouette edge toward infinity in the direction defined by the light source and the edge itself. The umbra and penumbra volumes are described elaborately in [7]. Here, as in Figure 2, the volumes are generated separately for each occluding triangle, since



(a) Shadow volume (b) Umbra volume (c) Penumbra volume

Fig. 2. Different types of volumes for shadows

no silhouette is calculated.

In the following, we describe a method that overcomes the problem of using the wrong silhouette. This is done because the shape of the soft shadow probably is an important factor in the perception of spatial relations between objects. The method is hardware accelerable with ability to run in real-time.

2. Our Method

The key part of the algorithm is to determine the boundary of the soft shadow correctly. This means, that we shall be able to determine the umbra and penumbra regions of the shadow receiver correctly. Secondly, we will make an estimate of the visibility of the area light source as seen from every point $\mathbf{x}$ in the shadow region. For these purposes, we will introduce some simplifications. First, a planar n-polygonal light source is approximated by a circular disk in order to determine the umbra and penumbra regions fast and to be able to use a single parameter, t, for the estimation of the visibility function. For the hardware implementation of the visibility calculation, we will introduce an extension of the shadow map concept - a so-called D-buffer, which among other things stores distances from the center of the light source to the umbra and penumbra volume, respectively. The method thus contains elements from both the shadow volume and shadow map methods.

The individual parts of the method are described in the following sections: Section 2.1 describes the procedure for generating the umbra and penumbra volumes. Sections 2.2 and 2.3 give an introduction to the rendering procedure. Sections 3 and 4 describe each of the two render passes. Sections 5 and 6 present the results and discus these along with suggestions for future work. Finally, we conclude in section 7.

2.1. Umbra and Penumbra Volumes

For this particular method, a fast procedure for generating umbra and penumbra volumes for a circular light source is required. As with shadow volumes, the procedure consists of: a) finding the silhouettes and b) extruding the actual volumes.

Silhouette Determination To determine the silhouettes, we test the front facing relationship between the

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.

particular face Φ and the light source for each face. Since the light source is a circular disk, the previously used method cannot be used directly. Instead we use the fact, that - in constant time - we are able to find a single point $\mathbf{p}_{near}$ on the disk which is used to determine whether no points on the disk are front facing to the given face. Similar, a point $\mathbf{p}_{far}$ determines whether all points are front facing. This is illustrated in Figure 3(a).

Assuming the normals are normalized, we find

$$\mathbf{p}_{near} = \mathbf{c} + r((\vec{n} \times \vec{n}_c) \times \vec{n}_c) \quad (1a)$$

$$\mathbf{p}_{far} = \mathbf{c} - r((\vec{n} \times \vec{n}_c) \times \vec{n}_c) \quad (1b)$$

where r, $\mathbf{c}$ and $\vec{n}_c$ are radius, center and normal of the light source, respectively. If $\mathbf{p}_{near}$ is in the negative half-space of Φ, so is every other point on the disk. Similarly if $\mathbf{p}_{far}$ is in the positive half-space, then every other point is too.

Using this property, the polygons are marked as being front facing to either the whole light source (all-frontfacing), or to no part of the light source (none-frontfacing). An edge belongs to the umbra volume silhouette if it connects a none-frontfacing face with one that is not. Similarly, an edge belongs to the penumbra volume silhouette if it connects an all-frontfacing face with one that is not. It must be noted that some special faces are neither all-frontfacing or none-frontfacing. These faces are located between the two silhouettes, and we denote these *silhouette faces*. The silhouette faces are described further in section 3.3.

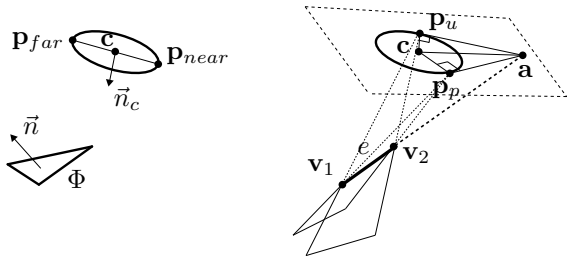
Volume Extrusion Once the edges defining the silhouettes are found, the volumes can be extruded. This is done similarly to the way a shadow volume is extruded. Here, the point light source is used as the extrusion point, that defines the direction, in which the silhouette edge is extruded.

When an area light source is used, we calculate separate extrusion points for each umbra and penumbra volume quad. These are denoted $\mathbf{p}_u$ and $\mathbf{p}_p$, respectively. In the following, we describe how to calculate these points for an edge e, which is assumed to be part of the silhouette for both volumes. This is not a general case, since an edge might be part of only one silhouette. $\mathbf{p}_u$ and $\mathbf{p}_p$ must be the two points each containing a plane through e that is tangent to the circle defining the light source.

If e, defined by the points $\mathbf{v}_1$ and $\mathbf{v}_2$, and the light source are not parallel, we find the intersection $\mathbf{a}$ between the light source plane and the line defined by e.

$$\mathbf{a} = \frac{\overrightarrow{\mathbf{cv}_1} \cdot \vec{n}_c}{\overrightarrow{\mathbf{v}_2\mathbf{v}_1} \cdot \vec{n}_c} \overrightarrow{\mathbf{v}_1\mathbf{v}_2} + \mathbf{v}_1$$

If $\mathbf{a}$ is outside the circle, $\mathbf{p}_u$ and $\mathbf{p}_p$ are found using regular 2D geometry, which can easily be deduced from



(a) Silhouette determination (b) Quad extrusion

Fig. 3. The points used to determine silhouettes and extrude the volume quads

Figure 3(b). If $\mathbf{a}$ is inside the circle, the extrusion points are undefined. Fortunately, this situation never occurs for a silhouette edge in a closed two-manifold mesh.

The special case where e is parallel to the light source require special treatment, see [17].

As seen in Figure 2, the extruded quads of the umbra volume need to be trimmed as they will otherwise intersect. Similarly, the penumbra volume requires extrusion of triangles between the quads in order to close the volume. The way this is done is completely dependent on the shape of the light source, but a solution is to form a connecting triangle between the extruded quads. This is correct for the triangular light source shown in Figure 2, and the approach can be used as an approximation for other light source shapes. In the case of a circular light source, it is preferable to extrude more triangles using interpolation between $\mathbf{p}_p$ for each of the two adjacent edges.

2.2. Rendering Procedure

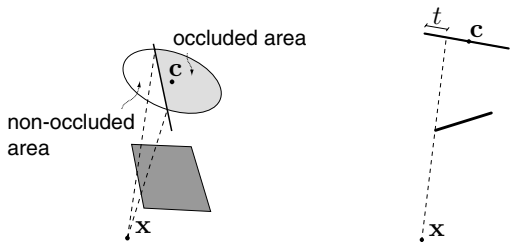
The proposed rendering method is a multi-pass algorithm. The first render pass renders the objects of the scene and the umbra and penumbra volumes as seen from the center of the light source. This is done using a vertex program where distances to the rendered entities are written to the individual color channels of the framebuffer. The result of this render-pass is stored in a *D-buffer*, similar to the way the shadow map method uses the Z-buffer. The main difference between the two buffers is, that the shadow map contains only one depth value, whereas the D-Buffer contains several distances. This render pass is where the method differs from other shadow volume based methods, where the volumes normally are rendered in screen space to the stencil buffer.

The second render-pass renders the objects in screen space using a fragment program to calculate visibility. This fragment program uses the distances from the D-buffer by performing a texture look-up and calculates the visibility using the retrieved values.

In order to make the generation of the umbra and

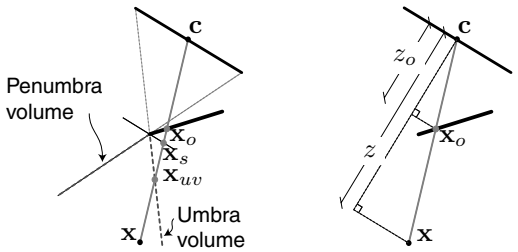
IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.



(a) Light source intersection (b) 2D view

Fig. 4. The 2D projection of the problem.



(a) D-Buffer (b) Z-Buffer

Fig. 5. Difference between shadow map and D-buffer. In the shadow map, only z_o is stored, whereas the D-Buffer stores the distances $|\mathbf{cx}_o|$, $|\mathbf{cx}_{uv}|$ ($|\mathbf{cx}_{pv}|$) and $|\mathbf{cx}_s|$

penumbra volumes fast, we have limited the scene to contain only one circular light source.

2.3. Parameterization

To calculate the visibility, V, we introduce a crude simplification:

Assumption 1 *For a given* $\mathbf{x}$ *in the penumbra region, the light source is divided between occluded and non-occluded area by one and only one straight line.*

The cases where $\mathbf{x}$ is outside the penumbra region are trivial, since the light source seen from such a point is either completely occluded or completely non-occluded, hence $V = 0$ or $V = 1$.

This enables a projection of the problem into 2D, which simplifies the problem into finding the parameter t, as illustrated in Figure 4. In general, the partition of the light source can be any curve resulting in both convex or concave areas.

Once t is found for the given $\mathbf{x}$, the problem is projected back to 3D using a parameter function, that describes the visible area of the light source. We have thus redefined the visibility function to a parameter function $V(t)$.

3. D-Buffer

The parameter t can, as it will be shown in section 4, be found from various distances along the line between the center $\mathbf{c}$ of the light source and the arbitrary point $\mathbf{x}$ on the receiver. These distances are stored in the *D-Buffer*, which can be thought of as a modified (and augmented) shadow map.

3.1. Definition

The differences between the ordinary shadow map and the D-Buffer is shown on Figure 5. The shadow map is generated by rendering the scene as seen from the light source and storing the contest of the Z-Buffer as shown in Figure 5(b). Instead of z-values, the D-Buffer stores distances from $\mathbf{c}$ to various geometry in the scene as illustrated in Figure 5(a).

The distance $|\mathbf{cx}_o|$ is the distance from $\mathbf{c}$ to the occluder. In the case where $\mathbf{x}$ is not placed in the umbra region $|\mathbf{cx}_o|$ and $|\mathbf{cx}|$ will be the same. The distances $|\mathbf{cx}_{pv}|$ and $|\mathbf{cx}_{uv}|$ are used to get information of distances from the light source to the penumbra and umbra volumes, respectively. $|\mathbf{cx}_s|$ is used to normalize the volume distances and describes the distance between the light source and the silhouette edge. The distance $|\mathbf{cx}_s|$ can be calculated when the volumes needed to determine $|\mathbf{cx}_{uv}|$ and $|\mathbf{cx}_{pv}|$ are rendered. In the case where the line $\mathbf{cx}$ intersects the umbra volume (see Figure 5(a)), the calculation is performed using the following formula for the particular silhouette edge e.

$$|\mathbf{cx}_s| = \frac{dir(\mathbf{c}, \mathbf{x}_{uv}) \cdot \vec{n}}{dir(\mathbf{c}, e) \cdot \vec{n}} dist(\mathbf{c}, e) \qquad (2)$$

where $\vec{n}$ is the normal of the light source, $dir(\mathbf{c}, e)$ and $dir(\mathbf{c}, \mathbf{x}_{uv})$ are normalized direction vectors from $\mathbf{c}$ to e and $\mathbf{x}_{uv}$, respectively. $dist(\mathbf{c}, e)$ is the distance between $\mathbf{c}$ and e which is uniquely defined because of assumption 1. If the line $\mathbf{cx}$ intersects the penumbra volume, $|\mathbf{cx}_s|$ is calculated using $|\mathbf{cx}_{pv}|$ instead of $|\mathbf{cx}_{uv}|$.

3.2. Rendering to the D-Buffer

The values $|\mathbf{cx}_o|$, $|\mathbf{cx}_{uv}|$ and $|\mathbf{cx}_{pv}|$ are generated by rendering the occluder and the volume geometry as seen from $\mathbf{c}$. Instead of using standard matrix multiplication to transform the vertices from world space to light source view space, the z-coordinate of the vertices is calculated as the distance between the vertex and $\mathbf{c}$ using the euclidian distance formula. This can be done by programmable vertex engines.

The distance $|\mathbf{cx}_s|$ is generated simultaneously with $|\mathbf{cx}_{uv}|$ and $|\mathbf{cx}_{pv}|$ by implementing equation (2) in the same vertex shader, that performs the specialized view transformation. This way, $|\mathbf{cx}_s|$ is generated without constructing additional geometry. This vertex shader must have the silhouette edge e passed as a parameter.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.

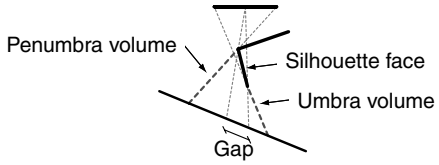


Fig. 6. Silhouette faces can cause undefined values in the D-buffer

Furthermore, color masking can be used to ensure that the distances are rendered to the appropriate color channels of the texture, that stores the D-Buffer.

During rasterization, the distances are interpolated. This yields inaccurate values, since the interpolation scheme is based on linearly distributed values instead of spherically distributed values which is the case for the D-Buffer. This causes problems if the occluding objects consists of very large polygons without subdivisions. In practice, this artifact is virtually unnoticeable.

3.3. Silhouette Faces

When the volumes are rendered into the D-buffer, a particular special case often arises, when a given edge is part of only one of the two silhouettes. When rendering only the volume quads into the D-buffer, gaps appear in the D-Buffer because of the presence of silhouette faces, as described in section 2.1. This is shown in Figure 6.

In the gap, neither $|\mathbf{cx}_{uv}|$ values nor $|\mathbf{cx}_{pv}|$ values are rendered as part of either the umbra or penumbra volume, respectively. The straight-forward solution to this problem is to render the silhouette faces as a part of the umbra volume.

There are two types of silhouette faces. The one shown in Figure 6 is front facing to less than half of the light source area. The other type is front facing to more than half of the light source area. To determine which category a given silhouette face belongs to, we check which half-space of the faces' plane the center of the light source lies in. If the center is in the negative half-space, it belongs to the category shown in Figure 6.

4. Visibility Approximation

The visibility V is calculated by a parameter t, which can be found from values in the D-Buffer. It can be shown using the 2D-projection that two congruent triangles exist and that these share an edge. This property yields

$$t = \begin{cases} \frac{|\mathbf{cx}_s|(|\mathbf{cx}|-|\mathbf{cx}_{uv}|)}{2|\mathbf{cx}_{uv}|(|\mathbf{cx}_{uv}|-|\mathbf{cx}_s|)} & , |\mathbf{cx}| < |\mathbf{cx}_o| \\ 1 - \frac{|\mathbf{cx}_s|(|\mathbf{cx}|-|\mathbf{cx}_{pv}|)}{2|\mathbf{cx}_{pv}|(|\mathbf{cx}_{pv}|-|\mathbf{cx}_s|)} & , |\mathbf{cx}| \geq |\mathbf{cx}_o| \end{cases} \quad (3)$$

The second branch of equation (3) is found by observing symmetry around $\mathbf{c}$ where $t = \frac{1}{2}$. This symmetry is only valid in the cases where the given edge is a silhouette edge in both the umbra and penumbra volume. However, when rendering the silhouette faces as part of the umbra volume, equation (3) is valid in general. In equation (3) only values from the D-Buffer and $|\mathbf{cx}|$ appear. This is essential for the possibility to calculate t (and thereby V) for each pixel in the framebuffer.

When using a circular light source, the function $V(t)$ can be found by integrating the equation of a plane circle. The function obtained by this is, however, quite complex and needs to be approximated by a simpler function. As noted in [21], a reasonable approximation would be to use

$$V(t) \simeq 3t^2 - 2t^3 \quad (4)$$

The described visibility calculation works for many scenes where assumption 1 holds. The cases where the assumption does not hold are:

1. Concave and/or overlapping occluders
2. No umbra region
3. Acute silhouette corners

In the first case, the problem is to determine from what occluding geometry the values stored in the D-Buffer should come from. This choice must be made, since the visibility function is only dependant of one parameter. When the desire is to be able to handle concave occluders, the best solution is to choose the geometry with the greatest distance from the light source. This can be done using depth test when rendering to the D-Buffer. In this case, the visibility function should still be equation (4).

The second case occurs when the light source is much larger than the occluder. In this case, a given point on the receiver can "see" the light source from more than one side of the occluder. The problem becomes to render appropriate geometry for the umbra volume to the D-Buffer because of self-intersection. This can be handled by geometrical trimming of the umbra volume, but this is not trivial. It is much easier to use depth test to ensure storing of proper distances in the D-Buffer. The visibility calculated using equation (4) will in this case yield values that are too low. Depending on the scene, an alternative function might be preferable.

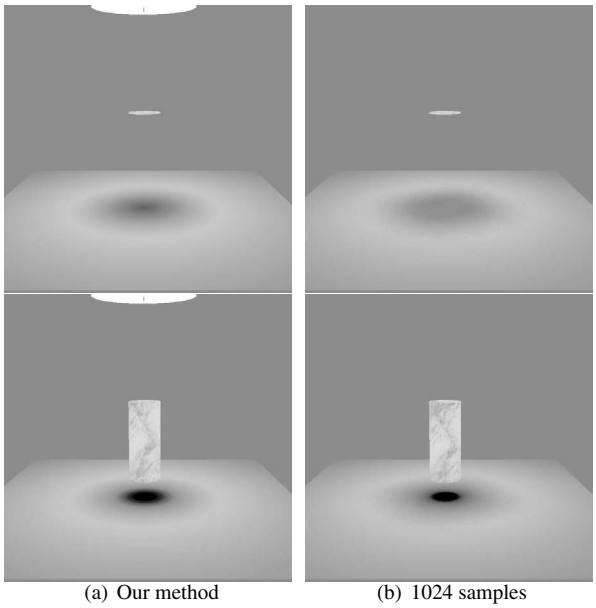
The third case is hard to improve, because of the way visibility is calculated. The artifacts due to this violation are fortunately less visible than the other two cases.

5. Results

The proposed method for approximating soft shadows was implemented using OpenGL, and the Cg language was used for the GPU-programming. The test system was a 3Ghz PC equipped with a QuadroFX 3000 Graphics card. The fragment program needed to calculate the

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.



(a) Our method (b) 1024 samples

Fig. 7. Scene as in Figure 1 where the umbra region is determined correctly. 7(a) is rendered with our method, and 7(b) is ray traced using 1024 samples of the light source

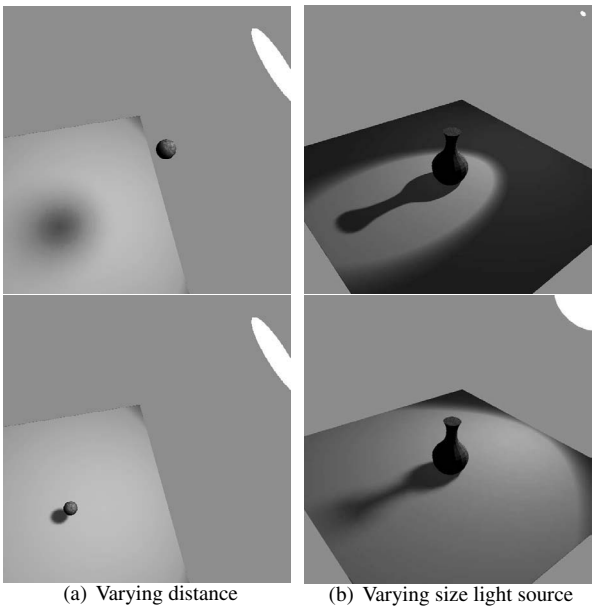
visibility contains 21 lines in the current implementation. Due to this simplicity, it is also possible to implement the method with the older NV20-architecture by using Register Combiners, as described in [17].

Figure 7 shows how the method handles the situation sketched in Figure 1. The images in Figure 7(a) are made by our method, and Figure 7(b) are ray traced reference pictures, where visibility is calculated using 1024 samples on the light source. These sequences show that the method calculates the shadow boundaries correctly.

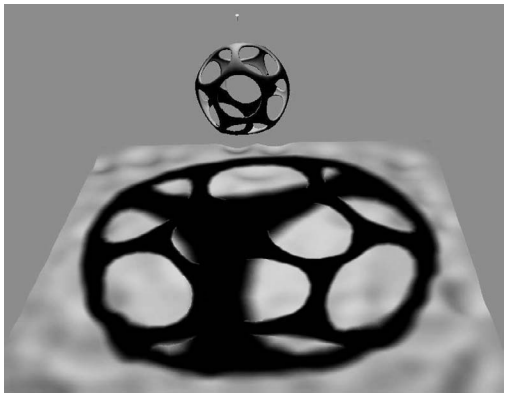
Figure 8 shows the capabilities of the method. In Figure 8(b) it is noticed that the penumbra region expands correctly as the light source size increases. Similarly, it is noticed in Figure 8(a) that the penumbra region diminishes as the occluder approaches the receiver. Figure 8(c) shows a scene with a fairly complex occluder. This image shows that the method is capable of casting shadows onto arbitrary geometry, which is the case because

Table 1. Performance of different scenes

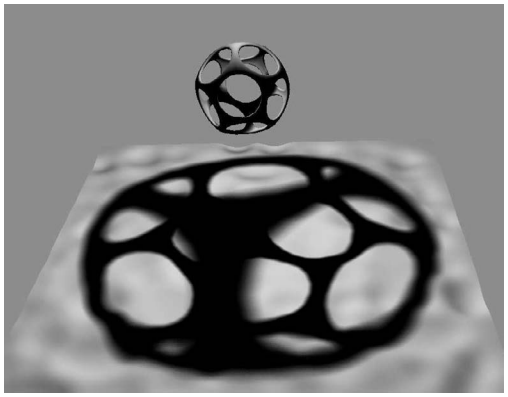
Scene	tri. count	FPS at different resolutions		
		300*300	600*600	1k*1k
Fig. 8(a)	224	212	120	61.0
Fig. 8(b)	960	107	75.0	45.0
Fig. 8(c)	5760	14.2	13.0	12.1
Fig. 8(c) *	1440	28.4	21.3	16.9
* Simplified occluder geometry				



(a) Varying distance (b) Varying size light source



(c) Complex occluder. Rendered using our method



(d) Same as above ray traced using 1024 samples

Fig. 8. Different scenes showing the capabilities of the method

visibility is calculated per pixel. The reference picture is shown in Figure 8(d). The overall performance of the

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.

method is shown in table 1. It is noticed, that the method is fill-limited for scenes with occluders of low complexity, and CPU-limited for scenes with high complexity.

6. Discussion

The presented method is based on conserving the properties mentioned in the introduction. Here, it was noted to which extent the shadows depend on having correct shadow boundaries.

Figure 7 shows how the correct shadow boundaries give more information of the shape of the occluder than the method proposed in [3] where a single silhouette is used. The method from [3] would render exactly the same shadow for the two different occluders.

Another very important benefit of our method is that it is implementable on a broader set of hardware architectures including the X-Box and all Nvidia GPUs since the GeForce3. Since the current implementation of the fragment program uses about 35% fewer instructions than the one used in [3], a performance increase is achieved compared to their method.

6.1. Optimizations

The performance of the method could be improved if some optimizations were made in the implementation. The current implementation uses no coherence between each rendered frame, which means that everything is calculated from scratch for each frame. This is done to ensure complete interactivity as everything in the scene can be changed in real time. However, if the volumes only were updated when the relation between occluders and light source is changed, it could be possible to improve the overall performance of the method.

Another point where the current implementation could be improved is the way the volumes are generated, since the method becomes very CPU-limited when the complexity of the occluder yields many geometry calculations done by the CPU. There is a possibility that a hardware accelerating method for volume generation could be used to improve this. It is, however, not trivial to accelerate the volume generation in current vertex engines, as the number of vertices needed for the various volumes is not constant. If future vertex engines were to allow scene geometry to be stored in the memory of the graphics hardware, this might be exploitable in order to achieve hardware accelerated volume generation. Furthermore, the method presented in [5] for generating shadow volumes entirely on the GPU could possibly be enhanced to generate umbra and penumbra volumes.

6.2. Artifacts

The primary drawback of the method is the limitations caused by the assumption. The fact that the light source for a given $\mathbf{x}$ only can be occluded by one convex object should diminish the number of scenes where the method is applicable. However, The implementation shows that in situations where the assumption is violated, the method still provides renderings that are reasonably satisfying, and in the cases where the assumption is valid, the results are quite convincing.

An example of this is seen in Figure 7(a) where the concave occluder violates the assumption. As noted earlier, the D-Buffer stores only the distances of the furthest of the overlapping occluding faces. This yields artifacts in the penumbra region of the occluder's self shadow, but this problem is not very noticeable in contrast to an incorrect penumbra region on the receiver object. This is because self shadows generally have small penumbra regions since distances between typical occluders and typical receivers are greater than distances within concave occluders. Furthermore, the stencil buffer is used to avoid "overwriting" umbra regions.

Another case where the assumption is not valid is when the occluder is very small as shown in the top image of Figure 8(a). In this case $\mathbf{x}$ can "see" the light source from more than one side of the occluder. The approximate visibility becomes too low, which makes small occluders cast more significant shadows than they physically ought to. This artifact can be reduced as shown in Figure 7(a), where a more suited visibility function is used ($V = \sqrt{t}$).

In the case where the angle between two connected silhouette edges is very acute, the penumbra region will have some artifacts due to the simplification of only dealing with occlusion of one of the two edges.

Because the method uses projective texturing, sampling artifacts are bound to occur because of the limited resolution of the texture. This artifact is usually handled by implementing the light source as a narrow spot light, thus using as much as possible of the resolution in the projection texture. This issue is discussed in [4].

6.3. Future Work

As mentioned earlier, the primary limitation is the restriction of only handling one intersection of the light source. This topic is where future expansions should focus. When dealing with more intersections, the D-Buffer must contain more values than the current method. This expansion could be performed by applying the *Depth Peeling* technique described in [10]. This would enable storing more intersections. The main problem lies in determining the area function when more intersections are present. Figure 9 shows this problem for two intersections of a circular light source.

As seen in the Figure, the angle α must be taken into account when the area function V is defined. The angle between some vector in the plane of the circle and the

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.

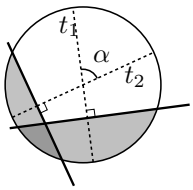


Fig. 9. Two silhouette edges cause a circular light source to be intersected by two lines.

axis of t could be stored in the D-Buffer by projecting the normal of the quads of volumes onto the circle. α could then be found as the difference between the angle corresponding to t_1 and t_2 respectively. The problem remains to determine a function $V(t_1, t_2, \alpha)$ that takes into account the overlap between the two areas. One possibility would be to use a look-up texture, as done in [3], instead of calculating $V(t_1, t_2, \alpha)$ on the fly.

If more than two intersections must be taken into account, the problem becomes even more complex.

7. Conclusion

We have presented a new method for producing soft shadows with correct boundaries in real time. The method is implementable on a wider range of existing graphics hardware than other existing methods producing similar (or better) quality shadows. The limitations discussed do of course narrow the range of scenes where the method can give acceptable results, but as the image in Figure 8(c) shows, quite complex geometry can be used with decent results.

Furthermore, we have succeeded in in resolving the problem from Figure 1, since our method determines the shadow boundaries better than other methods. This is mainly because the choice of using umbra and penumbra volumes has proven successful. The artifacts are all related to the visibility calculation, which implies that this is the weaker part of the method.

Even though the images from the current implementation are limited to a circular light source, the method in itself may, as it has been noted, be used with other shapes of light sources. The modifications needed for this are relatively small.

References

[1] M. Agrawala, R. Ramamoorthi, A. Heirich, and L. Moll. Efficient image-based methods for rendering soft shadows. In *Proc. of Siggraph*, pages 375–384, 2000.

[2] U. Assarsson and T. Akenine-Moller. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Trans. on Graphics (TOG)*, 22(3):511–520, 2003.

[3] U. Assarsson, M. Dougherty, M. Mounier, and T. Akenine-Moller. An optimized soft shadow volume algorithm with real-time performance. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conf. on Graphics hardware*, pages 33–40, 2003.

[4] S. Brabec, T. Annen, and H.-P. Seidel. Practical shadow mapping. *Journal of Graphics Tools*, 7(4):9–18, 2003.

[5] S. Brabec and H.-P. Seidel. Shadow volumes on programmable graphics hardware. *Computer Graphics Forum*, 22(3):433–440, 2002.

[6] S. Brabec and H.-P. Seidel. Single sample soft shadows using depth maps. *Graphics Interface*, pages 219–228, 2002.

[7] A. T. Campbell. *Modeling Global Diffuse Illumination for Image Synthesis*. PhD thesis, Dept. of Computer Sciences, University of Texas, 1991.

[8] E. Chan and F. Durand. Rendering fake soft shadows with smoothies. In *Proc. of the 13th Eurographics workshop on Rendering*, pages 208–218, 2003.

[9] F. C. Crow. Shadow algorithms for computer graphics. *Computer Graphics*, 11(2), 1977.

[10] C. Everitt. Interactive order-independant transparency, 2001. Available at www.nvidia.com.

[11] C. Everitt and M. J. Kilgard. Practical and robust shadow volumes, 2002. Available at rlwww.nvidia.com.

[12] E. Haines. Soft planar shadows using plateaus. *Journal of Graphics Tools*, 6(1):19–27, 2001.

[13] J.-M. Hasenfratz, M. Lapierre, N. Holzschuch, and F. cois Sillion. A survey of real-time soft shadows algorithms. In *Eurographics*, 2003. State-of-the-Art Report.

[14] P. S. Heckbert and M. Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, CS Dept., Carnegie Mellon U., Jan. 1997.

[15] T. Heidmann. Real shadows, real time. *Iris Universe*, 18:28–31, 1991.

[16] W. Heidrich, S. Brabec, and H.-P. Seidel. Soft shadow maps for linear lights. *Eurographics Workshop on Rendering*, pages 269–280, 2000.

[17] B. Jakobsen, K. S. Petersen, N. J. Christensen, and B. D. Larsen. Implementing boundary correct soft shadows. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[18] F. Kirsch and J. Doellner. Real-time soft shadows using a single light sample. *Journal of WSCG*, 11(2):255–262, 2003.

[19] T. A. Moller and U. Assarsson. Approximate soft shadows on arbitrary surfaces using penumbra wedges. *Eurographics Workshop on Rendering*, pages 1–9, 2002.

[20] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *Proc. of Siggraph*, pages 23–30, 1985.

[21] S. Parker, P. Shirley, and B. Smits. Single sample soft shadows. Technical Report UUCS-98-019, Computer Science Department, University of Utah, October 1998.

[22] C. Soler and F. cois X. Sillion. Fast calculation of soft shadow textures using convolution. In *Proc. of Siggraph*, pages 321–332, 1998.

[23] L. Williams. Casting curved shadows on curved surfaces. In *Proc. of Siggraph*, pages 270–274, 1978.

[24] C. Wyman and C. Hansen. Penumbra maps: approximate soft shadows in real-time. In *Eurographics workshop on Rendering*, pages 202–207, 2003.

[25] Z. Ying, M. Tang, and J. Dong. Soft shadow maps for area light by area approximation. *10th Pacific Conf. on Computer Graphics and Appl.*, pages 442–443, 2002.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on February 17,2010 at 08:59:27 EST from IEEE Xplore. Restrictions apply.