Real-time Shadows in CG

Christian Luksch*
MatNr. 0525392

Institute of Computer Graphics and Algorithms
TU Vienna

Abstract

This work should give an overview of various shadow techniques used in real-time rendering. Proper shadows are a crucial part of realistic looking scenes and therefore a huge research field in computer graphics. We will focus on the two major shadowing techniques shadow volumes and shadow mapping, including most popular rendering approaches, whereby the most practical approaches should be pointed out. Moreover, the difficulties and problems of them will be discussed, followed by a summarization and comparison.

Keywords: Real-time Rendering, Shadow Mapping, Shadow Volumes

1 Introduction

Shadows are important for the realism of computer generated images and play an essential role in human perception. A virtual scene without shadows looks very artificial and irritates our perception, because we actually expect shadows like in a natural scene. Shadows help us to identify the location and orientation as well as the shape of objects in a scene. Especially in 2D images where the depth-information is lost, shadows make a significant difference. Real-time rendering is limited by certain constrains which make accurate shadowing difficult.

There are many differing shadowing techniques and approaches to implement shadows. In real-time graphics usually shadows are simplified to guarantee a certain frame-rate while the different techniques try to provide a robust shadow test for all kinds of settings.
Mostly shadows are defined by a binary condition, that says either a point is shadowed or not, using a point or directional light. This produces hard outlines, therefore this approach is called *hard shadows*. In nature this condition is insufficient, because light sources always have a certain size and area and therefore generate smooth shadow outlines. This so-called *soft shadows* are computational much more expensive, but on the other hand they increase the realism distinctly.
In real-time rendering polygon rasterization is used, which makes the shadow calculation more difficult, because triangles are treated independently and therefore lack global information about their surrounding environment, unlike than with ray-tracing, photon-mapping or other usually in offline rendering used techniques. For real-time rendering variants of the shadow mapping algorithm [Williams 1978] as well as the shadow volume algorithm [Crow 1977] are among the most popular techniques. Shadow volumes use some sort of geometry that represents the volume that is occluded from a certain light source. Everything inside such a volume is shadowed, everything else remains unshadowed.
Shadow mappings uses as complete different approach to determine shadow. Shadows are created by testing whether a pixel is visible from the light source, by comparing it to a depth image of the light's view stored in the form of a texture.

Before these two techniques will be discussed in detail, we will focus on some further shadow basic.

*e-mail: christian.luksch@aon.at

2 Shadow basics

A shadow is a region of darkness where light is blocked. It occupies all of the space behind an opaque object with light in front of it. The cross section of a shadow is a two-dimensional silhouette, or reverse projection of the object blocking the light. [sha 2007a]
In computer graphics a light source is either defined by an area, a point or just a direction, latter generally used in real-time graphics. Both cases are illustrated in figure 1.

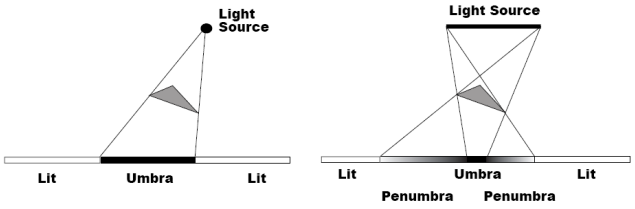


Figure 1: Hard shadows generated by a point light (left), Soft shadows from an area light source (right).

Hard shadows actually do not exist in nature. When light is emitted by some kind of light source the photons spreads from a the whole area of the light source, which results in soft shadow outlines. The size of the penumbra depends on the size of the light source, as well as the distance to the occluder and the shadow receiver. Certainly such shadows are much more difficult to realize in a real-time rendered scene, however the basic form of hard shadows are more practical. Therefore all techniques discussed in this work are aimed to generate hard shadows.
Figure 2 shows a comparison of hard and soft shadows in a demonstration scene.

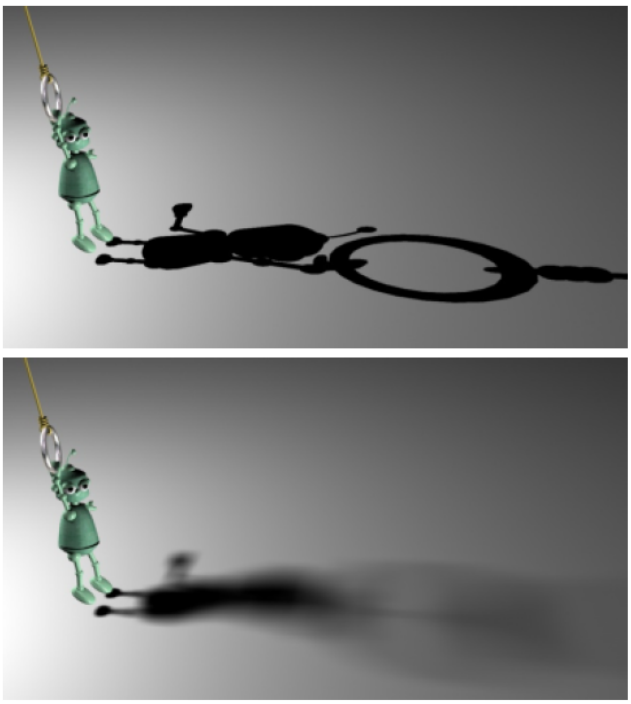


Figure 2: Compare of hard shadow (top) and soft shadows (bottom).

Moreover, the lighting of a surface is also defined by a remarkable amount of in-directional light reflected from other surfaces, while in real-time rendering mostly only the directional part is taken into account when drawing the surface material.

Shadow techniques provide a test to determine the amount of shadow for each point in the scene. When the scene is rendered the polygons are rasterized one after another, while there is no global information if the points are occluded from the light or not. Therefore this test needs to be prepared so that it can be looked-up when a polygon is rendered.

Some real-time rendering limitations already have been indicated as well as the simplified illumination, which is often simulated by the Phong illumination model [pho 2007], where the light is partitioned in three parts:

- Ambient part I_a: The amount of light that does not direct come from a light source, usually a constant term is used.
- Diffuse part I_d: The light reflected in the surface normal direction. To determine the amount generally the Lambert's cosine law is used. $L \cdot N$, dotproduct of surface normal N and light direction L.
- Specular part I_s: Light reflected by the surface normal into the camera. $(R \cdot V)^{\alpha}$, dotproduct of reflected light direction R and the view direction V.

Adding shadow to the material leads to the illumination equation:

$$I = I_a + S * (I_d + I_s) \tag{1}$$

where S is the shadow coefficient that has been provided by a certain shadow technique and determines the amount of shadow or the result of the binary condition of hard shadows. The formula means that there is no directional and specular light in shadows.
This is only one possible method to do the illumination and its also far from realistic, but when its combined with light- and specular-maps and other advanced texturizing methods quite authentically looking surfaces can be generated in real-time.

The next sections will explain the two main shadow techniques, shadow volumes and shadow mapping in detail, summarize common used rendering-techniques and point out their strengths and weaknesses.

3 Shadow Volumes

Shadow Volumes where originally presented by Frank Crow in his work "Shadow algorithms for computer graphics" [Crow 1977]. His technique uses a special geometry to describe shadows. It is build by extruding each shadow caster by its silhouette edges from the light's view towards the light direction until infinity, so that the geometry represents the volume that is occluded by the light. It can be constructed for point lights, spot lights and directional light sources and always produce pixel-accurate hard shadows.

3.1 Shadow-test

The shadow volumes can be used to determine if a certain point in the scene is shadowed by testing whether it is inside such a volume or not. This test is done by counting how many faces of the shadow volume are crossed by a ray from the point of view to the point that is processed. During this test a front-/back-face distinction is made and for each front face the counter gets increased and contrariwise for each back-face the counter gets decreased. If the counter is zero, either no shadow volume has been crossed or the shadow volumes has been crossed completely, hence the point is unshadowed. Otherwise if the counter is unlike zero, the point is anywhere inside a shadow volume and therefore occluded from the light.
Figure 3 clarifies the shadow-test by showing different cases in a typical scene. It shows that the illustrated test procedure properly works for all the points.

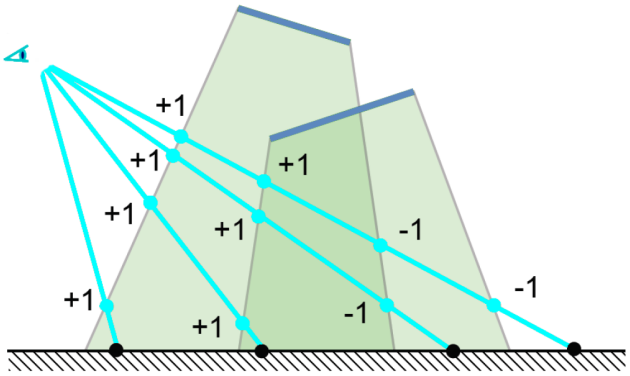


Figure 3: Determination if a point is shadowed using shadow volumes.

3.2 Rendering

If a method should be used in real-time rendering, there need to be an efficient way to process the shadow-test. The shadow volumes test can be easily implemented using today's graphics device. This

off-loads the CPU from work and it can do other important things. All that is required is a special buffer to do the counting, therefore usually the stencil buffer [ste 2007] is used.
Often shadow volumes are automatically associated with the stencil buffer, leading to the wrong conclusion that fast shadow volumes are impossible without stencil buffer support. Roettger et al. showed that the following algorithms can also be utilized on a normal color buffer in combination with certain blend states to emulate the counting. [Roettger et al. 2002]

3.2.1 Heidmanns original approach

The original approach using a stencil buffer has been introduced by Tim Heidmann [Heidmann 1991] and it is often called *Depth-pass*. His approach has some weaknesses and therefore multiple improvements exists, but it is crucial to understand the basic principle.
First the scene gets commonly drawed only using ambient and emissive lighting contributions. Now the color and depth buffers contain the color and depth values for the closest fragment rendered at each pixel. The next step is to build the shadow mask by rendering the shadow volume polygons into the scene but disabling color- and depth-writing and just updating the stencil buffer. This is done in two steps. First all front-faces of the shadow volumes gets drawed using hardware face culling while the stencil buffer is increased on depth pass, this is where the name comes from. Next the shadow volumes gets rendered again by only drawing their back-faces and decreasing the stencil buffer.

After this process the stencil value tells us if a pixel is shadowed or not. This stencil mask is now used to re-render the scene with the appropriate light configured and enabled. To ensure that only the right unshadowed pixels gets re-drawed, stencil testing with a zero stencil value and depth equal test is enabled.
In an elder sample included in the DirectX 8 SDK the whole process of re-rendering is replaced by rendering a black alpha-blended full-screen quad that uses the stencil mask to darken the shadowed regions, while the scene in the first pass was rendered lit. The performance gain is surely quite remarkable, but the simplification limits the material authenticity. This approach does not cooperate with material that use the Lambert's cosine law. The amount of shadow will be calculated twice e.g. on the backside of an object. As well as if one look at the Phong illumination, equation 1 of section 2, the specular highlights which do not appear in shadows would be visible.

The testing-process can be repeated for multiple light sources, clearing the stencil buffer between rendering the shadow volumes and summing the contribution of each light. [Everitt and Kilgard 2002]

3.2.2 Depth-fail

While looking at Figure 3 one might already have noticed that the view point is outside the shadow volumes, but in an interactive scene the case with the view point inside a shadow volume can not be avoided. In this situation the shadows would be inverted. Knowing that the view point is inside such a shadow volume, the re-rendering condition just need to be inverted for proper shadowing, but testing if the view point is inside a shadow volume is not very simple and would drastically reduce the performance.
Therefore an a little modified rendering algorithm for shadow volumes, which properly works for all camera positions, has been introduced around 2000. This method swaps the order of back-face and front-face culling and the stencil buffer is changed only on depth-fail unlike depth-pass. This so-called *Depth-fail* or also called *Carmack's Reverse* is more robust than Heidmanns original approach and the today's convenient way to implement the shadow volumes shadow-test. [van Waveren 2005]

3.2.3 Near- and Far-plane clipping

Because of rendering shadow volumes uses the normal view-projection-matrix, near- and far-plane clipping is performed as well. With arbitrary scenes, this may (and, in fact, often will) clip and slice the infinite shadow volumes. Handling the shadow volume in this way leads to incorrect shadow depth counting that, in turn, results in glaringly incorrect shadowing.

A suggested solution to the shadow volume near plane clipping problem, using Heidmanns original approach, is capping off the shadow volume's intersection with the near clip plane. The problem with near plane capping of shadow volumes is that projecting all back-faces to the near-plane can be difficult. This can create shadowing artefacts that appear as exceedingly narrow regions of the final scene where areas that clearly should be illuminated are shadowed and vice versa. These artifacts are painfully obvious in animated scenes. Furthermore the near-plane intersection will burden the CPU. [Everitt and Kilgard 2002]

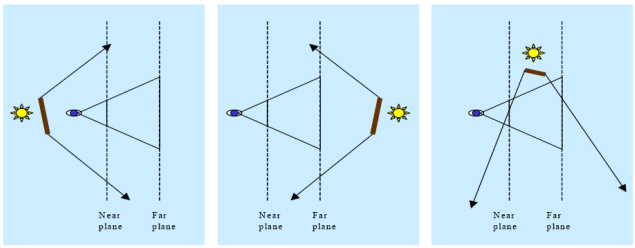


Figure 4: Three cases where capping fails because some or all pixels requiring capping are covered by neither a front-nor back-facing polygon.

The work of Cass Everitt and Mark J. Kilgard [Everitt and Kilgard 2002] presents a robust solution to solve the clipping and capping problem. They use the Depth-fail approach to render the shadow volumes. Therefore has to be ensured that shadow volumes clipped by the far-plane get capped correctly. Assuming that infinite shadow volumes are always clipped by the far-plane, they use a projection matrix with the far-plane at infinity to render the scene. With their matrix the depth buffer precision is only reduced marginally. The shadow volume is build by projection the silhouette edges from light view to infinite eye view. Their technique provides the opportunity for 3D games and applications to integrate shadow volumes in a robust way, but unfortunately it requires the CPU to build the shadow volume caps.

3.2.4 Correct depth-pass stencil shadows

In performance evaluation between the original *depth-pass* and Everitt and Kilgard's method, the latter approach is slower, because its robustness comes at the cost of speed. For better performance Hornus et al. presented a new algorithm short called *ZP+* (depth- or actually *Z*-pass plus). [Hornus et al. 2005] Considering a case where the occluder shadow volume is partly clipped by the near-plane, the original *depth-pass* method will fail. This is because exactly at the intersected area a part of the shadow volumes is not

drawed correctly and the stencil information is wrong, which leads to incorrect results.
The $ZP+$ approach constructs a frustum from the light's position to the camera near-plane, which contains all the clipped off parts of the shadow volumes, needed for a correct result. Rasterizing this geometry projects its fragments onto the original near clipping plane where it can be used to properly and robustly initialize the stencil buffer.
Because of arithmetic precision errors the virtually initialized caps may differ from the correct silhouette edges projected on the near-plane, which causes wrong pixel artefacts at these edges. However, Hornus et al. described a procedure to eliminate all the artefacts. All together the performance of the $ZP+$ approach is generally faster than other capping methods and behaves particularly well with large meshes.

3.2.5 GPU generation

So far the generation of shadow volume geometry has not be discussed in detail. It was assumed to be build by the CPU. Because shadow volumes required to be re-build every time the light or the object position has changed, which often is at every frame that is rendered and very CPU intensive. Furthermore shadow volume generation also needs to re-compute the skinning and tweening of animated objects on the CPU which can not be handled in the same way. Also numerical differences between the CPU and GPU precission can lead to strange artifacts.

In the book ShaderX Chris Brennan describes a way to use vertex shader combined with some preprocessing to remove the need for all CPU computations and therefore allows the GPU to do all the work. [Brennan 2002]
The approach is to pre-build a special geometry for each object which can be extruded to the shadow volume for every light position. The mesh additionally contains a quad for each edge with the same length than the edge and zero width. The four vertices of the quad consist of two vertices and the normal of one adjacent triangle and the other two vertices with the normal from the other triangle. During the render process the normal gets tested with the light direction either if the triangle is a front- or back-face. Silhouette edges are the border of a front-facing and a back-face triangle. Then each back-face gets extruded toward the light direction and with the additional quad, which now gets stretched, the mesh represents the proper shadow volume and still is complete and closed. The inserted quads are indicated in figure 5.
The whole process can be implemented using a short vertex shader, which makes additional vertex transformations no longer problematic, while resulting in a good performance.

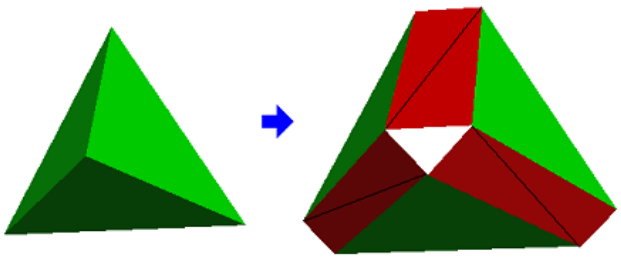


Figure 5: Shadow volume mesh generation, basic mesh (green) inserted polygons by Chris Brennan's approach (red).

Another approach which avoids the creation of shadow volumes on the CPU has been presented by [Brabec and Seidel 2003]. To do the silhouette detection, first all vertices of all shadow casters get assigned a unique index, as well as all the edges. Since connectivity and index numbers remain constant, this can be implemented as a pre-processing step.
In a shader all meshes vertices are transformed to world position and rendered with their index to an ARGB off-screen render-target with floating-point precision.
The next task is the classification of possible silhouette edges. It consists of checking the front-/back-face condition of the two connected triangles with respect to the given light source. Like in the world space transformation step all edges are rendered as single points to an off-screen surface. All edges get passed the four indices of the four vertices of the two triangles. The triangles are reconstructed by looking up the vertex-texture with the vertex indices and then the silhouette detection can be done using the sign of the plane equation according to the light source position of the two triangles. If the signs of the two plane equation distances differ, the edge is marked as a silhouette edge and the result is written to the a second off-screen surface.
The marked edges can now be used to render the shadow volumes in a third pass by proceeding the two textures in a vertex shader, which renders the shadow volumes geometry. During development of this approach the capability of reading textures in the vertex shader stage was not available at that time. Therefore they were restricted to use a very slow mechanism, that transfers the data from the two buffers to host (CPU) memory and immediately downloads the same data as vertex attribute data.

The benefit of both techniques is not only the gain in speed, what is most important is that shadow volumes can now easily be generated for geometry that is transformed by programmable vertex engines. The algorithms itself relies on shader capabilities of available graphics cards.
What in both works is missing a statement that their approach can be used with Everitt's and Kilgard's solution for the clipping problem described in the previous section or any other solution for that kind of problem, but theoretical it should be possible without major modifications.
Further a direct comparison of the two approaches has not been published so far and because there is no continuing work of the second one it can not be done jet. Anyway the first approach works very well and does not need to setup any extra render-targets and can be run with only a few hardware requirements.

3.3 CC Shadow Volumes

The major problem of the traditional shadow volumes algorithm is that the rasterization of the shadow volumes can be expensive in complex scenes, because the shadow volumes usually produce much rasterization overhead. Thereby an 8bit stencil buffer might easily overflow when the number of shadows visible between the eye and the object surface is too high.
Main sources of unnecessary shadow volume rendering are:

- large regions of empty space
- shadow casters completely enclosed in other shadow volumes
- shadow generation on not visible parts of the scene

There haven been some early approaches to reduces these problems. Lengyel suggests using the scissor test to restrict shadow volumes withing the light bounds in his article. [Lengyel 2002] McGuire et al. improved upon Lengyel's algorithm by adding culling and using the depth bounds test to further restrict shadow volume rendering. [McGuire et al. 2003]

The CC Shadow Volumes algorithm performs the optimizations on a larger-scale. It has been developed to minimize the rasterization costs. The CC stands for culling and clamping. [Lloyd et al. 2004]

First culling is used to remove nested shadow volumes or shadow volumes that do not produce shadows in the final image. Therefore two list of potential shadow receivers and casters are build.
Potential shadow receivers are objects that are visible from the eye point of view. They can be calculated using frustum culling or occlusion queries. This step is often done anyway before the scene is rendered.
In a second step all potential shadow casters are calculated, which means that they are visible from the light's view. Additional all previously calculated shadow receivers are used to further limit the shadow casters.
Figure 6 (b) shows the result of the culled shadow volumes.

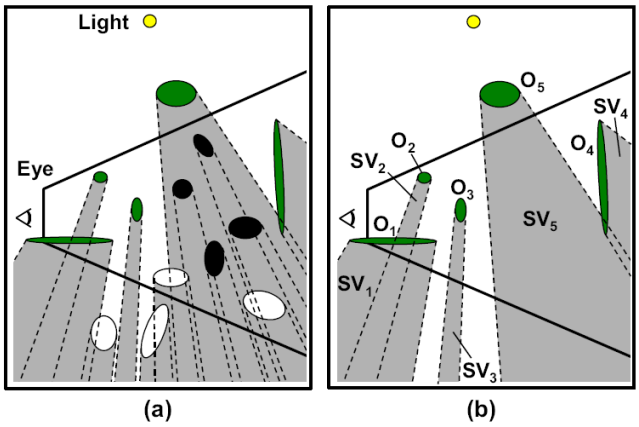


Figure 6:

Clamping further reduces the size of the shadow volumes so that only the minimum needed of the shadow volumes get rendered. Usual shadow volumes extend from the shadow caster towards the light until infinity, but they often include large unnecessary parts.
Lloyd describes two different clamping techniques, continues and discrete clamping.
Continues clamping uses the axis-aligned bounding boxes of the shadow receivers projected on the light image-space to determine the z-interval CV_i of the shadow volumes needed to be drawn correspond on the receivers. This step is entirely performed on the CPU.
In contrast to continue clamping discrete shadow clamping uses the GPU to test for shadow receivers within discrete shadow volume intervals. The view-frustum is split into slices. The shadow volumes get tested using occlusion queries to determine the slices within the shadow volumes intersect with shadow receivers. The clamped shadow volumes are more exact and save more fill-rate, but the additional occlusion queries are very expensive and will only be worth the effort for very complex shadow volumes.

The result of these two techniques is shown in figure 7, continues clamping in (c) and discrete clamping in (d)

Continuous and discrete clamped shadow volumes can be rendered in a similar manner. Both performed well in the tested scenes. A drawback of the culling and clamping is that artefact can occur which also can lead to incorrect shadows.
CC Shadow volumes are not ideal for every scene, only if the performance is limited by the shadow volume rasterization they are a good trade-off to use the CPU in combination with occlusion

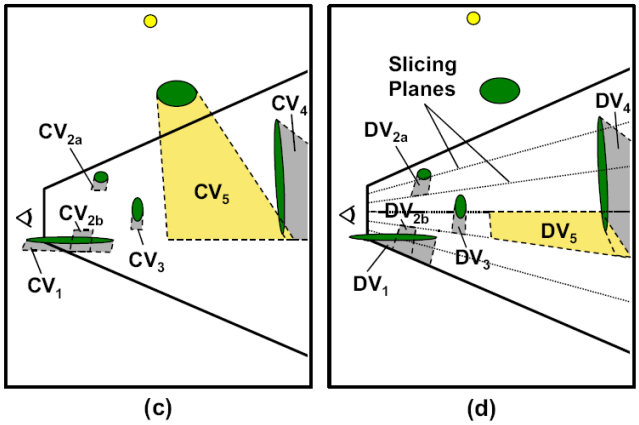


Figure 7: Continues (c) and discrete clamping (d).

queries to minimize the rasterization effort. A comparison of traditional shadow volumes and the CC shadow volumes is shown in figure 8.

Figure 8: Comparison of standard shadow volumes and CC shadow volumes.

3.3.1 Split-plane shadow volumes

Split-plane shadow volumes is an other approach to improve the performance of shadow volumes. In the final performance evaluation of the *ZP+* method, the results showed that in some cases *depth-fail* performs faster than the new method. So the question is to find a way to decide which method to use. Samuli Laine presented a novel method for rendering shadow volumes, which chooses between these two methods locally a per-tile basis and reduces the number of updated stencil buffer pixels by a high factor. [Laine 2005]

Both shadow volume rendering methods generate the same result and provide a robust shadow test, while one counts all intersections of a ray between the camera the surface and the other from the surface until infinity. Laine new approach merges both stencil buffer adjustment conditions to a new one $S\Delta$:

$$S\Delta = \begin{cases} +1 & \text{if} \quad z_{frag} < z_{pixel} < z_{split}, \quad facing = front \\ -1 & \text{if} \quad z_{frag} < z_{pixel} < z_{split}, \quad facing = back \\ -1 & \text{if} \quad z_{frag} \geq z_{pixel} \geq z_{split}, \quad facing = front \\ +1 & \text{if} \quad z_{frag} \geq z_{pixel} \geq z_{split}, \quad facing = back \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

With this condition the stencil buffer adjustment is defined by the depth value in the depth buffer z_{pixel} and the depth of the shadow volume fragment z_{frag} in relation to split-plane depth value z_{split}. The split-plane must remain the same, while processing one shadow volume, but it can be different for the every shadow volume. z_{split} can then be calculated by the plane equation. Worded this means, that only stencil values are updated when the shadow volume fragments are either between the split-plane and the camera depth value z_{pixel}, where the $ZP+$ method is used, or when the fragments lie behind z_{pixel}, but is in front of the split-plane, where *depth-fail* is used.
This generally yields a reduction in the number of stencil buffer updates, when the split-plane is setup wisely. The split plane should split the shadow volume efficiently in two halves, as seen from the camera. In Laine's work he presented different methods to align such a split-plane for a shadow volume and showed a hardware implementation for the complete process. To efficiently exploit fast hardware culling his implementation works on an 8×8 pixel tile basis.

3.4 Limitations

Because shadow volumes require some sort of geometry to build from, it is typically hard or not possible to construct shadow volumes for occluders that have no polygonal structure such as alpha-tested or displacement mapped geometry. [van Waveren 2005]
Alpha testing is a crucial method for efficient rendering of fine structures by using a texture with transparent parts like a chain link fence, leaves on a tree, or tufts of grass.
Displacement mapping can work on vertex or on pixel basis. The new vertex positions could be re-calculated while building the shadow volume, but on pixel basis it is hardly possible.
With that limitations one lose some important rendering techniques or proper shadowing, if using shadow volumes to produces the shadows in the scene. This is the big disadvantage someone has to live with.

3.5 Summary

The shadow volumes approach can be applied to a scene quite easily while most of the CPU intensive work can be offloaded to the graphics hardware. Anyway, with very complex geometries the amount of overdrawing increases drastically, but the amount can be restricted with more or less computational expenses. The shadow outline is very sharp and there are only a few artifacts that may appear, but they can be hidden quite good. Moreover, the mentioned limitation can be avoided in most situations, which makes shadow volumes a practical approach.

4 Shadow Mapping

In 1978, nearly about the same time when the Shadow Volume algorithm was presented, Lance Williams introduced another concept to process shadows called shadow mapping in the paper "Casting curved shadows on curved surfaces" [Williams 1978].
The idea is to use a depth comparison from the light view. Imagine a view from the light position onto the scene. Everything that is seen from that view appears in light, everything else behind those objects is in shadow. Therefore the light's view depth values are stored in some sort of a buffer and used when the scene is rendered to decide the lighting condition. For each pixel in the final scene the depth value from the light's view is calculated and tested with the stored one. If the depth value is greater than the from the light's view, the pixel is in shadow. This compare can be used to do the proper light calculation for all pixels in the scene.
Moreover, this approach supports shadows in a geometry independent way. It is applicable on towday's graphics hardware and therefore a widely-used shadowing algorithm in real-time rendering beside shadow volumes. [Hempe 2005]
In theory this approach seems robust, but it is not that easy than described above to get a useful result without any artifacts.

4.1 Shadow Mapping Algorithm

The shadow mapping algorithm can be implemented using today's graphic device hardware capabilities. Rendering a shadowed scene involves two major drawing steps. The first produces the shadow map itself, and the second applies it to the scene.
First the scene is rendered from the light's view. Thereto a common view matrix with an uniform projection which includes the hole field of view including all shadow casters can be used. During rendering all material are omitted and just the depth-value needs to written to the output buffer. As output buffer usually a single channel floating-point render-target with its own depth-stencil surface is used . The depth-value can also be encoded in an RGBA render-target using special encoding, which allows elderly graphic cards which do not support float-point render-targets to compute the shadow depth-map. If it is supported by the used graphics device also only a depth stencil texture can be used with disabling all the color writing. All that is required is the capability to set this depth buffer also called shadow map as texture for the shadow test to read back the depth value from the light view.
When the scene is rendered the shadow map gets projected onto the scene, which partitions the view volume of the light into two regions: the shadowed region and the unshadowed region. The visibility test is of the form

$$p_z \leq ShadowMap(p_x, p_y), \tag{3}$$

where p is a point in the lights image space. [Cass Everitt 2004]
To get the texture coordinates p_x and p_y the world position gets transformed into the light view space using the light's view-projection matrix. Then this coordinates gets transformed into texture space which means a scalation by $[0.5, -0.5]$ and a shift by $[0.5, 0.5]$ [pro 2007]. The final coordinates are now used to read the depth value in relation to the light from the shadow map.
Then the visibility test (Equation 3) compares this value with the into the light's view-space transformed word coordinates depth value, which theoretically would have been written to the shadow map if it would had been visible by the light's view and hence is in light, otherwise shadowed.
This approach is often called uniform shadow mapping (USM) or standard shadow mapping (SSM). Theoretically it provides a correct shadow test, but cause of finite resolution of the shadow map it leads to incorrect shadowing artifacts. Advanced shadow mapping techniques or other small modifications try to minimize them. This will be discussed in the next paragraphs.

Figure 9 illustrates a basic shadow mapping scenario. Image (a) and (b) is the scene rendered from the camera and the light's point

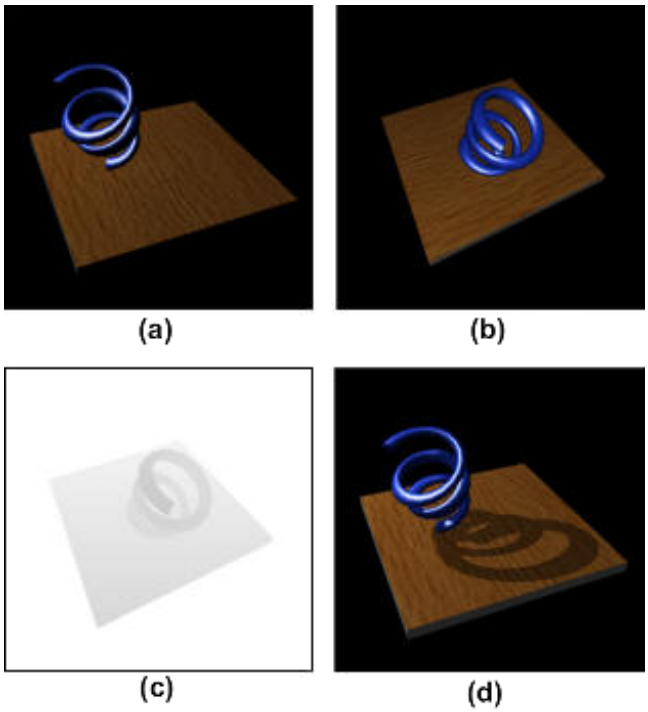


Figure 9: (a) The scene unshadowed, (b) the scene from the light's view, (c) the rendered depth map, (d) shadow mapped scene

of view. These two images are for demonstration only and are no actual render passes in the shadow mapping process. (c) is the depth map rendered from the light's view and (d) shows the result of the final scene rendered with the shadow map.

4.2 Problems and solutions

The process of shadow mapping involves several problem that have to be solved to get shadow mapping to work properly. This field has been well elaborated by the computer graphics community in the last years. The next sections will generally describe the problems followed by usable solutions.

4.2.1 Focusing the light-view

Before the shadow map can be rendered it is important to focus the light's view on that region that is actually needed for the shadow-test to gain a good utilization of the shadow map. This can be abstracted in set theory formulas where $\cap$ denotes a clipping operation and $\cup$ denotes the convex hull operation.
To get the focused intersection body H [Stamminger and Drettakis 2002] suggests to build a convex hull of the view frustum V and the light l, then clipping by the scene bounding volume S to get all visible objects and finally culling with the light frustum L.

$$H = (V \cup l) \cap S \cap L \tag{4}$$

Scherzer further suggests in his thesis to start with clipping the view frustum V with the scene bounding volume S to get all visible objects and then continue in the way described above,

$$B = ((V \cap S) \cup l) \cap S \cap L \tag{5}$$

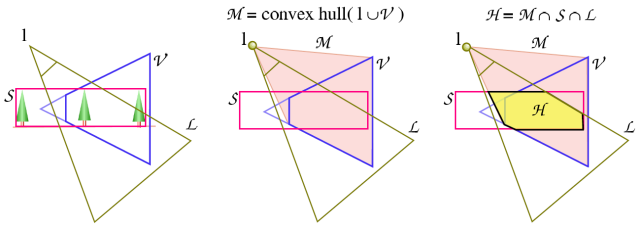


Figure 10: Finding the volume of all potential shadow casters.

which should give a smaller intersection body B and nevertheless always include all potential shadow casters.
This provides an robust way to calculate the focused region which is required for building the projection matrix to setup the light view. However, there might be cases where simpler or more involved methods for exact volume determination might be more efficient. [Scherzer 2005]

4.2.2 Aliasing

The shadow mapping depth-test compares a value from eye-space with one that is in light-space. Because of the finite resolution of the shadow map, for a fragment in eye-space the corresponding depth value in light-space can only be approximated by sampling the nearest value or doing some sort of interpolation. The lack of accuracy causes a blocky appearance and unaesthetic incorrect shadowing results. A robust algorithm would mean that the resolution of the shadow map should vary and should give us an individual depth value for each transformed eye-space fragment. [Scherzer 2005]
The whole shadow mapping process suffers from two types of aliasing artifacts, perspective and projective aliasing.

In a perspective view objects near the camera are larger than distant objects, however standard shadow maps have an uniform distribution of their resolution. The outcome is a shadow resolution that is too low for nearby objects and too high for distant objects in eye space. This effect is called perspective aliasing. The artifacts are biggest if the light direction and the view direction of the camera are perpendicular to each other and smallest if the light direction and the camera are facing in the same direction. [Scherzer 2005]
Figure 12 illustrates perspective aliasing.

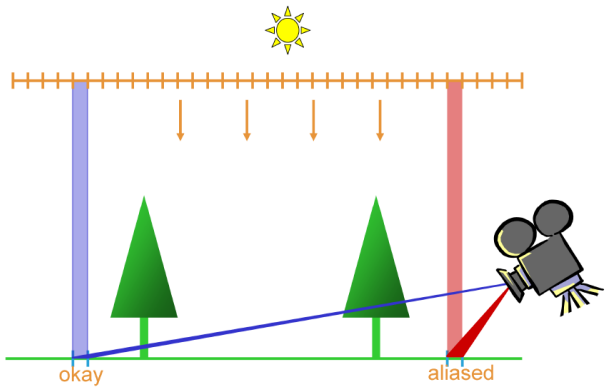


Figure 12: Perspective aliasing

Perspective aliasing can be reduced by optimizing the distribution of the resolution in the shadow map. This is complex topic and

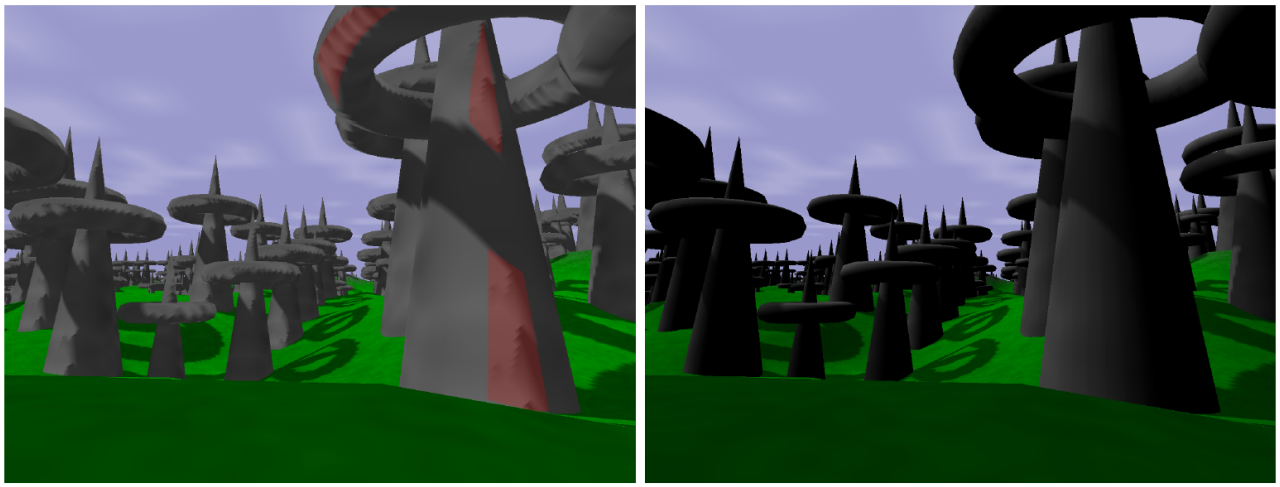


Figure 11: Projective aliasing artifacts in an outdoor scene (left), artifacts hidden by diffuse lighting (right).

will be discussed with advanced shadow mapping techniques in the next section. The goal is to ensure that for each point view-space a point in the light's view is stored to the shadow map. To satisfy this condition for arbitrary situations with one shadow map and one linear projection is not possible. This problems is especially in large environments with global directional lighting. In other not arbitrary cases e.g. flashlights or static lights in enclosed environments which have been bounded at level-design time, standard shadow mapping may be sufficient. [King]

Projective aliasing is independent of the camera, it only depends on the angle between the light direction and the surface normal. The effect is largest if the surface normal is perpendicular to the light direction, where the shadow map contains no depth information for the surface. It is difficult to counteract this shadow mapping problem because it is dependent on the scene geometry. Therefore projection aliasing cannot be solved by a simple global method that operates on the whole scene, but requires a detailed analysis of the scene geometry. A correct solution to projective aliasing remains an open problem for real-time shadow mapping approaches. [Scherzer 2005] Figure 12 illustrates projective aliasing on a sheer surface.

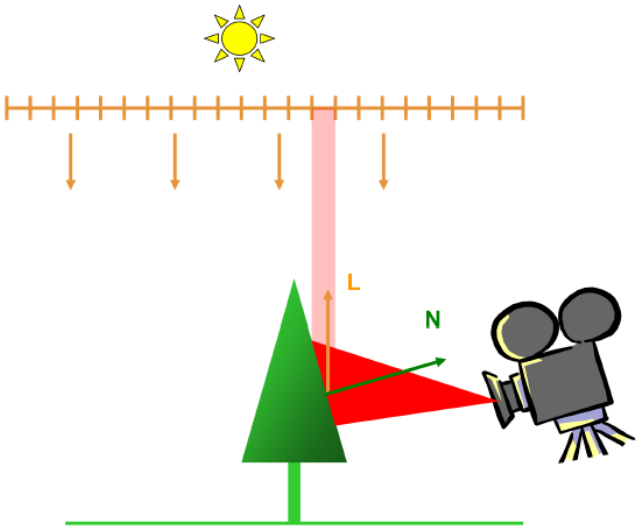


Figure 13: Projective aliasing

Even if there is no solution for projective aliasing the caused artifacts can be hidden quiet good.
A roughly perpendicular aligned surface to the light direction means that it will only be little illuminated in that cases, when using e.g. a Phong illumination model, equation 1 of section 2. The direct illumination calculated by Lambert's cosine law is reverse the error of projective aliasing caused by projecting the shadow map on the scene from the same light. Therefore projective aliasing artifacts can be hidden quite good by simulating little illuminated surfaces.

4.2.3 Depth biasing

Depth biasing is adding a certain amount to the depth value before it is actually compared with the depth value from the shadow map. This is necessary because at different point during the shadow mapping process inaccuracies occur. Aliasing as well as the quantization of the depth values and multiple arithmetic operations during the process lead to in inaccurate values which should be theoretically equal.
A sketch of this problem is shown in figure 15.

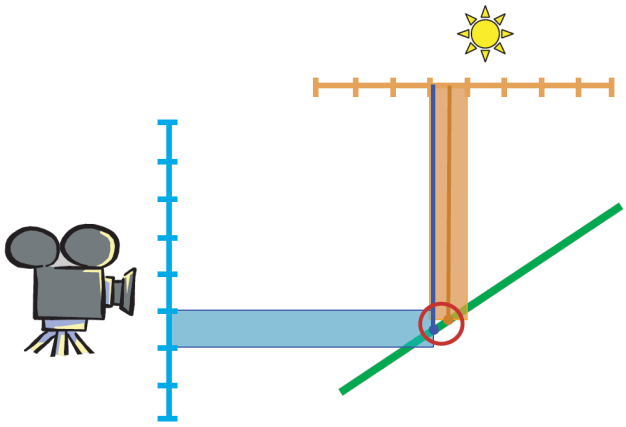
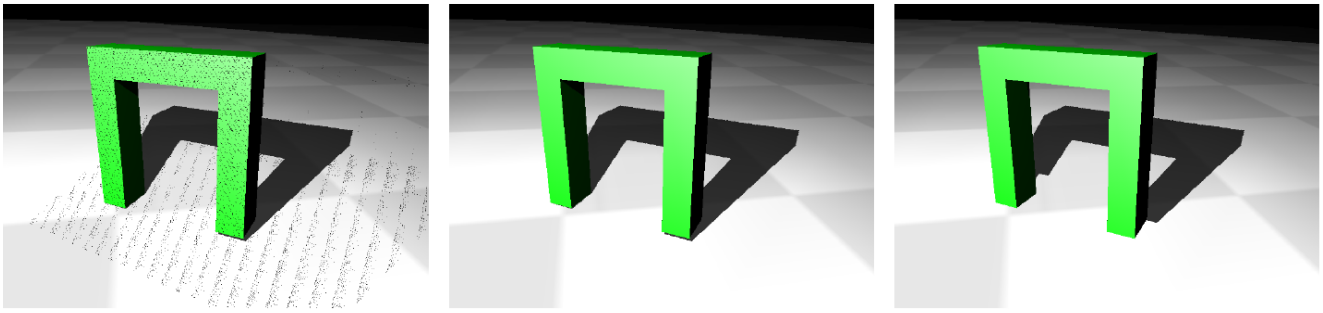


Figure 15: Incorrect self-shadowing.

This inaccuracies lead to incorrect self-shadowing artifacts characterized by shadowed spots in the middle of a lit surface.



(a) bias too low (b) just right (c) bias too high

Figure 14: Biasing scenarios

Therefore it is also often called shadow acne. Figure 14 (a) illustrates this effect.

The standard solution to incorrect self shadowing is a manually defined depth bias that is added to the shadow map, which solves in most cases the problem. If a more reliable avoidance of incorrect self-shadowing errors is needed the bias has to be quite large, which leads to noticeably misplaced shadows shown in figure 14 (c).
With a constant bias all depth values are moved the same amount, but actually the amount needed depends on the depth slopes of the polygon. For a depth slope near zero, hardly any biasing is needed, while for a polygon that is almost parallel to the light direction a big bias is appropriate. Therefore the solution using a slope-scale biasing value has be introduced. Here the bias is altered dependent on the depth slope of the polygon.
Figure 16 shows a sketch of biasing and slope-scale biasing.

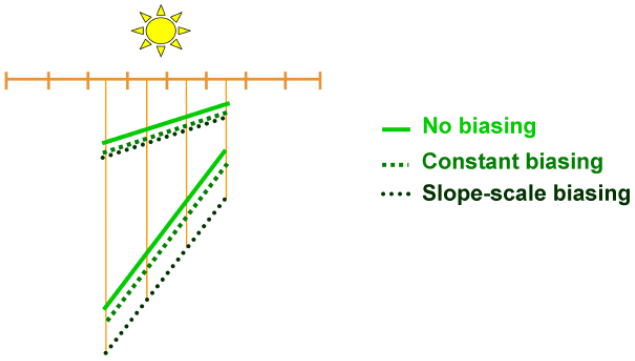


Figure 16: Biasing and Slope-scale biasing.

Because such biasing functions are crucial in computer graphics, nowadays graphics devices are capable of doing constant biasing and slope-scale biasing calculation in hardware on demand.

4.2.4 Omni-directional lights

A shadow map can only represent a limited field-of-view, which is kind of a problem when generating a suitable shadow map for a given point light source. In cases where a point light source radiates over the complete hemisphere or even omni-directional the traditional shadow mapping fails.

The basic solution for this problem is to use more than one shadow map. One way is to setup a cube map to cover the complete environment. Graphics hardware also directly support texture fetching using cube maps. However each face has to be rendered in a single pass, which can lead up to six passes for omni-directional light sources.

A better parameterization has been introduced with paraboloid mapping. Brabec et al. developed an implemented using this mapping for shadow map rendering of point lights. [Brabec et al.] With a single shadow map the complete hemisphere over a point can be represented. For omni-directional lights, only two shadow maps are required.
With a simple transformation vertices can mapped to their paraboloid coordinates. Unfortunately rasterization hardware is only capable of perspective correct interpolation. However a linear interpolation can be accepted, in case that the geometry is tessellated fine enough. The whole transformations can be easily implemented in shaders. The whole process is detailed described in the paper.
Figure 17 shows a scene illuminated by a hemispherical point light on the ceiling. On the right the parabolic mapped depth map and light's view is illustrated.

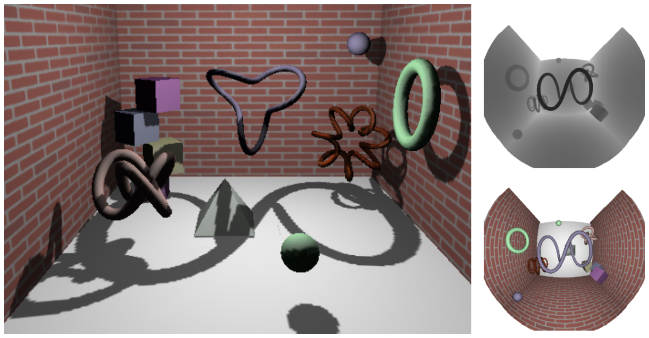
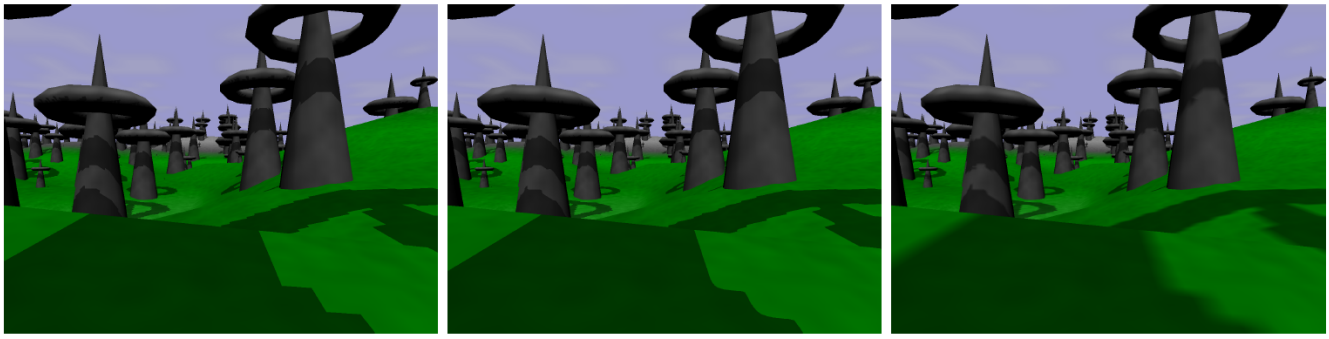


Figure 17: Scene illuminated by a hemispherical point light (left), parabolic mapped depth map and light view (right).

4.2.5 Filtering

Because shadow mapping is a texture based algorithm, filtering is important to increase the shadow quality. Standard shadow mapping with simple nearest filtering leads to very "blocky" looking shadows. Their outline is in fact sharp, but the silhouette of



(a) Nearest filtering (b) Bilinear filtering (c) Percentage closer filtering

Figure 18: Shadow map filtering operations

the shadow casters is not as clear as with shadows volumes. Figure 18 (a) gives a first impression on what will be discussed in this section.

The first thing one might think about when hearing of filtering is bilinear filtering. This takes the four closest pixels and linearly interpolates in horizontal and vertical direction to calculate the final sample. With shadow mapping this means that a interpolated depth-value is calculated and then the shadow-test is performed. Thereby the sharp outlines of the shadows will be sustained. The result comes closer to the shadow volume approach, but if the shadow map resolution is too low bilinear filtering leads to unsatisfactory results, as one can see in 18 (b). The further and more serious problem is the fact that the bilinear interpolation makes no sense for the depth values along the edges of objects. Interpolating generates a new non existing depth value, which does not accord to a surface in the scene, therefore the shadow test will not be appropriate as well as the shadow edges will be sharp and vulnerable to artifacts.

A special for shadow mapping developed kind of filtering is percentage close filtering (PCF). Percentage close filtering reverses the order in which the filtering and the comparison steps are applied of bilinear filtering. The interpolation is done on the results of the depth comparisons. This means that after the depth comparison a binary map results and on this binary map the filtering takes place. The result is smooth shadow boarders, hiding some of the blockyness of a unfiltered shadow map. To improve the smoothness of the filter result a larger filter kernel can be used at cost of performance. E.g a 4x4 box filter requires 16 texture accesses instead of 4 with the original 2x2 filter kernel. [Scherzer 2005]

A comparison of these filters is illustrated in figure 18. It shows that the result of nearest and bilinear filtering is not very convenient. The result of PCF is however quite practicable.

In the next section we will now discuss a technique called variance shadow mapping (VSM)[Donnelly and Lauritzen 2006], which attempts to improve the quality by focusing on the filtering problem. It enables to use common hardware filters like bilinear or anisotropic filters for depth map filtering.

4.2.6 Variance shadow maps

The variance shadow map (VSM) approach comes up with a new way to render the shadow map and to perform the shadow test, which enables to use hardware filtering and improve the shadow quality. [Donnelly and Lauritzen 2006]

The algorithm works quite simple and can be easily combined with other shadow mapping techniques. First of all the shadow map get rendered and instead of storing only the depth value also the squared depth is stored in the second channel, which can be interpreted as first and second momentum of a distribution. The result of a sampling operation on this texture with any filter will be the mean of a distribution of the depth values with the mean value μ in the first channel. Using the squared depth value of the second channel the variance σ^2 can be calculated using $\sigma^2 = E(x^2) - E(x)^2$. ($E$ stands for the statistic estimator of the mean value.)
As result, this places a bound on how much of the distribution can be concentrated far away from the mean. This bound is stated precisely with Chebyshevs inequality (on-sided version):

$$P(x \geq t) \leq p_{max}(t) \equiv \frac{\sigma^2}{\sigma^2 + (t-\mu)^2}, \tag{6}$$

on which the authors proved the equality of percentage closer filtering on with planar occluders.
The shadow test is performed by first testing the depth value of the fragment on $< \mu$, then it is unshadowed. Otherwise the variance is computed and with Chebyshevs inequality the amount of shadow p_{max} can be calculated.
Figure 19 compares this approach to common shadow map filtering techniques. As the figure demonstrates the results of PCF and variance shadow mapping is almost identical.

What can not be seen in the figure 19 is that light bleeding artifacts can occur in scenes with a high depth complexity on overlapping shadows when the variance is high. In practice these artifacts can be hidden very good and will not be noticeable. Also problems with numeric stability when using 16bit floating-point values are mentioned, but with 32bit precision there would not be any visible artifacts.

Filtering can now be done on the graphics hardware as well as usage of automatically generated mipmaps, however only the latest models support filtering of floating-point textures. Additionally the authors suggest to apply an optional gaussian blur on the shadow map in a separate pass to get smoother shadow borders. Because the effort is about the same than with standard shadow mapping, the performance is barely reduced.
To conclude, variance shadow maps are a simple and effective solution to filter shadow maps and reduce aliasing artifacts as well.

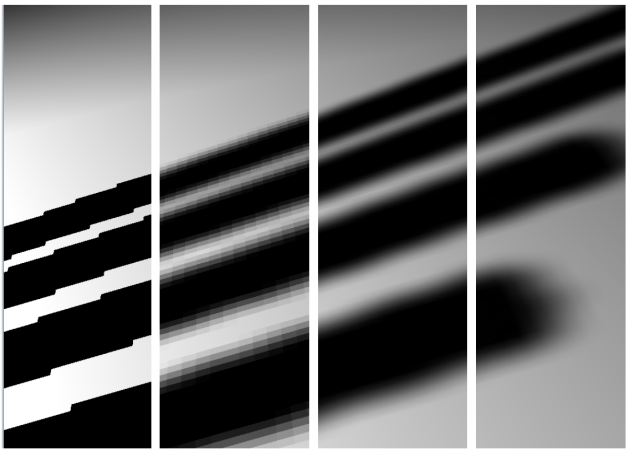


Figure 19: Comparison of filtering techniques, left to right: 1) Standard shadow mapping, 2) 5x5 PCF, 3) 5x5 bilinear PCF, 4) variance shadow mapping

4.3 Advanced shadow mapping techniques

Today's applications require flawless shadows, whereby many researches have worked on improving the shadow mapping process and therefore many advanced shadow mapping techniques exists. Their goal is to maximize the quality and concurrently keeping the additional expenses as low as possible. [Hempe 2005]

All the following techniques try to reduce the aliasing error. They can be categorized in how the approaches work. In this work we use the following categorization scheme:

- Warping techniques
- Shadow map partitioning
- Pixel accurate shadow maps

The next we will list the most popular techniques in these categories.

First there are techniques which try to solve perspective aliasing by introducing a view-dependent redistribution of the shadow map, so that objects near to the camera have more space in the shadow map. These class we can call warping techniques. The most popular techniques are:

- Perspective shadow maps [Stamminger and Drettakis 2002]
- Trapezoid shadow maps [Martin and Tan 2004]
- Light-space perspective SM [Wimmer et al. 2004]

The next techniques that will be discussed in this works are which that partition the logical shadow map into smaller parts and build a shadow map for each one. For each part an individual parameterization can be used too, which means a combination of warping and partitioning.
The most popular partitioning techniques are:

- Parallel split shadow maps [Zhang et al. 2006]
- Face partitioned shadow maps
- Plane optimal shadow maps [Chong and Gortler 2004]

An alternative kind of shadow mapping techniques depart from convenient straight forward rendering of shadow maps and compute a unique shadow query for smaller parts of the scene to guarantee an accurate shadow test for each pixel.
Thereto counts techniques like:

- Adaptive shadow maps [Fernando et al. 2001]
- Queried virtual shadow maps [Giegl and Wimmer 2007]

At first these techniques will be short explained followed by a final discussion in the summary.

4.3.1 Perspective shadow mapping

Marc Stamminger and George Drettakis developed the first real-time approach to reduce perspective aliasing with their technique called perspective shadow maps (PSM). [Stamminger and Drettakis 2002]
In the first part of their work they describe the aliasing problem of shadow maps in detail and formalized the amount of aliasing for point lights in a general formula

$$d = d_s \frac{r_s}{r_i} \frac{\cos\beta}{\cos\alpha}, \tag{7}$$

where the result d is the size of a projected shadow map pixel on the screen. Further d_s is the size of a shadow map pixel, or the reciprocal shadow map resolution; r_s and r_i the distances of the projections and $\frac{\cos\beta}{\cos\alpha}$ the amount of projection aliasing.
The following figure illustrates the combination.

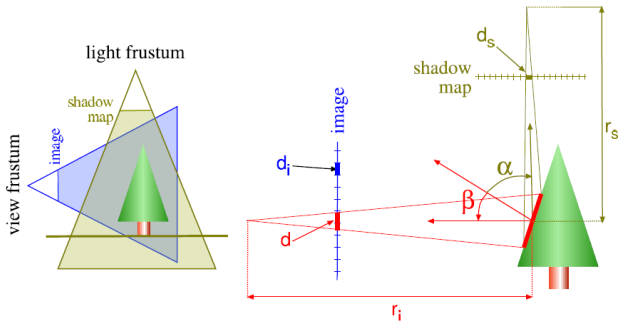


Figure 20: Shadow map projection quantities.

Due to limited memory, the shadow map resolution $\frac{1}{d_s}$ can only be increased up to a certain limit in practice. However perspective aliasing can be avoided by keeping the fraction $\frac{r_s}{r_i}$ close to a constant. To achieve this term their approach is to compute the shadow maps in post-perspective space of the current camera.
The scene gets normally mapped to post-perspective space, which is a unit cube and from that the shadow map is generated in this space by rendering an view from the transformed light source to the unit cube. In this process it has to be considered what the view projection means for the light source. A point light's position in post-perspective space can be transformed to infinity and therefore becomes a directional light, while rays of directional lights intersect at a finite point and therefore becomes a point light in post-perspective space, except in a few cases.

This basic idea does not considers objects behind the camera, which in general cases also can cast shadows in a scene. Therefore

Stamminger and Drettakis suggest to generate the shadow map from a virtual camera, which views the whole frustum of potential shadow casters, described in equation 4 in section 4.2.1. On the other hand the virtual camera reintroduces perspective aliasing and in worst case decrease towards standard shadow mapping, which means that the quality depends heavily on the light position in camera space.

The non-uniform distribution of perspective shadow maps makes the biasing problem worse, because the needed bias depends on the texel position, as well as on light position in relation to the post-perspective space. Moreover most of the shadow map resolution is wasted on objects near the camera and the quality of shadows in the distance is poor. A suggestion in the article is to move the near-plane as far to the front as possible by reading back the depth-buffer each frame to get the minimum used z-value. However, this is quite expensive, because it requires reading back a large amount of video memory, causes an additional CPU/GPU stall and does not work well with swizzled and hierarchical depth buffers. It is hard to say if it is really worth the desired effort. [Stamminger and Drettakis 2002]

In 2004 Kozlov publishes an article in the book "GPU Gems 2" where he further analyses PSM and present practicable techniques to improve the quality and robustness. [Kozlov 2004]
To avoid the virtual camera, Kozlov introduced an new projection matrix, which is capable of mapping vertices beyond the unit cube. The new projection increases the depth range, however a 24bit quantization is enough for reasonable cases.
In contrast to reading back the depth buffer to find the optimal near-plane position he refers to CPU based scene analysis, which avoid the readback and it should be accurate enough.
To achieve a better balance between the shadow quality of objects near and far from the camera Kozlov's suggestion is to increase the near-plane "virtually", when using his new projection matrix. This is done by moving the camera back like with the virtual camera, but while the near-plane remains at the same actual position.
Another improvement is to use cube maps instead of a common texture for the depth map. In post-perspective space most lights become point lights and it is more natural to use cube maps for them. The platform for the cube map faces are the back faces of the unit cube and the center of the cube map is given by the light position in relation to the unit cube. The number of required passes depends on the light position. If it is inside the unit cube all faces have to be rendered, in other cases where the light is outside only the required faces to cover the whole unit cube have to be rendered, which are only three in the best case. The cube map approach has better quality with the same total texture size as a single texture. The difference is the cost of the render target switches and the additional instructions to compute cube map texture coordinates in the vertex and pixel shaders. [Kozlov 2004]

With the new distribution of depth-values in the shadow map more accurate shadows can be draw than with uniform shadow maps. However, all together it makes perspective shadow mapping a very complex technique.

4.3.2 Trapezoid shadow mapping

Performing the shadow calculations in post-perspective space is rather unusual and the singularities makes the process pedestrian. Therefore trapezoid shadow maps follow a more analytical way to construct a perspective deformation for the shadow map. Figure 21 illustrates the basic idea of a trapezoid approximation (b) and compares it with a bounding box approximation (a) used by standard shadow mapping. [Martin and Tan 2004]

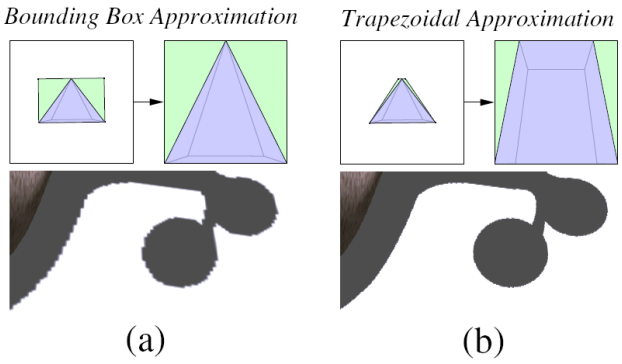


Figure 21: A bounding box approximation (a) compared with a trapezoid approximation (b).

Trapezoid shadow maps (TSM) assume that the eye frustum is seen from the light as trapezoid. Therefore the algorithm approximates a better utilization of the shadow map for the area within the eye's frustum as seen from the light by transforming the trapezoid to a cube. The distribution of depth samples along the view axis is controlled by a parameter called the 80%-line.
In the case that the projected near-plane lies within the projected far-plane from the light's view no trapezoid transformation can be calculated. Hempe illustrates this problem where TSM falls back to standard shadow mapping. Further he describes an enhancement of the original approach to smooth the reduction of the shadow quality when the view directions and the light directions become similar. [Hempe 2005]
A general drawback of TSM is that depth-biasing is worsened, therefore the authors recommend replacing the output depth with the non-warped depth instead.

4.3.3 Light-space perspective SM

About the same time when trapezoid shadow mapping was developed other researchers experimentalize with another approach based on the idea of perspective shadow maps, but using a different formulation for the perspective parameter. [Wimmer et al. 2004]

The light-space perspective shadow mapping (LispSM) approach uses a perspective transformation of the light space based on view and light direction with a free parameter n which determines the strength of the perspective transformation. The parameter n can be automatically calculated to minimize the perspective aliasing error. Scherzer further evaluates different approaches to calculate N_{opt} in his thesis and suggest a new formula in a general case beside the original one, but he points out that in the worst case, the duelling frusta case where the view direction points towards the light direction, a parameterization is insufficient.[Scherzer 2005]
In comparison to PSM the distribution of the depth samples is well balanced. The aliasing error is maximum at the near and far plane, but generally the distribution is quite even and therefore generate a good shadow quality at all distances.

4.3.4 Parallel split shadow map

The original approach of splitting the view frustum along the camera z-axis to approximate the continuously varying of the resolution along the distance from the camera was introduced by [Tadamura et al. 1999]. Parallel split shadow maps (PSSM)

are an advancement of this idea considering today's hardware capabilities of graphics devices. [Zhang et al. 2006] By the way a similar approach called *cascaded shadow maps* (CSM) has been implemented, but their details remains unpublished.

The process of rendering a shadowed scene with PSSM is illustrated in figure 22 and involves following steps:

1. Split the scene into m depth parts V_i.
2. Split the light's frustum into m smaller ones, each of which covers one split part also the objects potentially casting shadows into the part.
3. Render each part to a separate shadow map.
4. Render the scene, while sampling the according shadow map for the shadow test.

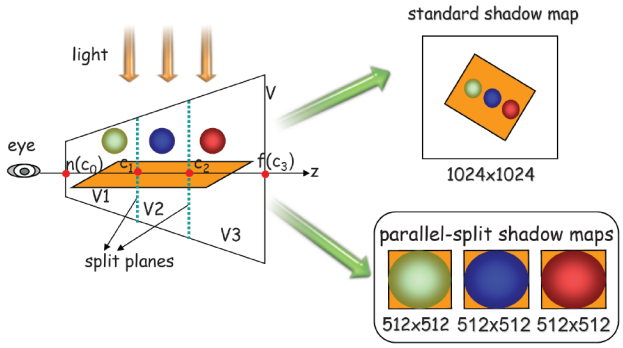


Figure 22: Splitting of the view frustum

To find the optimal split distances Zhang et al. evaluated an uniform and a theoretical optimal logarithmic split scheme to find a practical split scheme and came to the conclusion that a combination of both achieves a better tradeoff between theory and practice. They also propose using a small bias to suppress artifact that may occur on the borders between the split parts.
The authors reduced the required passes from $2m$ conceived by the original approach which does not consider hardware acceleration to $m+1$ by using a shader program to select the appropriate shadow map and sample the according depth value when the scene is rendered.

4.3.5 Face partitioned shadow maps

The idea of partitioning the view frustum along its faces came up during an analysis of the worst case of warping algorithms, in which the light comes from behind the camera and the view frustum approximates a square. Figure 23 (a) illustrate such a case. In fact, for high depth ratios, the near plane is very small and can be left out altogether. By stretching the sides slightly the near plane will be covered with only a slight increase in error. An illustration is shown in figure 23 (b).

This partitioning scheme results in trapezoidal parts which can be warped independently, for example Lloyd et al. suggest using LispSM and showed that the aliasing error is reduced and nearly uniform distributed in all light/camera configurations.
The implementation works similar to PSSM and also combining both has been suggested, however the number of shadow maps increases greatly, although aside from the case in figure 23 not always all faces have to be rendered. [Lloyd et al. 2006b]

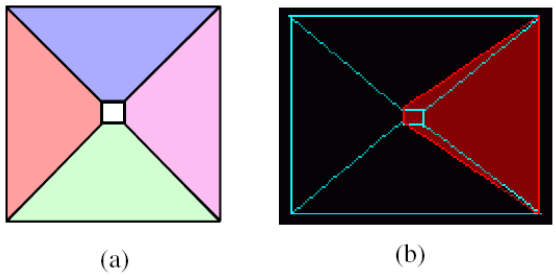


Figure 23: Face partitioning.

4.3.6 Plane optimal shadow maps

The plane optimal shadow mapping approach tries to ensure perfectly placed samples on an arbitrary plane seen by the camera. If the shadow is casted on a surface that is significantly different aligned than the plane the shadow quality will decrease. Because of that several planes of interest can be chosen, but this has to be done by the environment's author. For each of the planes a optimal transformation according to the camera from world to the light's image space is calculated by solving a small linear system. With that transformation the various shadow maps are rendered. When the final scene is rendered the best plane of a pixel can be selected in fragment shader.
However, there are some cases where the quality is worse than with standard shadow mapping. One case described by the authors is when the camera is very close to an optimal plane (e.g. the floor) and then looking at other shadows on surfaces not aligned to a plane of interest. [Chong and Gortler 2004]

4.3.7 Adaptive shadow maps

An alternative approach to solve the aliasing problem was introduced with adaptive shadow maps (ASM). They use a hierarchical grid structure representation of the shadow map, which is updated continuously during a walk through. [Fernando et al. 2001]

ASMs are organized as trees. Each node in an ASM tree has a shadow map of a fixed resolution and a partitioning of that shadow map into a fixed number of cells. Each of these cells may contain another tree node. The creation and deletion progress of nodes is determined by a cost benefit metric. The maximum number of nodes is limited by a from the user specified maximum amount of memory.
The determination of which resolution of the shadow map is required for a corresponding cell is provided by a test that uses hardware mip-mapping to approximate the projected area of a pixel. Therefore aliasing artifacts are generalized so that perspective and projective aliasing is minimized. Anyway, the complexity of the ASM requires a lot of processing on the CPU, which makes this technique too expensive for realtime rendering. However, high resolution shadows maps can be simulated, which generates nearly pixel accurate shadows. A comparison to standard shadow mapping is illustrated in Figure 24.

With today's graphics devices much of the ASM data structures can be handled by the GPU. Lefohn et al. showed a GPU-based approach, but it is limited by many expensive CPU readbacks and the frame-rate is still barely realtime. [Lefohn et al. 2005]
Both implementations of ASM are not very practical for current scenes, which can even have over a million triangles.

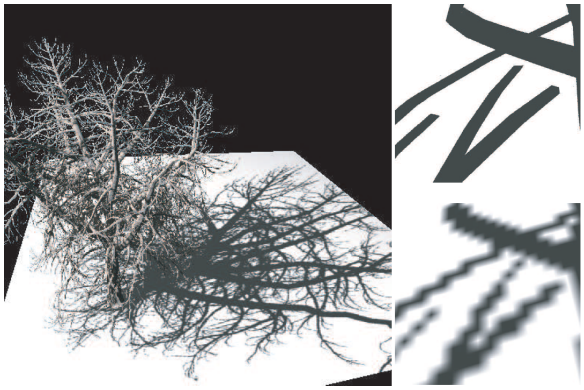


Figure 24: GPU-based ASM, simulating a 131072^2 shadow map resolution. The detail at the bottom right shows the result of a 2048^2 standard shadow map.

4.3.8 Queried virtual shadow maps

Another technique which simulates high resolution shadow maps is queried virtual shadow mapping (QVSM), but it performs better than ASM, because it uses GPU-based optimizations and supports a easy to use performance/quality tradeoff parameter. [Giegl and Wimmer 2007]

The approach is based on a simple algorithm that splits the virtual shadow map in equally-sized tiles, while only one shadow map is allocated of all of them. Then for each tile the shadow map is rendered, while the scene is immediately shadowed, so that the same shadow map can be overdrawn with the next tile.
This process could be implemented using simple multi-pass rendering, but the rendering overhead would be intolerable. Therefore the authors developed a technique derived from deferred shading, called deferred shadowing, with which the scene is only rendered once and the shadow from a tile can be fast applied by rendering a full-screen quad.
An important improvement to limit the number of passes is to start with one big tile, which is adaptively refined in a quad-tree like fashion only when necessary. The refinement is done until the improvement that was achieved falls under a certain level. The level is measured in changed pixels, which can be easily provided by hardware occlusion queries. The authors made a detailed performance analysis and came up with further optimizations to improve the performance, but keeping the same quality.

In comparison to the hardware implementation of adaptive shadow maps [Lefohn et al. 2005] this techniques performs much better and also offers a very accurate shadow test.

4.4 Summary

Shadow mapping follows a simple theoretically approach, but in practice there are many difficulties and problems, which result in unesthetic artifacts. There are various attempts to improve the quality with less expenses, many focused on the aliasing problem, but there are still cases where these techniques fail. However, the warping and partitioning techniques showed that they are able to provide a robust shadow test with little more computational expenses.

The benefit of the three warping techniques, (PSM, TSM and LispSM) is that they can be easily adopted from standard shadow mapping just by using different transformations during the shadow map rendering and the shadow test. With PSM the singularities of the post-perspective space have to be considered too. A further drawback of PSM is that the new distribution of depth samples is not well balanced from near to far objects and as noted before, PSM is an unusual technique. However, TSM and LispSM use with the 80%-line and N_{opt} a parameterization to balance the distribution of depth samples. The shadow quality of both approaches is similar with small variations.

Lloyd et al. further investigated this field and tend to optimze the distribution of depth samples over the whole shadow map. The projective transform, like used in the techniques so far, can diverge significantly from the optimal distribution, which is logarithmic. They presented a promising sketch with a new logarithmic shadow map parameterization, which decreases the aliasing error for a given resolution, or allows to reduce the size of the shadow map.
Unfortunately, logarithmic rasterization is not supported by current graphics hardware, but Lloyd et al. showed that it could be realized with only small enhancements to the current rendering pipeline. Anyway the amount of arithmetic operations is higher, but in savings of memory bandwidth, which however would exploit current hardware trends. [Lloyd et al. 2006a]

Unfortunately, a general drawback of the warping techniques is that the shadow map alignment depends on the view and the light direction, which causes the shadow edges to crawl in an animation. One suggested solution is to align the shadow map to a fixed vector in world space. As the view frustum moves, the shadow map is permitted to move only in increments of a shadow texel, eliminating the crawling. [Lloyd et al. 2006b]
In some special cases where light direction and view direction are nearly similar and in the case where the view direction points towards the light direction, also called the duelling frusta case, the shadow quality of warping algorithms degenerates to the of standard shadow mapping. With reparameterization no acceptable result can be achieved in that case. Wimmer also suggested partitioning of the view-frustum and building an individual shadow map for each part. [Wimmer et al. 2004]

Partitioning techniques require extra render passes according to standard shadow mapping and warping techniques, anyway the quality is significantly better, most notably because they are or can be combined with a warping technique. Thereby even in a duelling frusta case are enough samples to generate adequate shadows. When Lloyd et al. suggested even combining face partitioning with parallel splitting, they also mentioned that the partitioning scheme should actually be optimal chosen from frame to frame, but this can cause disturbing popping. [Lloyd et al. 2006b]

Finally with pixel accurate shadow mapping techniques, shadows can be rendered as desired with no artifacts. The computational expenses on the other hand highly restrict the performance, whereby these techniques barely find use in practice. However, queried virtual shadow mapping seems quite promising for upcoming applications, if and how practicable this technique is remains to be seen.

All together, shadow mapping is an attractive technique, because it works geometry independent. Even with this number of discussed problems it is a widely used technique and very important technique for real-time shadows. Shadow mapping is still a hot topic in computer graphics.

5 Comparison

In this section a comparison of the two major shadowing techniques, shadow volumes and shadow mapping is made. The following table gives an overview of the hard facts. Vantages are marked with +, disadvantages with - and an enumeration of the rendering cost is given.

Shadow volumes	Shadow mapping
+ pixel accurate shadows + independent of light type - restricted to geometry - much overdraw	+ flexible + simple and fast - aliasing artefacts - lots of techniques and hard to optimze - additional effort for omni-directional lights

Rendering costs:

• 2×Scene (illuminated and unshadowed with stencil mask) • 2×Shadow volumes (front and back faces)	• n× Scene from the light's view; number depends on type of light and the technique • Illuminated scene

Both techniques are well adapted on today's graphic hardware and make use of hardware capabilities. High quality shadows can be generated with both of them too.

6 Hybrid approaches

So far we have only discussed shadow volumes and shadow mapping as separate techniques. Finally we will discuss approaches which try to combine the benefits of both techniques, the perfect shadow edges from shadow volumes and the flexibility of shadow mapping. The two main hybrid approaches are:

- Shadow volume reconstruction from depth maps [McCool 2000]
- A hybrid approach which uses shadow volumes only for the shadow edges. [Chan and Durand 2004]

Next these approaches will be explained followed by a final discussion in the summary.

6.1 Shadow volume reconstruction from depth maps

The shadow volume reconstruction approach is aimed to generate sharp shadow volume shadows without to rely on the scene geometry. [McCool 2000]
McCool starts to point out that shadow volumes and shadow mapping are the only two approaches, which are practical for real-time rendering. After an introductive summarization of their vantages and disadvantages, he describes a combined approach which generates a single shadow volume for the whole scene from a shadow depth map. In detail the algorithm performs the steps as follows:

1. Render the shadow depth map from the light's point of view.
2. Use an edge-detection filter to get the silhouette edges.
3. Reconstruct the shadow polygons from the edges and build a shadow volume.
4. Render the scene using the reconstructed shadow volume for shadowing.

Depth map rendering is actually a very important step, because the result depends on the accuracy of this step. Theoretically any shadow mapping approach can be used, however in the original paper a standard hardware implementation is used.
The edge detection step uses computer vision techniques to find the minimal required shadow volume. The first derivation in combination with a threshold is used to filter the silhouette edges. Two sets of flags are generated, one horizontal *u* and one vertical *v*, but actually these flags are stored redundant per cell in a 4bit field which marks the edges around it. The result is illustrated in figure 25.

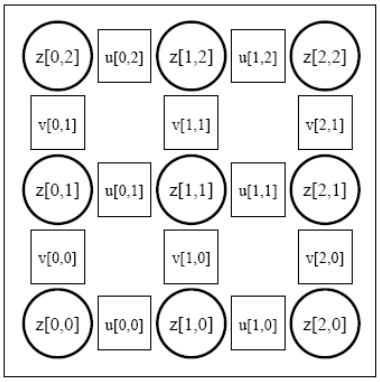


Figure 25: Horizontal and vertical edge flags.

False positive detected edges will not effect the final image, because they are culled by the depth map anyway, but false negative detected will result in a visible error. False negatives mostly occur at the end of silhouette edges when the depth contrast is to low. The threshold can be lowered carefully, but on the other side this introduces new false positives. Therefore a more complex algorithm to reduce false positives, which also involves the second derivation and two additional thresholds, is described in the paper. To improve the performance of this step graphics hardware can be uses.
Once the edges flags are set the shadow polygons can be constructed. A simple approach would be to generate a quad for each cell where at least one silhouette edge has been detected, but this will generate shadows with 90° jaggies. Therefore a polygon depending on the combination of edges of the cell (stored in the edges code) can be generated with a lookup table. Both approaches are illustrated in figure 26

Using multiple shadow maps is also no problem, when they are aligned carefully and have a small overlapping area to maintain continuity.
When the shadow volume is finally rendered, clipping problems will occur like described in section 3.2.3. McCool also describes a solution for such cases, but also someone might use the approach of Everitt and Kilgard.

In a final performance evaluation the author tests his implementation in different scenes. A direct comparison to shadow mapping and shadow volumes is unfortunately missing. What definitely can be said is that this hybrid approach will perform better in very complex scenes than using shadow volumes. However, the aliasing artefacts of the shadow map can not be

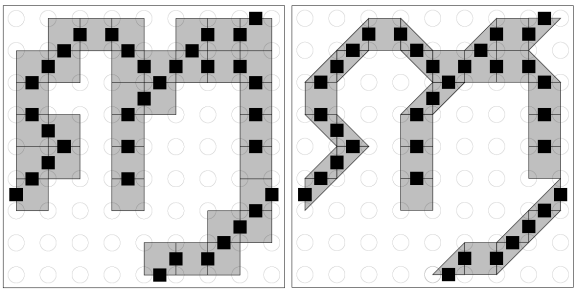


Figure 26: Simple polygon generation (left), polygon generation depending on the cell edges code (right).

removed completely. Figure 27 shows the reconstructed shadow volume and the artifacts of projective aliasing.

Figure 27: Reconstructed shadow volume.

McCool's original implementation is not up-to-date anymore, which makes an estimation of the usability of this approach difficult. The reconstruction process could be revisited to adapt better to today's graphics hardware.

6.2 An efficient hybrid shadow rendering algorithm

The hybrid approach of Chan and Durand has been conceived to improve the performance of shadow volumes. The decisive reason to use shadow volumes is that they generate sharp shadow silhouettes, while shadow mapping suffers from aliasing artifacts. Their approach uses the accurate shadow volumes for silhouette pixels and the faster shadow maps everywhere else. [Chan and Durand 2004]

Short summarized the algorithm for this hybrid approach works as follows:

1. Render a shadow map.
2. Build a mask for shadow silhouette pixels in the final scene.
3. Draw the shadow volumes only at masked regions.
4. Redraw the scene with the stencil mask for the shadow edges and shadow mapping for the rest of the scene.

To render the shadow map, standard shadow mapping is sufficient in any case. Aliasing is only noticeable at the shadow edges, which are rendered using shadow volumes anyway.
The computation mask for the silhouette pixels can be determined with percentage closer filtering, where the four nearest depth-values are tested. When the results disagree the pixels are tagged by writing 1 to the color buffer.
To omit drawing shadow volumes at the marked pixels, first the z-values of all non-silhouette pixels are overwritten with an unusual value using simple full-screen quad. Then the shadow volumes are rendered using the *depth bound test* to early discard drawing these pixels.
In figure 28 the results of these steps are illustrated. It shows the marked silhouette pixels of a simple scene and the shadow edges rendered with shadow volumes.

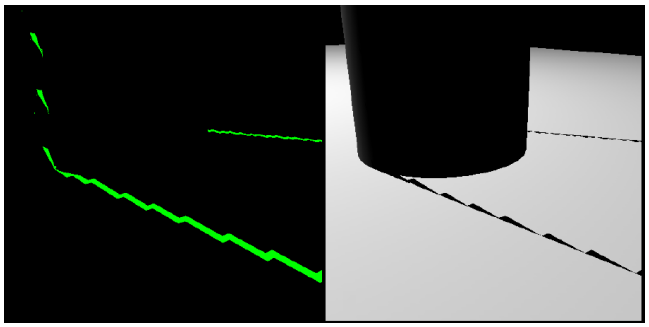


Figure 28: Masked silhouette pixels (left), shadow edges rendered with shadow volumes (right).

The last part of the paper is a performance comparison to convenient shadow mapping and shadow volumes. The performance gain results quite in the middle of both techniques. The authors also consider further improvements with Lloyds method, culling and clamping of shadow volumes, described in section 3.3, because most of the computation time is still spent with rendering shadow volumes.
Another important factor is that the performance also depends on how well the computation mask is implemented. Hardware supported computation masks would bring an major improvement.
This approach brings an important performance improvement of shadow volumes. The times in the timing charts also show, that the rendering time of the shadow map is only a minor part in comparison to the total effort. The reason is that already a low-resolution shadow map is sufficient.

6.3 Summary

Hybrid approaches are not widely spread, probably because they are much more effort to implement than a single standard technique and there are also quite good improvements of the standard techniques available. So that the question is: Why use a combination of two techniques, when one optimzed technique can do the job as well?

The general aim of the discussed approaches is to bring some improvements, but on the other hand they also inherit the weaknesses and restrictions of both techniques.
However, McCool's approach shows that in combination with shadow maps, shadow volumes can be made flexible as well.

With the second approach the combination results in an essential performance improvement over shadow volumes, while pixel accurate shadows are obtained.

7 Conclusion

The number of techniques summarized in this work show the huge area of shadows in real-time rendering and refer to much more techniques developed for shadows in computer graphics. Therefore no single solution for a robust shadow test, that fulfills all very different requirements, exists.
Many works do not only bring a new approach, they also conclude to open questions, where still a lot remains to be answered. The recently launch of the next generation of graphics devices opens new ways for real-time rendering, which bring up new research subjects that definitely will lead to new shadowing approaches too.

With the given overview of the field of real-time shadows in this work, the choice of a shadowing technique should be easier. Much factors decide the usability of a certain technique and an individual trade-off between the vantages and disadvantages has to be made. The overview maybe also inspire for new approaches.

References

AILA, T., AND AKENINE-MOELLER, T. 2004. A hierarchical shadow volume algorithm. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, ACM Press, New York, NY, USA, 15–23.

ASSARSSON, U., AND AKENINE-MOELLER, T. 2003. A geometry-based soft shadow volume algorithm using graphics hardware. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM Press, New York, NY, USA, 511–520.

BRABEC, S., AND SEIDEL, H., 2003. Shadow volumes on programmable graphics hardware.

BRABEC, S., ANNEN, T., AND SEIDEL, H.-P. Shadow mapping for hemispherical and omnidirectional light sources.

BRABEC, S. 2003. *Shadow Techniques for Interactive and Real-Time Applications*. PhD thesis, MPI Informatik.

BRENNAN, C. 2002. Shadow volume extrusion using a vertex shader. 188–194.

CASS EVERITT, ASHU REGE, C. C. 2004. Hardware shadow mapping.

CHAN, E., AND DURAND, F. 2004. An efficient hybrid shadow rendering algorithm. In *Proceedings of the Eurographics Symposium on Rendering*, Eurographics Association, 185–195.

CHONG, H., AND GORTLER, S. J. 2004. A lixel for every pixel. In *Proceedings of Eurographics Symposium on Rendering 2004*, Eurographics Association.

CHONG, H. Y.-I., 2003. Real-time perspective optimal shadow maps, April.

CROW, F. C. 1977. Shadow algorithms for computer graphics. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 242–248.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 161–165.

Shadowvolume sample. DirectX SDK (December 2006).

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

FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. 2001. Adaptive shadow maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 387–390.

GIEGL, M., AND WIMMER, M. 2007. Queried virtual shadow maps. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 65–72.

HEIDMANN, T. 1991. Real shadows, real time. *Iris Universe 18*, 28–31. Silicon Graphics, Inc.

HEMPE, N. 2005. *Robuste Echtzeitschatten fur komplexe, dynamische Szenen*. Master's thesis, University of Koblenz-Landau.

HORNUS, S., HOBEROCK, J., LEFEBVRE, S., AND HART, J. C. 2005. Zp+: correct z-pass stencil shadows. In *ACM Symposium on Interactive 3D Graphics and Games*, ACM Press, ACM.

IVICA KOLIC, ELJKA MIHAJLOVIC, L. B. 2004. Stencil shadow volumes for complex and deformable objects. 314–317.

KING, G. Shadow mapping algorithms.

KIRSCH, F., AND DOELLNER, J. 2003. Real-time soft shadows using a single light sample. V. Skala, Ed., vol. 11, 255–262.

KOZLOV, S. 2004. *Perspective Shadow Maps: Care and Feeding*. Addison Wesley, 217–244.

LAINE, S. 2005. Split-plane shadow volumes. In *Proceedings of Graphics Hardware 2005*, Eurographics Association, 23–32.

LEFOHN, A., SENGUPTA, S., KNISS, J., STRZODKA, R., AND OWENS, J. D. 2005. Dynamic adaptive shadow maps on graphics hardware. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications*.

LENGYEL, E., 2002. The mechanics of robust stencil shadows, Oktober.

LLOYD, B., WENDT, J., GOVINDARAJU, N., AND MANOCHA, D., 2004. Cc shadow volumes.

LLOYD, B., GOVINDARAJU, N. K., TUFT, D., MOLNAR, S., AND MANOCHA, D. 2006. Practical logarithmic shadow maps. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, ACM Press, New York, NY, USA, 103.

LLOYD, B., TUFT, D., YOON, S., AND MANOCHA, D. 2006. Warping and partitioning for low error shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2006*, Eurographics Association, 215–226.

MARTIN, T., AND TAN, T.-S. 2004. Anti-aliasing and continuity with trapezoidal shadow maps. In *Proceedings of the 2nd EG Symposium on Rendering*, Eurographics Association, Springer Computer Science, Eurographics.

MCCOOL, M. D. 2000. Shadow volume reconstruction from depth maps. *ACM Transactions on Graphics 19*, 1, 1–26.

MCGUIRE, M., HUGHES, J. F., EGAN, K., KILGARD, M., AND EVERITT, C. 2003. Fast, practical and robust shadows. Tech. rep., NVIDIA Corporation, Austin, TX, Nov.

2007. Phong reflection model, March. `http://en.wikipedia.org/wiki/Phong_reflection_model`.

2007. Projective texture mapping, March. `http://en.wikipedia.org/wiki/Projective_texture-mapping`.

ROETTGER, S., IRION, A., AND ERTL, T., 2002. Shadow volumes revisited.

SCHERZER, D. 2005. *Shadow Mapping of Large Environments*. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

2007. Shadow, March. `http://en.wikipedia.org/wiki/Shadow`.

2007. Shadow comparison, March. `http://www.acm.org/tog/editors/erich/shadow_comparison.html`.

2007. Shadow mapping, March. `http://en.wikipedia.org/wiki/Shadow_mapping`.

2007. Shadow volumes, March. `http://en.wikipedia.org/wiki/Shadow_Volumes`.

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

2007. Shadow volumes, March. `http://en.wikipedia.org/wiki/Stencil_buffer`.

STEINER, C. 2006. *Shadow Volumes in Complex Scenes*. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

TADAMURA, K., QIN, X., JIAO, G., AND NAKAMAE, E. 1999. Rendering optimal solar shadows using plural sunlight depth buffers. In *CGI '99: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 166.

VAN WAVEREN, J. 2005. Shadow volume construction. Id Software, Inc.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 270–274.

WIMMER, M., SCHERZER, D., AND PURGATHOFER, W. 2004. Light space perspective shadow maps. In *Rendering Techniques 2004 (Proceedings of the Eurographics Symposium on Rendering 2004)*, Eurographics Association, A. Keller and H. W. Jensen, Eds., Eurographics, 143–151.

ZHANG, F., SUN, H., XU, L., AND LUN, L. K. 2006. Parallel-split shadow maps for large-scale virtual environments. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, ACM Press, New York, NY, USA, 311–318.