Real-time Shadows in Computergraphics

Matthias Buchetics*
Vienna University of Technology

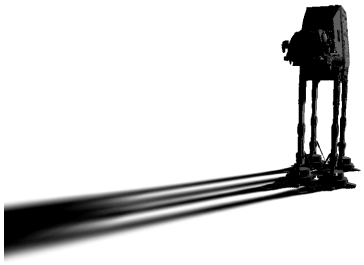

Figure 1: Shadows can greatly enhance computer generated images.

Abstract

Recent advances in 3D GPU technology have led to an increased in realistic graphic effects, like shadows. Extensive research has taken place in that field and the advances in the last years have been considerable. Real-time shadows are already considered indispensable in a range of applications and further improvements such as soft shadow generation continue to be a challenging research topic. The goal of this paper is to give an extensive overview of existing technologies. The most common problems are explained and solutions provided. Furthermore soft shadow techniques are described leaving the reader with enough knowledge to choose the best method for his or her needs.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture; I.3.3 [Picture/Image Generation]: Bitmap and framebuffer operations; I.3.1 [Hardware Architecture]: Graphics processors

Keywords: shadow algorithms, shadow volumes, shadow mapping, soft shadows, real-time

1 Introduction

Shadows are crucial for realistic computer generated images. They provide important visual cues to understand the geometry, position and size of the shadow occluder as well as information about the shadow receiver geometry. Tremendous advances in 3D graphics hardware technology led to a high interest in real-time shadow rendering. As a result, extensive research has been performed in the field in recent years. However, producing realistic shadows, especially soft shadows, in a real-time environment continues to be a difficult and challenging task.

Figure 2: Shadows help one to perceive geometric relations like the relative position of objects. Image courtesy of Hasenfratz et al.

Although all shadows in nature are soft and blend in with the environment, the most used form of computer generated shadows are currently hard shadows. Using this approach a point is either inside or outside the shadow area which can lead to serious aliasing artifacts easily noticed by the human eye. Soft shadows do not simply divide the region in "shadow" and "not shadow", they provide different intensities of shadows and a more realistic and visually pleasing end result.

The first comprehensive survey of shadows in computer graphics was written by Woo [Woo et al. 1990] but since then major advances in computer graphics technology have occured. A survey of real-time soft shadows [Hasenfratz et al. 2003] was done in 2003, followed by a state of the art report [Scherzer 2004]. Recent work not only deals with generating soft shadows but also with the various artifacts original real-time shadow techniques produce.

This paper first takes a look at those basic techniques, explains the problems of the different algorithms as well as solutions before giving a short overview of recent advances in the field of soft shadows. It is structured as follows:

In Section 2basic notions of shadow generation, hard and soft shadows and classic techniques to generate hard shadows are re-

*e-mail: mbuchetics@gmail.com

viewed. Section 3 will provide explanations of the most common aliasing artifacts using shadow mapping. Methods to resolve these issues are explained in Section 4. We will then look at soft shadow generation, emerging problems and outline popular algorithms in Section 5 before a summary and outlook is given in Section 6.

2 Basic Concepts

Imagine a scene with a *light source L*. All objects which are possibly illuminated by the light are called *receivers*. The *umbra* of the light source is the region of the scene that is not lit by L (it can be lit by other light sources though). If L is not just a single point but an area which emits light, sections of the scene can may be partly lit. This section is called the *penumbra* of the light source. The combination of umbra and penumbra is the shadow. An object blocking the light is called *occluder*. Any occluder can also be a receiver. In fact an object can be a receiver and occluder of the same light source which is a special case of shadow called "self shadow".

2.1 Hard Shadows

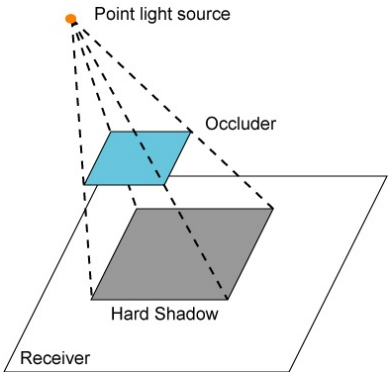

Figure 3: The umbra of a hard shadow. Image courtesy of Hasenfratz et al.

Most people see a shadow as a binary decision where a point can either be *in the shadow* or *outside*. This assumption is only true for point light sources. This single point is either totally visible for a given point in the scene or it is completely occluded. Point light sources do not exist in reality and while they simplify the shadow model and therefore the algorithms for calculation they give the image a rather unrealistic look. Figure 3 shows the geometry of a hard shadow. Hard shadows only have an umbra and no penumbra.

2.2 Soft Shadows

When considering that the light source is not just a point but an area or a volume which emits light, it is possible that only a fraction of the light source is visible from a given point in the scene. A partly hidden point lies within the penumbra while a completely hidden point is in the umbra of the light source. Looking at 4 one can see that the geometry of the umbra and penumbra is not only dependent on the light source and occluder, but also on the distance between the two. A soft shadow can not be mistaken as just a blurred version of a hard shadow because the degree of blurriness of a soft shadow varies with the distances involved between light source, shadow occluder and receiver. See Figure 2 for an example where parts of the shadow are more blurred than others.

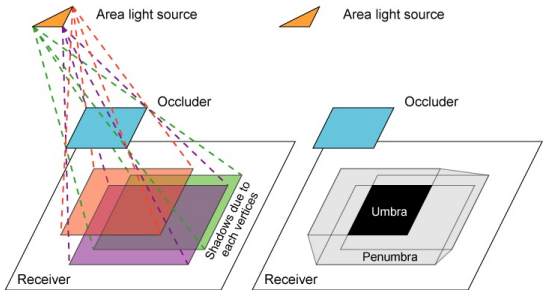

Figure 4: Umbra and penumbra of a area light source. Image courtesy of Hasenfratz et al.

2.3 Basic techniques

Considering that we are focusing on real-time techniques where the applications need to run at 30 fps or more, non real-time methods like ray tracing or radiosity will not be described in this paper. The two main real-time approaches to shadowing are shadow volumes and shadow mapping, both are going to be described in the next sections.

2.4 Shadow volumes

Shadow volumes were first described by Crow [Crow 1977]. In contrast to the shadow mapping method shadow volumes is a object-space technique. The algorithm build shadow volumes by first finding the silhouettes of the occluder before extruding them in direction of the light to infinity. Every point inside the shadow volumes is inside the umbra while all other points are illuminated by the light.

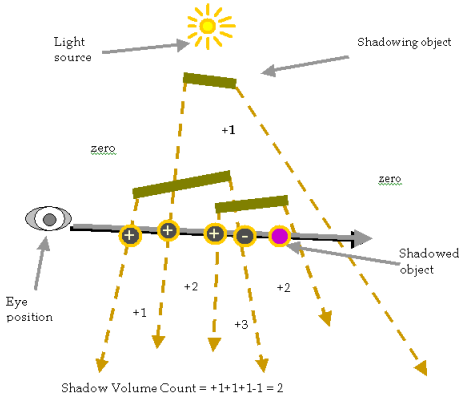

Figure 5: Shadow volume test.

An inside-outside test is used to determine if a point of the scene is inside the shadow volume or not. The number of faces of the shadow volume crossed is calculated for every rendered pixel. Front-facing faces increment the count and back-facing faces decrement it (see 5). The pixel is inside the shadow volume if the total count is positive whereas a zero or negative count will result in a illuminated pixel. This step of the algorithm can easily be done

in hardware using the stencil buffer, as a result most time is spent extracting the silhouettes and generating the shadow volumes. The cost of the algorithm is directly linked to the number of edges of the shadow volume resulting in a performance penalty on scenes with a high polygon count. Several publications exist trying to improve the shadow volume algorithm in various ways. Brabec [Brabec and Seidel 2003] developed a way to generate the shadow volume on the hardware resulting in a considerable performance gain compared to previous CPU based methods. Everitt and Kilgard [Everitt and Kilgard 2003] have described a robus shadow volume implementation using the zfail technique which was discovered independently by Carmack and Bilodeau/Songy.

Shadow volumes genereally do not have any aliasing problems because they provide pixel perfect shadows in eye-space which is also the big advantage of this method. They also handle self shadowing without any extra steps. On the other side there is the need for vast amounts of fillrate, depending on the complexity of the scene. Even though approaches of soft shadow volumes exist [Assarsson and Akenine-Möller 2003] the technique usually suffers from the very hard shadow edges leading to a unrealistic look.

2.5 Shadow mapping

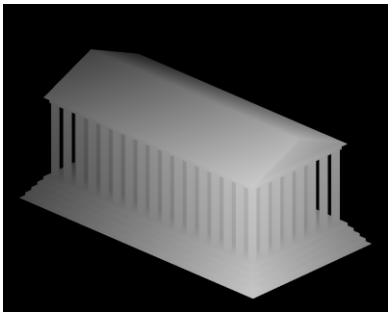

Figure 6: Shadow map rendered from the light view.

Shadow mapping was first introduced by Williams in [Williams 1978] and has been the most popular real-time shadowing technique in recent years. The two-pass technique starts by rendering the scene from the point of view of the light source (see Figure 6). Using z-buffering the z-values of the nearest objects are stored and the result is the *shadow map*. In the second pass the scene is rendered from the eye position. Each pixel is transformed to the light-space where the depth can be compared to the depth previously stored in the shadow map. If the depth value of the shadow map is closer to the light the pixel lies within the shadow. All other pixels are illuminated since they are closer to the light source than any possible occluder. Figure 7 shows the shadow map projected on the scene, the resulting image can be seen in figure 8.

Many advantages made shadow mapping popular. First and foremost, it can be completely implemented in modern computer graphics hardware. The algorithm itself is independent of object geometry making it suitable for very complex scenes. Furthermore the basic algorithm, also called *uniform shadow mapping*, is easy to implement. Shadow mapping is an image-space technique and consequently suffers from aliasing artifacts. We are going to look at these artifacts in the next section.

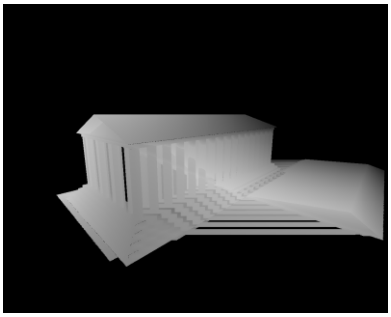

Figure 7: Visualization of the shadow map projected onto the scene.

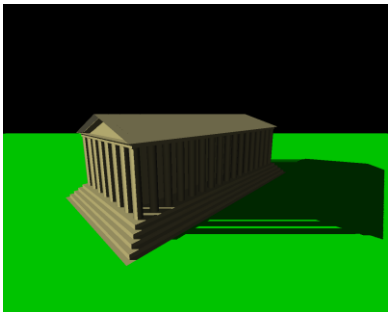

Figure 8: The resulting image.

3 Shadow map aliasing

Like most other image-based techniques shadow mapping suffers from various aliasing artifacts caused by the discretizations of the shadow and image buffer. Additionally the depth values of the shadow map are quantized [Williams 1978]. While the aliasing errors are worst in large scenes where the shadow map represents a high depth range, they generally always occur when the local sampling density is too low. These artifacts can be classified into *perspective aliasing* and *projection aliasing*.

3.1 Limited depth buffer precision

Artifacts called *surface moire* or *incorrect self shadowing* are the result of limited precision of the depth buffer [Williams 1978]. During shadow map generation, depth values are quantized, which can result in incorrect classifications at the later depth test, where the stored values are compared to the depth values of the current pixel. Everitt and Kilgard [Everitt et al. 2002]pointed out that incorrect classification can also happen if infinite precision of the depth buffer is assumed. Because both the shadow map buffer, as well as the image buffer, are discretized and sampled at regular intervals the intersection points on the surface may differ slightly for the shadow map generation pass and later depth test.

3.2 Perspective and projection aliasing

Stamminger and Drettakis formalized the aliasing problems as follows:

d_s describes the size $d_s * d_s$ of a shadow map pixel, which is represented by a sheared pyramid of rays passing through the shadow map from the light reference point. $1/d_s$ can then be called the *shadow map resolution*. The smaller the size of the shadow map

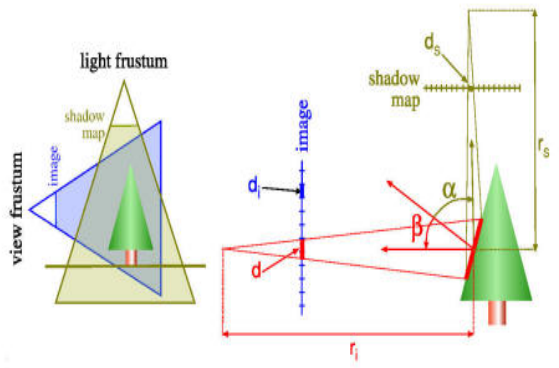

Figure 9: Formalization of shadow map aliasing. Image courtesy of Stamminger and Drettakis.

pixels, the larger is the actual shadow map resolution. The intersection size, when a shadow map pixel with the size of d_s hits the surface, can be approximated with $d_s r_s / \cos\alpha$ where α is the angle between the light ray and the surface normal. The intersection size in the image buffer can be calculated with Equation 1.

$$d = d_s \frac{r_s}{r_i} \frac{\cos\beta}{\cos\alpha} \tag{1}$$

Undersampling Shadow mapping undersampling occurs if the size of intersection d is larger than the pixel size d_i on the image plane. In that case one shadow map pixel maps to multiple image plane pixels resulting in unsmooth edges of the shadow.

Undersampling, by far the biggest aliasing problem of the shadow mapping technique, can have two different causes. The first is called *perspective aliasing* and happens when the user zooms in close to a shadow ($d_s r_s / r_i$ becomes large). Due to the limited size of the shadow map only a certain precision can be reached and multiple shadow map pixels become clearly visible at a given distance to the shadow receiver. *Projection aliasing* is the second cause of undersampling and appears whenever the light rays become nearly parallel to the surface of the shadow receiver (the term $\cos\alpha\cos\beta$ becomes large).

Oversampling The opposite effect when d is smaller than the pixel size d_i resulting in more than one shadow map pixels being mapped to a single image plane pixel is called oversampling. Oversampling can often be noticed as flickering edges since the influencing shadow map pixel may change from frame to frame.

In the perfect case of d being equal to d_s (one shadow map pixel maps to exactly one image plane pixel) neither undersampling nor oversampling will occur. Due to the nature of most complex scenes this is almost never the case for all points of a scene.

4 Artifact reduction

The above described artifacts are serious disadvantages of the shadow mapping technique. Though many methods reducing these problems have been developed all of them have their drawbacks and the area continues to be an interesting research topic. The following section will give an overview of some of the recently proposed solutions.

4.1 Undersampling

Undersampling is the result of an undersized shadow map when one pixel of the shadow map maps to multiple image buffer pixels. The easiest method to reduce undersampling is to increase the resolution of the shadow map. As explained above this will result in smaller shadow map pixel sizes.

A global resolution increase may eliminate artifacts in some scenes, but will not be sufficient for most, especially larger scenes. For those, extremely large shadow maps would be necessary, which highly affects the fillrate as well as memory consumption and the overall performance. Additionally, the buffer sizes of current hardware are limited making huge shadow maps unfeasible. With these limitations in mind a global resolution increase can only reduce the artifacts to a certain amount creating the need for more intelligent approaches. The common aspect of these approaches is that they attempt to make optimal use of the available resolution by increasing it locally for parts of the scene where more detail is needed.

Most approaches fall in one of the following categories: *warping algorithms* and *partitioning algorithms*. Whereas warping algorithms reparameterize the 4x4 matrix, the shadow map is rendered with partitioning algorithms that split the scene in different partitions (hence the name) and use separate shadow maps for each one.

4.1.1 Warping algorithms

Shadow map warping was introduced by Stamminger and Drettakis in their work about **perspective shadow maps** [Stamminger and Drettakis 2002]. The purpose of perspective shadow maps is to use post-perspective space instead of the world space for the shadow map generation. Both the scene and the light source are first transformed to the post-perspective space and are then rendered from the viewpoint of the light with depth buffer enabled. The result is stored in the perspective shadow map. Later the scene is rendered from the camera viewpoint similar to the traditional uniform shadow map algorithm. Perspective aliasing is decreased because the shadow map is looking at the scene after perspective projection and objects closer to the viewer will be drawn larger. Therefore more pixels in the shadow map are used for close objects (see Figure 10).

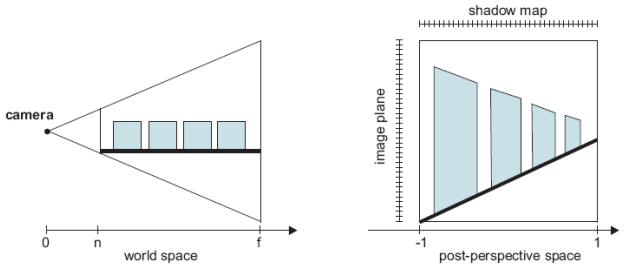

Figure 10: The scene as it is viewed in post-perspective space.

It is important to know that the perspective transformation can change the type of light sources from directional to point-light or vice versa. While a directional light can be seen as a point-light at infinity the perspective mapping may change the position from infinity to finite positions. Stamminger and Drettakis discuss six different cases with different lights and types.

Another difficulty with perspective shadow maps is to ensure that all potential shadow casters are included in the shadow map and

must not be clipped before. The original article suggests to move the camera backwards until all shadow casters are included in the frustum of the camera. Using intersection calculations of the scene bounding box a convex hull of all objects can be created in order to update the camera position. However, the resulting frustum and therefore also the shadow map will cover a larger area and higher resolutions are necessary to avoid artifacts.

As Martin and Tan [Martin and Tan 2004] as well as Wimmer et al. [Wimmer et al. 2004] point out, perspective shadow maps come with a number of difficulties and drawbacks. These include:

- The implementation is nontrivial due to many special cases.
- The bias problem is worsened using the post-perspective space.
- The camera adjustment to include all shadow caster reduces the shadow map quality significantly.
- PSMs do not perform well for distant objects and most pixels are used for closer objects.

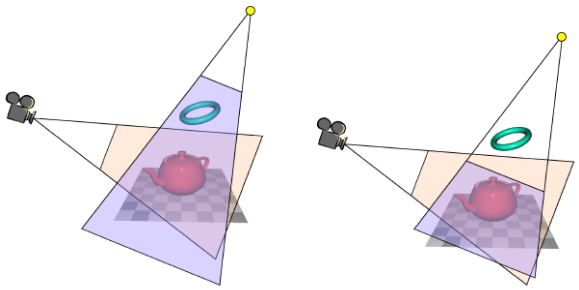

Figure 11: Left: Standard near/far setup. Right: Tight fitting near/far setup. Image courtesy of Brabec et al.

Practical shadow mapping as proposed by Brabec et al. [Brabec et al. 2002] is based around the idea to analyze the scene and set the light's view frustum accordingly in order to improve depth precision. But as one can see in Figure 11 simply adjusting the near and far plane is not sufficient, since shadow casters may be outside the new light frustum (in this case the torus which is not seen by the camera but still casts a large shadow on the scene). If the scene on the right side of Figure 11 would be rendered using a traditional shadow map approach the torus would be clipped away. Instead of clipping depth values outside the valid range, practical shadow mapping clamps the values. To avoid culling of objects which are completely in front of the near plane, a vertex shader is used to set the z component of the output position to $0.5 * w$. As a result all vertices are then inside the valid [0;1] z-range and do not get culled away. The clamping of depth values can be done using a pixel shader.

In addition to adjusting the near and far plane the remaining four sides of the light's view frustum are important for the shadow map resolution. Practical shadow mapping uses a bounding rectangle (the fastest and easiest way is to use an axis aligned bounding rectangle) that encloses all relevant pixels of the scene. In order to compute those pixels the scene is first rendered from the camera viewpoint and projective texturing is used to map some sort of control texture on the scene. The texture is projected from the light position and later read back from the frame buffer to analyze which regions of the shadow map are used. The resulting bounding rectangle can then be used to focus the shadow map on the relevant pixels in the scene enhancing the shadow map resolution.

Trapezoidal shadow maps is another warping technique and was proposed by Martin and Tan [Martin and Tan 2004]. This approach uses a trapezoid to approximate the view frustum, which is done after the view frustum has been transformed to post-perspective light space. This can be done very efficiently because only the eight corners of the view frustum plus the centers of the near and far plane have to be transformed and the algorithm scales well for large scenes. Using the trapezoid, a reparametrization matrix can be calculated which is then used to generate the shadow map. Because scene information is not used in the process to compute the reparametrization matrix this approach can not focus on objects smaller than the view frustum. The approach furthermore decreases the continuity problem significantly where shadows flicker from frame to frame.

Compared to uniform shadow maps TSMs provide the best results if the scene is relatively large with a small eye view frustum (as seen from the light source). Furthermore, the light source is best located perpendicular to the scene and the camera is close to the ground.

Improvements will not be visible if 1) the scene (again as seen from the light source) is small compared to the eye frustum, 2) the camera is facing the light direction (or the opposite light direction) or 3) the camera is not located on the ground of the scene but is looking down from a bird's eye view.

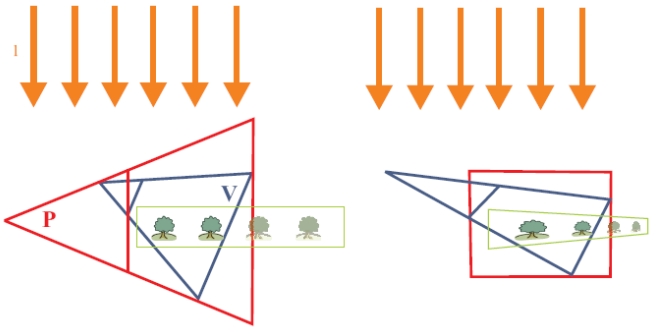

Figure 12: The same scene before (left) and after the application (right) of the LiSPSM transformation. The light direction stays the same. Image courtesy of Wimmer et al.

Light space perspective shadow maps is another approach which extends on the idea of Stamminger et al. It is based on the observation that there is no need to tie the perspective transformation to the view frustum as in PSM. Any arbitrary perspective transformation can be used and it is also sufficient to use a warp that affects the shadow map plane [Wimmer et al. 2004]. The perspective transformation in LiPSM is specified in regard to the coordinate axis of the light and it does not change the direction of the light (in contrast to PSM). This avoids some of the problems PSM has and is also more intuitive therefore easier to implement.

The LiPSM technique works as follows: First a convex body which usually includes the view frustum and all possible shadow

casters is calculated. The shadow map is then focused on this body exactly the same way as is done with PSMs. Different from PSM, a special transformation which encloses the calculated body with an perspective frustum is now constructed instead of using the projection of the viewplane. The parameters for this transformation are found using the light space which is defined by the light source direction, the shadow plane and the view direction. The authors note that there are no singularities in the combined perspective mapping and point lights can be treated as directional lights after the perspective transform. The near and far planes are placed at the minimum and maximum light space z-coordinates of the convex body parallel to the xy-coordinate plane. After calculating the x and y coordinates of the projection reference point there is one free parameter left, which can be used to control how strong the shadow map will be warped. A value close to the near plane will result into a strong distortion similar to the original perspective shadow maps. On the other side the parameter can be chosen to minimize the perspective distortion, resembling uniform shadow maps. Wimmer et al. show that the optimal choice in case of a view direction perpendicular to the light vector is $n_{opt} = z_n + \sqrt{z_f z_n}$ (z_n = near plane distance of the eye view frustum, z_f = far plane distance).

If the perspective frustum has been found it is combined with the standard projective mapping and applied to the standard shadow map generation and rendering process.

Llyod et al. [Lloyd et al. 2006] point out that one disadvantage warping algorithms have is that the shadow map alignment depends on the view and light may lead which to "crawling" edges in animated scenes.

4.1.2 Partitioning algorithms

The idea of **adaptive shadow maps** was first brought up by Fernado et al. [Fernando et al. 2001]. Basically, the traditional shadow map is subdivided hierarchically to provide higher resolutions in visually important regions. Only regions containing shadow boundaries need to be sampled densely and to avoid aliasing artifacts the resolution in these areas should be at least as high as the corresponding region in the eye view. The hierarchy of adaptive shadow maps, which needs to be updated whenever the viewpoint of the camera changes, is organized in the form of a tree structure. Each node in the tree has a shadow map and a partitioning into a fixed number of cells assigned to it. Each cell may contain another node. The tree is updated continuously, assigning new nodes to empty cells if the resolution is not high enough or deleting nodes if they are not needed or the memory restrictions are reached.

In order to calculate the projected area of a pixel which is required to determine whether the resolution is sufficient or not, the adaptive shadow map approach uses mip-mapping and read-backs from the graphics hardware. As Arvo [Arvo 2004] points out, this can be a performance bottleneck since off-the-shelf graphics hardware does not usually support very fast read-backs which would be required for this technique. Furthermore many rendering passes are required for the traversal and refinement operations.

Arvo [Arvo 2004] presented a **tiled shadow map algorithm** which can be seen as a simplified variant of adaptive shadow maps. The light view is partitioned in several adjacent regularly sized rectangles (the *tiles*) in form of a *tile grid*. While in uniform shadow mapping all tiles would cover the same region, tile weights which determine how much shadow map area each tile should allocate are assigned to each tile. Using these weights the shadow map is divided and each tile is rendered separately into the shadow map.

The tile weights are calculated in an additional low-resolution render pass called *light view analysis*. The weights are dependent on three variables: whether pixels are on the shadow boundary, the distance from the shadow caster to the receiver and the distance from the viewpoint to the receiver. The calculated weight is proportional to the sampling rate of the shadow map. A ten times larger tile weight will therefore result in ten times more sampled shadow map pixels.

After calculating the weights the shadow map is divided accordingly. Like adaptive shadow maps this requires a read-back operation from the graphics hardware, but since only one value per tile and not per pixel has to be read-back (the pixel values are summed on the GPU) the amount is considerably smaller. The division of the shadow map is done recursively using a binary cut algorithm where the cutting direction (horizontally or vertically) is alternated. The advantage of this technique is, that no pixel is wasted and the whole shadow map resolution is used. However, new artifacts may be introduced by the subdivision process if the assigned areas on the shadow map are extremely non-square.

The light viewport is then adjusted for each tile and the depth values are rendered into the assigned shadow map area. The shadow map lookup is almost identical to uniform shadow mapping requiring one additional dependent texture access.

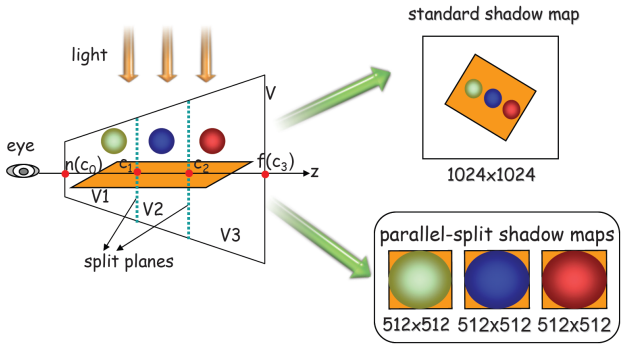

Figure 13: The idea behind the PSSM algorithm: splitting the view frustum into several parts and using an independent shadow map for each part. Image courtesy of Zhang et al.

Zhang et al. [Zhang et al. 2006] developed the **parallel-split shadow maps** (PSSMs) technique where the view frustum is split into different depth layers by using parallel split planes. For each split part an independent shadow map is rendered leading to different sampling densities at different positions in the view frustum. Figure 13 shows an example of this approach. Using independent shadows maps different parameterizations can be applied for different split parts. Furthermore, the used texture memory for all independent and smaller shadow maps is usually less than the memory used by a single large shadow map. Also, the worst case of shadow mapping, when the view and light directions are nearly opposite, is handled well because each shadow map is focused in smaller subfrusta [Zhang et al. 2006].

Using parallel-split shadow maps the view frustum is first split into several parts which is based on the idea that different depth layers need different shadow map resolutions to avoid artifacts. Figure

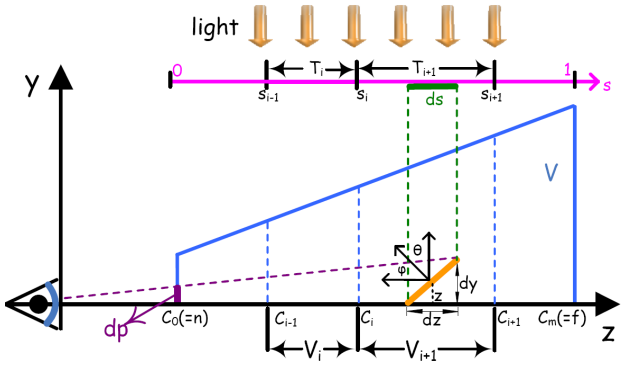

Figure 14: The frustum is split along the z axis in parts at split positions C_i. Image courtesy of Zhang et al.

14 shows an example of a view frustum which is split into parts at certain positions C_i. The PSSM approach describes three different schemes to select the split positions: uniform split scheme, logarithmic split scheme and practical split scheme which combines the first two (see Figure 15).

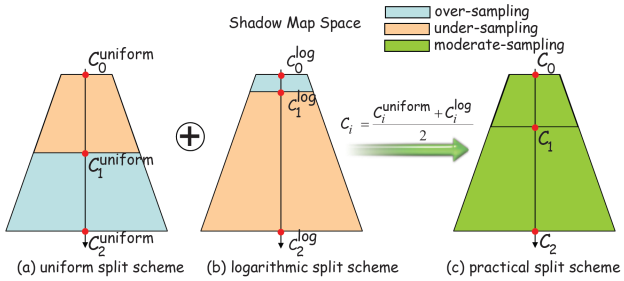

Figure 15: The three different split schemes. Image courtesy of Zhang et al.

As we have shown above, the optimal distribution of perspective aliasing errors make dp/ds constant and the logarithmic function is used to approximate this distribution. The more split parts that are used in order to discretize the logarithmic function, the closer the distribution will to the ideal. The problem with the *logarithmic split scheme* is, that the split parts close to the viewer are too small and only a few objects will be included. Zhang et al. describe that "this is due to the theoretically optimal parameterization assumes that the shadow map accurately covers the view frustum and no any resolution is wasted on invisible parts of the scene." In practice this leads to over-sampling in parts near to the viewer and under-sampling in parts further from the viewer.

The *uniform split scheme* simply splits the frustum uniformly along the z axis. Similar to uniform shadow mapping this scheme results in under-sampling at points closer to the viewer because the distribution of perspective aliasing errors is the same. Furthermore over-sampling is introduced in points further away.

Both, logarithmic and uniform split schemes, do not produce appropriate sampling densities for the whole scene. The *practical split scheme* is designed to combine the advantages of both while reducing their draw backs and aliasing origins. The calculated split positions $C_i^{uniform}$ and $C_i^{logarithmic}$ are simply combined by $C_i = (C_i^{uniform} + C_i^{logarithmic})/2$. See equation 2 for the complete computation.

$$C_i = \frac{n(f/n)^{i}/m + n + (f-n)i/m}{2} + \delta_{bias} \tag{2}$$

notation	description
C_i	depth of the i-th split plane
n	near plane
f	far plane
m	number of splits
i	current split

Besides splitting the view frustum as explained above the light frustum W is also split into smaller parts W_i. Each of the light frustum parts covers one view frustum part $V_i$and the bounding boxes of these view frustum split parts are used to focus the light frustum parts on the relevant areas. Afterwards each split part V_i is rendered to an independent shadow map T_i using the light space W_i. Shadow map sizes of 512x512 are usually sufficient and other shadow mapping techniques such as the before described warping approaches can be integrated into PSSMs. Generally three splits already produce good results and use less memory than one uniform 1024x1024 map. Since additional splits result in additional render passes the performance decreases with an increasing number of splits. Zhang et al. eliminated this problem by integrating the shadow map selection into the pixel shader which is possible as long as the number of splits does not excess the number of available textures.

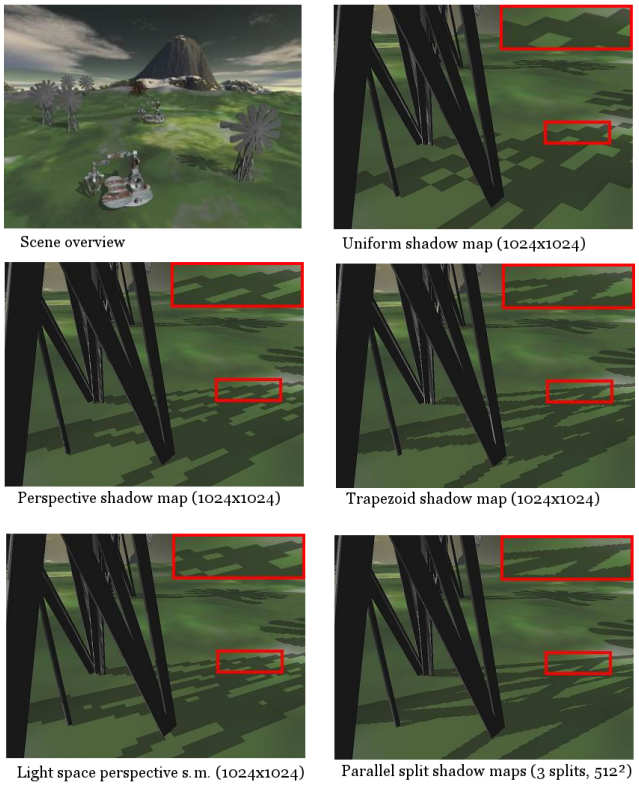

Figure 16: Six shadow map algorithms compared. Images courtesy of Zhang et al.

Experiments have shown that parallel split shadow maps work extremely well for very large environments. The example in Figure

16 is a outdoor environment rendered in the view frustum with near = 1m and far = 1000m. There are about 55 complex objects randomly located in the $2.56km^2$ environment. The authors state that rendering performance of the 1.3 million polygons was almost the same for all methods used.

4.1.3 Other approaches

Donnelly and Lauritzen [Donnelly and Lauritzen 2006] introduced *variance shadow maps* to filter shadow maps. As Reeves et al. [Reeves et al. 1987] pointed out it is not possible to use ordinary bilinear or trilinear filtering for that manner because "the filtered depth value would be compared to the depth of the surface being rendered to determine whether or not the surface is in shadow at that point." Therefore the built-in methods of modern graphics hardware, mipmapping and anisotropic filtering, are inapplicable because they would interpolate the depth values of neighboring pixels resulting in a possible false classification of pixels. A technique called percentage closer filtering, usually also used to reduce oversampling, was proposed by Reeves et al. [Reeves et al. 1987] (see the next section for more details). While in a standard shadow map each texel only represent the depth of one point variance shadow maps represent a distribution of depths at each texel. The distribution of depths is approximated by storing the mean and mean squared values (first and second distribution moments). The average of two distributions can then be easily approximated by averaging those two moments. The moments can be used "'to compute a bound on the fraction of the distribution that is more distant than the surface being shaded."' which provides a good approximation of light reaching a surface [Donnelly and Lauritzen 2006]. Standard built-in filtering techniques of graphics hardware can be used to interpolate the moments effectively. The performance is comparable to ordinary shadow maps but provide much more pleasing results (see figure 17). Another advantage is that this technique can easily be combined with existing warping algorithms to further enhance the results.

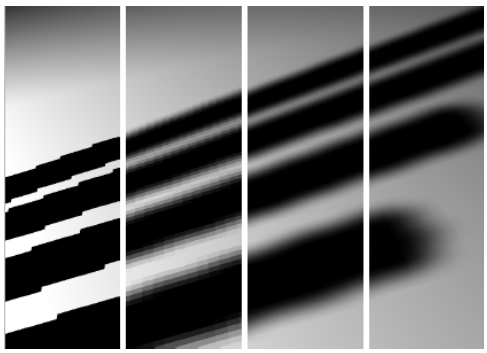

Figure 17: Left to right: 1) standard shadow mapping, 2) 5x5 percentage closer filtering, 3) 5x5 bilinear percentage closer filtering, 4) variance shadow maps with 5x5 separable gaussian blur. Image courtesy of Donnelly and Lauritzen.

Lloyd et. al. evaluated several algorithms comparing the maximum perspective aliasing error over the entire view frustum [Lloyd et al. 2006]. Ignoring the projection aliasing developed an error metric and applied it to current warping and partitioning algorithms. Furthermore, a combination of warping and partitioning algorithms is proposed which delivers a rather low aliasing error with a small number of shadow maps.

4.2 Oversampling

When more than one shadow map pixel maps to a single image buffer pixel oversampling occurs. Reeves et al. [Reeves et al. 1987] proposed *percentage closer filtering* to eliminate oversampling artifacts by first testing all pixels separately to create a binary map where neighboring pixels are later averaged to create a filtered result. The result is a percentage between 0 and 100% which is then used to shadow the pixel accordingly. Percentage closer filtering also smoothes the edges of the shadow reducing some undersampling artifacts although other methods like the above described *variance shadow maps* by Donnelly and Lauritzen [Donnelly and Lauritzen 2006] give significantly better results.

4.3 Limited depth buffer precision

Both problems *surface moire* and *incorrect self shadowing* can be solved by adding a bias value before the depth test. The disadvantage is that an optimal value can often not be found [Everitt et al. 2002] and the choice is often a compromise. [Hourcade and Nicolas 1985] proposed a method called *second depth shadow mapping* where the depth value of the second surface seen from the light is stored instead of the first surface. The depth test is modified accordingly. He also discussed the use of a p-Buffer instead of a depth buffer where object-IDs are stored instead of depth values.

5 Soft Shadows

Similar to the basic concepts shadow maps (Section 2.5) and shadow volumes (Section 2.4) soft shadow algorithms can be divided into image-based and object-based approaches. Whereas image-based approaches extend on the shadow map technique, object-based approaches are built upon shadow volumes. Like before we are going to focus on the more popular shadow map methods.

Hasenfratz et al. [Hasenfratz et al. 2003] lists a number of image-based methods:

- Combination of multiple hard shadow calculations in order to approximate an area light source.
- Replacing the shadow map with a layered depth map storing multiple depth values for all objects visible from at least parts of the light.
- Using image analysis techniques on standard shadow maps.
- Using an image of the light source and convolute it with the shadow map.

Combining multiple hard shadows is simple and usually more physically accurate than some of the other methods. The drawback is that the number of render passes increases with the number of samples taken. Examples are given in [Scherzer 2004] and [Hasenfratz et al. 2003].

An interesting approach are **penumbra maps** as proposed by Wyman and Hansen [Wyman and Hansen 2003]. The penumbra map is used in addition to the standard shadow map and allows

polygonal objects to cast approximate soft shadows on themselves and other objects. In a three pass process a standard shadow map is created before the penumbra map is calculated in the second pass. The penumbras themselves are constructed using cones and sheets. The first pass then uses the intensity information from the penumbra map in combination with the shadow map's depth information to render the scene.

While penumbra maps can be quite easily integrated in existing shadow map algorithms, their performance suffers especially in scenes with a high polygon count. A significant amount of time is spent calculating the penumbra.

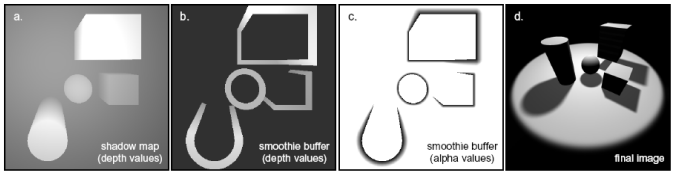

Figure 18: Left to right: a) standard shadow map, b) smoothie buffer (depth values), c) smoothie buffer (alpha values), d) finale image. Image courtesy of Chan and Durant.

Another variation of the shadow map approach are the so called **smoothies**. The algorithm first renders an ordinary shadow map and then extracts the silhouettes of the shadow occluders. Similar to penumbra maps the silhouettes are then extended by geometric primitives (the smoothies) which are later rendered into a smoothie buffer. In addition to the smoothie depth an alpha value which depends on the ratio of distances distances between the light source, blockers and receiver is stored. Finally the image is rendered by using the shadow map and smoothie buffer to determine whether and how a point is shadowed. See Figure 18 for an overview of the process.

Both, smoothies and penumbra maps only compute the outer penumbra, which makes scenes a lot darker than anticipated because shadow occluders will always project an umbra even if the light source is very large. One approach that handles inner and outer-penumbras is the **soft shadow map** technique [Atty et al. 2006].

The algorithm starts by dividing occluders and receivers, one object can not be occluder and receiver at the same time and the soft shadows are only generated from the occluders onto the receivers. Instead of one depth, buffer two buffers, one for the occluders and one for the receivers, are computed. The occluder depth buffers is converted into a set of rectangles or so called micro-patches. The size of these rectangles depends on the distance from light source to the pixel in the occluder buffer. The further away a pixel is the larger the resulting rectangle.

The depth values of the receivers are stored in the receiver buffer. In order to compute the soft shadow map for each light source, the soft shadow for every occluder micro-patch is calculated and summed. This calculation takes the relative distance between the occluders, the receiver and the light source into account. Given the fact that the micro-patches are parallel to the light source this computation can be done quite fast.

While the algorithm is fast and provides good results the largest limitation is the fact that self-shadowing is not supported. Also, the technique can not be easily integrated into existing hard shadow map methods.

This was only a short overview of some of the existing soft shadow mapping techniques. For more take a look at the surveys of Hasenfratz et al. [Hasenfratz et al. 2003] and Scherzer [Scherzer 2004].

6 Conclusion

While shadow mapping became very popular in recent years because of the simplicity of the basic algorithm, we have seen that various aliasing artifacts such as perspective and projection aliasing make new approaches and extensions of the original uniform shadow mapping technique necessary.

In order to understand the problems involved, the basic concepts of hard and soft shadows as well as the two most popular techniques, shadow volumes and shadow mapping were explained. Later we gave a detailed look into the possible aliasing artifacts and described some of the most important methods to reduce them. These methods were classified in regard to what aliasing artifact they are supposed to eliminate. Most research goes into the reduction of undersampling artifacts and all ideas are based on the idea to provide a higher shadow map resolution were needed and a lower sampling density for distant regions. Most proposed methods can be divided into warping and partitioning algorithms. Whereas the former modify the perspective transform in order to reach the goal, the latter divide the shadow map or the view frustum in smaller parts for specific regions of the scene. These parts can be handled individually and their size varies depending on the requested shadow map size.

At the end a short overview of existing soft shadow techniques was given, introducing even more challenges to the already complex area of real-time shadows. After studying this paper the reader should be able to understand the basic concepts and problems of real-time shadows. Choosing a shadowing technique is not a simple task since due to the nature of the problem there is no single ideal or optimal solution. Therefore, I hope that this paper can help a reader get started in this interesting field of computer graphics.

References

ARVO, J. 2004. Tiled shadow maps. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, IEEE Computer Society, Washington, DC, USA, 240–247.

ASSARSSON, U., AND AKENINE-MÖLLER, 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.

ATTY, L., HOLZSCHUCH, N., LAPIERRE, M., HASENFRATZ, J.-M., HANSEN, C., AND SILLION, F. 2006. Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum 25*, 4 (dec). (to appear).

BRABEC, S., AND SEIDEL, H.-P. 2003. Shadow volumes on programmable graphics hardware. In *Eurographics 2003*.

BRABEC, S., ANNEN, T., AND SEIDEL, H.-P. 2002. Practical shadow mapping. *J. Graph. Tools 7*, 4, 9–18.

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.

EVERITT, C., AND KILGARD, M. J., 2003. Practical and robust stenciled shadow volumes for hardware-accelerated rendering.

EVERITT, C., REGE, A., AND CEBENOYAN, C., 2002. Hardware shadow mapping.

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.

HASENFRATZ, J.-M., LAPIERRE, M., HOLZSCHUCH, N., AND SILLION, F. 2003. A survey of real-time soft shadows algorithms. *Computer Graphics Forum 22*, 4 (dec), 753–774.

HOURCADE, J. C., AND NICOLAS, A. 1985. Algorithms for antialiased cast shadows. *Computers and Graphics 9*, 3, 259–265.

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 *Rendering Techniques*, 153–160.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. *SIGGRAPH Comput. Graph. 21*, 4, 283–291.

SCHERZER, D. 2004. Real-time soft shadows. In *Europgrahics*.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps.

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.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A survey of shadow algorithms. *IEEE Comput. Graph. Appl. 10*, 6, 13–32.

WYMAN, C., AND HANSEN, C. 2003. Penumbra maps: approximate soft shadows in real-time. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 202–207.

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.