# Shadow Map Silhouette Revectorization (SMSR)

Vladimir Bondarev
NHTV Breda University of Applied Sciences

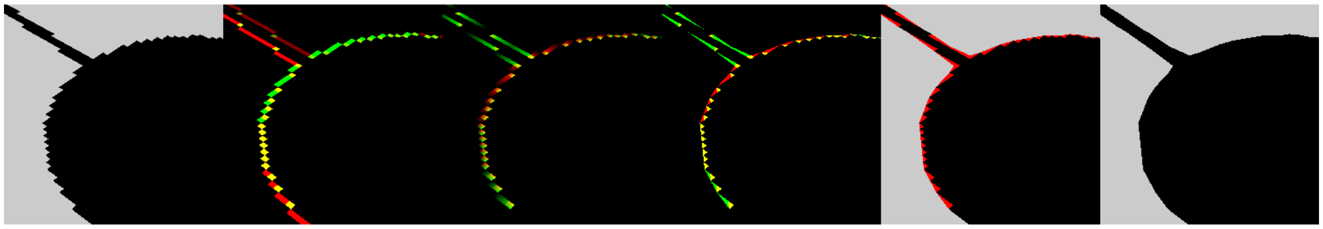


***Figure 1:*** *From left to right, revectorization process.*

Shadow Map Silhouette Revectorization is a two-pass filtering technique which aims to improve the visual quality of a shadow map by reapproximating a new shadow silhouette edge in undersampled areas. In most cases undersampled areas result in a higher shadow silhouette quality. However, the technique is relatively new and has some issues which require attention.

## 1. Introduction

Shadow mapping[1] is known for it's compatibility with rendering hardware, low implementation complexity and ability to handle any kind of geometry. However, aliasing is also a very common problem in shadow mapping. This paper introduces a new shadow map filtering technique which reapproximates new shadow silhouette edges.

Projection and perspective aliasing[6] are the two main discontinuity types which deteriorate projected shadow quality. Since the introduction of shadow mapping, many clever algorithms have been developed to reduce or even completely remove shadow map aliasing. Algorithms which are developed to remove aliasing completely, are unfortunately not compatible to run in real-time with current GPU architecture and in some cases propose additional hardware changes to allow for real-time application (LogPSM[6], Irregular Z-Buffer Technique[4]). Some algorithms which run in real-time are focused on optimal sample redistribution (PSM[9], TSM[7], LiSPSM[11], CSM[1]) and other use filtering techniques to achieve their objectives (VSM[2], PCF[8], BFSM[5]).

Shadow Map Silhouette Recevorization (SMSR) is a filtering technique which re-approximates shadow silhouette based on MLAA[3] implementation for GPU. SMSR consists of two main passes. First pass searches for discontinuity information. Second pass determines the discontinuity length, orientated normalized discontinuity space, fills new edge area and eventually merges lighting or image buffer with new edge information.

## 2. Implementation

SMSR consist of two passes the first pass finds the shadow silhouette edge and compresses the edge discontinuity into a 2-component output vector. Based on discontinuity, the second pass creates a coordinate space in which a new silhouette edge is interpolated.

### 2.1 First pass

In screen-space, we look for shadow discontinuity by offsetting the projected coordinate in light space by one shadow map sample in four directions (left, top, right, bottom). The discontinuity is distinguished into two main types. Exiting discontinuity, where the current fragment sample is inside the umbra (area completely occluded from direct light-source) and next neighboring sample is outside the umbra. Entering discontinuity, where the current fragment sample is outside the umbra and next neighboring sample is inside the umbra.

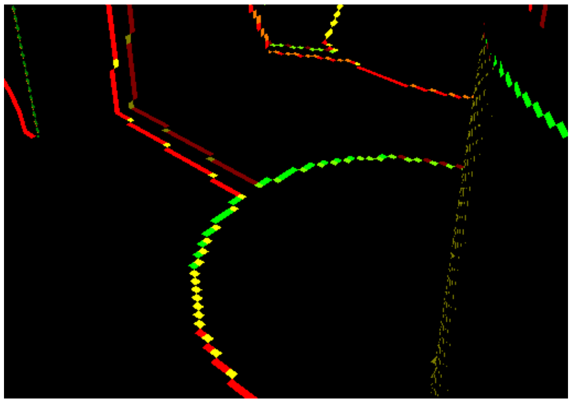


***Figure 2:*** *Compressed silhouette discontinuity.*

In the current SMSR implementation we are only concerned with the entering discontinuity of the shadow silhouette edge. When an entering discontinuity is detected, the direction from current fragment sample towards discontinuity is compressed into one of the output channels (used in second pass to determine discontinuity orientation). Horizontal discontinuities are stored into the red channel and vertical discontinuities are stored into the green channel. Each channel has four possible states, for example the red channel uses the value 0.0 to indicate no discontinuity, 0.5 discontinuity to the left, 0.75 discontinuity to the left and right, 1.0 to the right. The green channel uses the value 0.0 to indicate no discontinuity, 0.5 discontinuity to the bottom, 0.75 discontinuity to the bottom and top, 1.0 to the top.

| **Value/Channel** | **Red** | **Green** |
|---|---|---|
| 0 | No discontinuity | No discontinuity |
| 0.5 | Left | Bottom |
| 0.75 | Left and right | Bottom and top |
| 1 | Right | Top |

***Table 1:*** *Compression-value matrix.*

### 2.2 Second pass

The second is the most important and demanding pass. It determines the discontinuity length, orientation and fragment position which is translated into orientated normalized discontinuity (xy-coordinate) space. The normalized space is then used to perform a linear line interpolation which determines the new silhouette edge.

The main challenge is to correctly approximate discontinuity length from the screen-space in projected light-space. In order to approximate the discontinuity length, we have to find the new screen-space position of the neighboring projected shadow-map sample. The new screen-space position is determined by transforming the current fragment's world-space position into projected light-space, applying a xy-offset to the center of the new shadow map sample, fetching the depth value from the shadow map, replacing the depth of projected world-space coordinate and then projecting it back onto the screen-space.

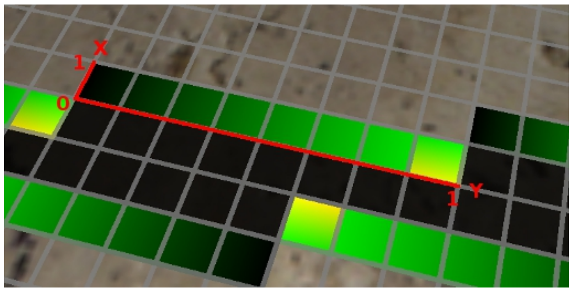


***Figure 3:*** *Orientated normalized discontinuity space, discontinuity transformed into the xy-coordinate space between 0 and 1.*

After we know where in screen-space our neighboring shadow-map sample is located, we step until we find a break in discontinuity. The discontinuity break is initiated either by exceeding delta-depth threshold or by reaching maximum search distance or by reaching a non-discontinuity sample on the opposite axis. By performing this iteration in screen-space, we approximate the length of the entering discontinuity.

Discontinuity contained in both channels (red and green) indicates a discontinuity-end. The discontinuity-end is used to determine the orientation of a discontinuity along the shadow silhouette edge. Knowing the discontinuity length and the discontinuity-end, we determine orientated normalized discontinuity space position (see figure 3). Once we have orientated normalized discontinuity space, we can fill the new shadow silhouette edge.

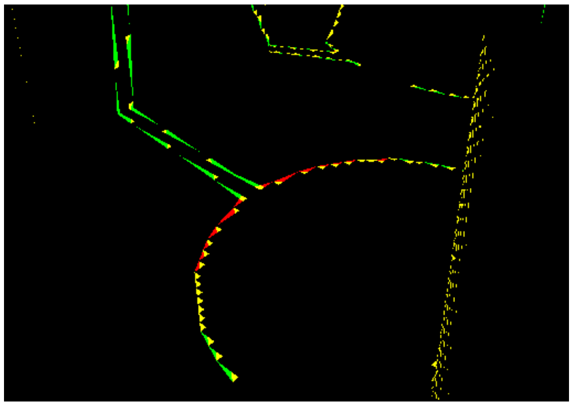


***Figure 4:*** *Clipped Area from Orientated Normalized Discontinuity Space.*

On low sample density areas, integrating new silhouette information into the lighting or image buffer-data will result in a smoother shadow edge.

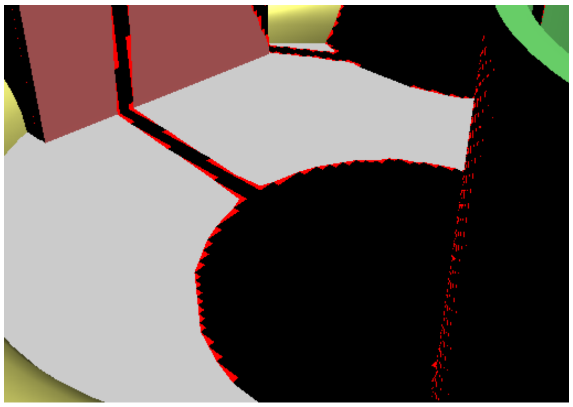


***Figure 5:*** *Red indicates new shadow silhouette.*

## 3. Results

SMSR technique performs particularly well in undersampled areas of the shadow map. It uses shadow discontinuities to approximate a new shadow silhouette, which results in improved visual quality.

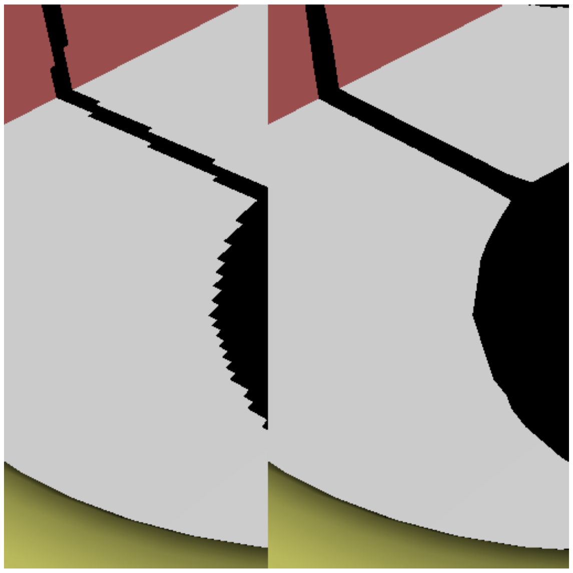


***Figure 6:*** *Left no SMSR, right with SMSR. 256×256 Shadow Map.*

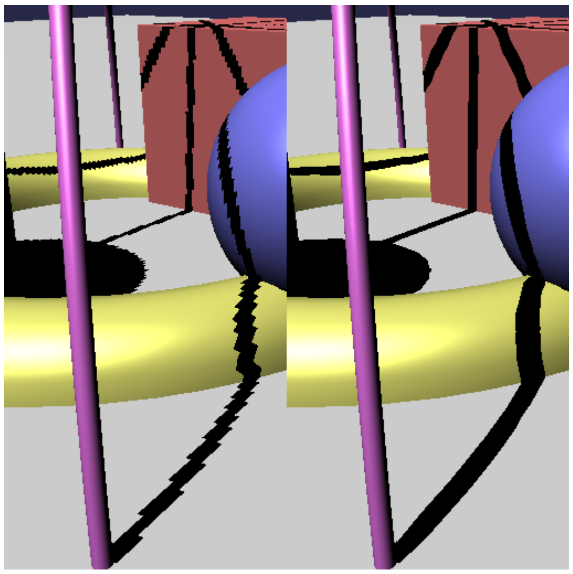


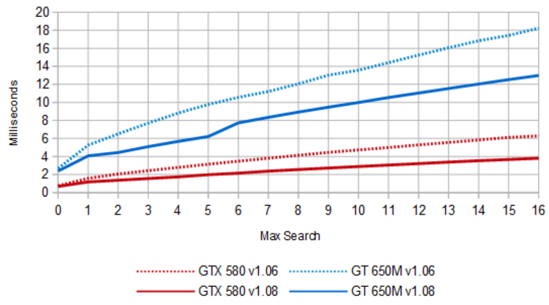
***Figure 7:*** *Left no SMSR, right with SMSR. 256×256 Shadow Map.*

## 4. Timings

Some machines exhibited a minor deviation of standard WUXGA resolution (a maximum deviation of ~120 pixels on vertical axis). In order to effectively compare the samples, time-samples are normalized back to WUXGA. The normalization is a very coarse estimation (the real scaling is not perfectly linear). Ideal sampling environment would require a machine with identical software and hardware setup, which is only possible for a certain range of GPUs.

| GPU / Resolution | WXGA (1280x720) | WUXGA (1920x1200) |
|---|---|---|
| GTX 275 | 2.45 | 4.91 |
| GTS 450 | 3.98 | |
| GTX 460 | 3.42 | 7.23 |
| GTX 470 | 1.80 | 3.95 |
| GT 540M | 5.06 | 9.00 |
| GT 555M | 6.89 | |
| GTX 560 Ti | 1.41 | 3.15 |
| GTX 570 | 1.31 | 2.54 |
| GTX 580 | 1.18 | 2.57 |
| GTX 670 | 1.97 | 2.49 |
| GTX 770 | 2.04 | 2.36 |
| GTX 780M | 2.18 | 2.87 |
| GTX 780 | 1.56 | 1.94 |

***Table 2:*** *WXGA vs WUXGA of SMSR v1.08. All timings are in milliseconds, max search is set to 8.*



***Graph 1:*** *Red, GTX 580. Blue, GT 650M. Continuous line, post optimization. Dotted line, pre-optimization. The "optimization" consists of moving 1x inverse and 2x matrix concatenation out of the search loop.*

It is debatable whether the performance is inside an acceptable range, however there are a couple of tricks to reduce the processing time:

- Reducing max search will significantly reduced processing time at the cost of revectorization quality (see graph 1 and figure 8).
- Calculating light-to-view matrix every single

fragment is a waste, the matrix can be moved to CPU side and parsed as a uniform parameter.

- In first pass, early discontinuity rejection if the sampling rates are equal or higher than one shadow map sample per camera-view texel.
- Before compiling the shader, variables such as shadow-map-size and bias-offset, can be replaced with constant values (shader pre-processing / shader builder).

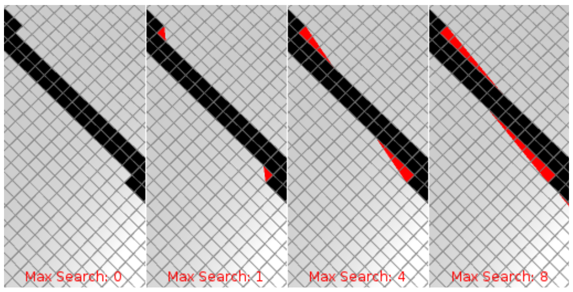
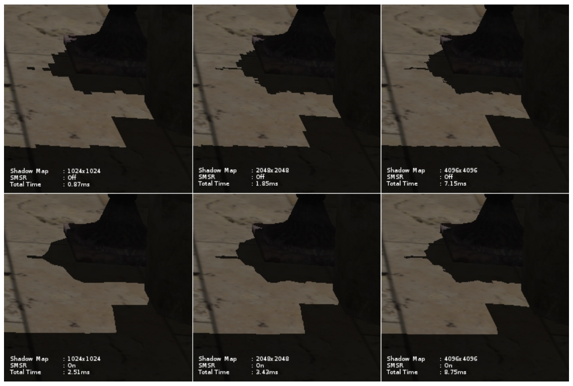


***Figure 8:*** *Various max search parameterization, red indicates new shadow silhouette. Smaller max search will result in more jaggy interpolation.*

### 4.1 Shadow map size: 1024 vs 2048 vs 4096

Performance-wise, the main strength of SMSR is being a screen-space algorithm which thrives on lower fill-rates of the shadow map. Densely populated scenery often results in high fill-rates during the rasterization of the shadow map. At a larger shadow map solutions, the fill-rates become unacceptable to run in real-time. This is where SMSR comes in, it reduces the fill-rates by approximating new shadow silhouette edge on lower shadow map resolutions and in many cases can also improve the final visual quality (see figure 10).



```
Shadow Map    : 1024x1024
SMSR          : Off
Total Time    : 0.87ms

Shadow Map    : 2048x2048
SMSR          : Off
Total Time    : 1.85ms

Shadow Map    : 4096x4096
SMSR          : Off
Total Time    : 7.15ms

Shadow Map    : 1024x1024
SMSR          : On
Total Time    : 2.51ms

Shadow Map    : 2048x2048
SMSR          : On
Total Time    : 3.43ms

Shadow Map    : 4096x4096
SMSR          : On
Total Time    : 8.75ms
```

***Figure 9:*** *Timing and visual quality of SMSR with different shadow map resolutions. Top, without SMSR. Bottom, with SMSR. Total time only represents shadow rasterization and SMSR (not a full frame-cycle).*

## 5. Inconsistencies

SMSR may be an effective way of improving the visual quality of low resolution shadow maps, however the algorithm also suffers from some major imperfections. Inconsistencies are categorized as special cases, discontinuity of discontinuity, and mangled silhouette shape.

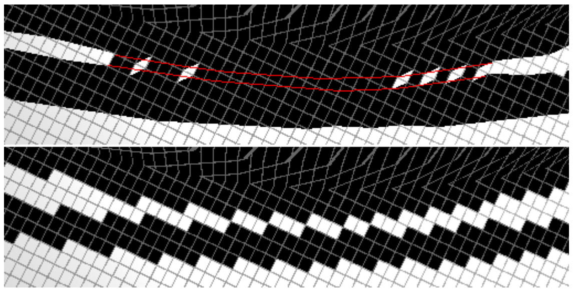


***Figure 10:*** *Top, with SMSR. Bottom, no SMSR. Two separate shadows are merged. Caused by discontinuity aiming in both directions on the same axis (code 0.75, see First Pass). Red lines indicate correct separation.*

Low shadow map resolution may leave out some geometrical detail, but it will still look relatively smooth and offer good performance. High resolution will provide more detail and smoothness, however the overall processing time will increase.

### 5.1 Special Cases

A combination of insufficient data and data generalization may result in unhandled special case scenarios. Leaving a special case unhandled, will result in unpredictable behavior. However, handling a special case will slow down the implementation and in some cases may also worsen the discontinuity (figure 10 and 11).

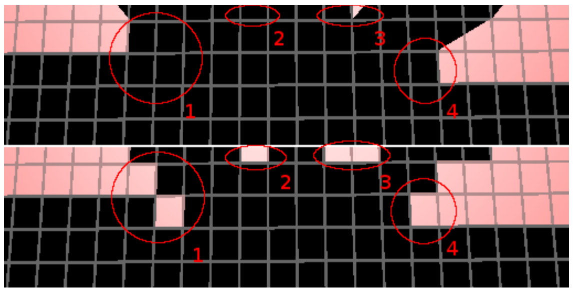


***Figure 11:*** *Top, with SMSR. Bottom, without SMSR. 1-discontinuity in two directions. Bottom, 4-discontinuity in three directions.*

### 5.2 Absence of Required Data

In specific case scenario, the shadow discontinuity is prematurely interrupted by an occluder (figure 13) or the search function reaches the end of the buffer (figure 14). These cases may result in incorrect or varying edge discontinuity length during the search step and results in a less natural edge reapproximation.

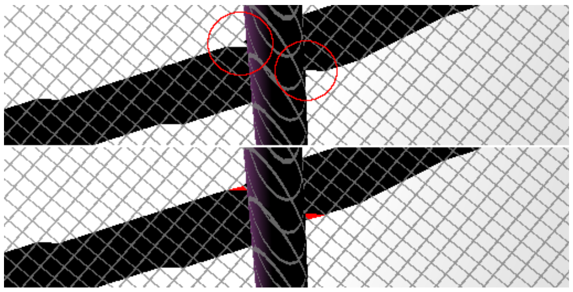


***Figure 12:*** *Top, error in interpolation marked with red circles. Bottom, correction filled with red.*

In figure 14, the shadow silhouette edge is partially clipped by the viewport, partially interrupting the discontinuity. As in most post-processing effects this type of problem is solvable by rending the viewport in a higher resolution and then clipping to the final resolution. This solution will work in this particular type of discontinuity, however it will not work if the edge-length discontinuity is occluded by an object (see figure 13).

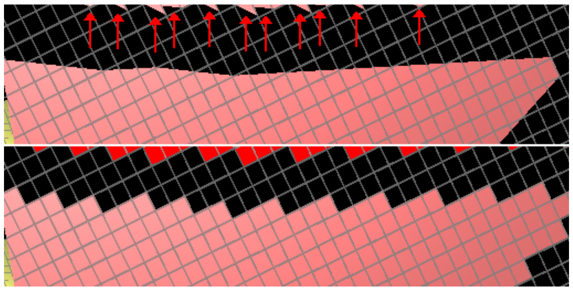


***Figure 13:*** *Top, red arrows aim at surface area which supposed to be filled. Bottom, correction marked with red.*

### 5.3 Mangled Silhouette Shape

A typical MLAA[3] approach distinguishes discontinuities into L, Z and U shape-patterns. Taking the shape pattern into account, helps to increase the precision of the edge reapproximation and results in a higher image quality. The current approach of SMSR is unable to distinguish shape-patterns and processes all discontinuities as L-shape-patterns. The inability to recognize shape-patterns leads to a coarse edge reapproximation and in some cases may also produce undesired results (figure 14).

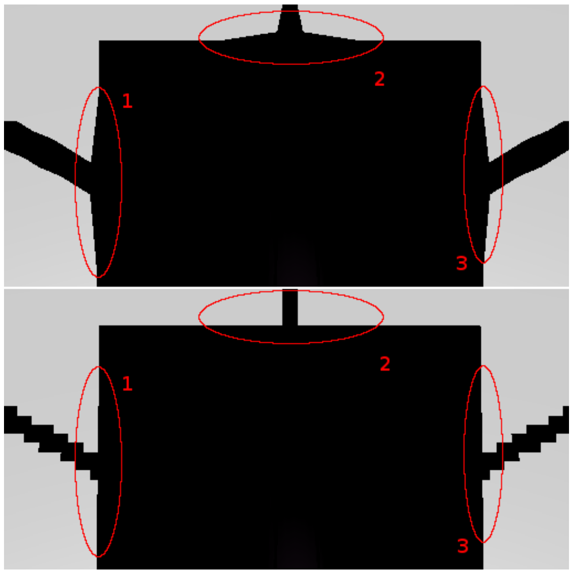


***Figure 14:*** *Top, incorrect silhouette approximations marked with red circles.*

## 6. Conclusion

Shadow Map Silhouette Revectorization is a filtering technique which is able to smoothen shadow map edges in undersampled areas. This technique particularly shines in large scenes containing high fill rates. It allows for the use of reduced shadow map sample density and yet maintain a smooth shadow silhouette edge. The reduction in sampling density of a shadow map reduces the rasterization latency, however it also reduces fine details and increases temporal aliasing with dynamic lighting and objects.

The technique is in its early stages and must be further researched in different areas such as interpolation based on shape patterns (to improve edge revectorization), soft shadows (to improve realism), temporal aliasing (to reduce jagged edges) and can be combined with other sample-redistribution techniques such as Cascade Shadow Maps (to optimize the use of shadow sample density where it is needed).

## 7. References

[1] Dimitrov, R. (2007). *Cascaded Shadow Maps*. Retrieved 27, 2013, from http://developer.download.nvidia.com/SDK/10/opengl/screenshots/samples/cascaded_shadow_maps.html

[2] Donnelly, W., & Lauritzen, A. (2006). *Variance Shadow Maps*. Retrieved December 27, 2013, from http://www.punkuser.net/vsm/

[3] Jimenez, J., Masia, B., Echevarria, J. I., Navarro, F., & Gutierrez, D. (2011). *Practical Morphological Anti-Aliasing*. Retrieved  27, 2013, from http://www.iryoku.com/mlaa/

[4] Johnson, G. S., Mark, W. R., Burns, C. A., & Lee, J. (2005). *The Irregular Z-Buffer and its Application to Shadow Mapping*. ACM Transactions on Graphics, 24(4), 1462-1482. Retrieved from ftp://ftp.cs.utexas.edu/pub/techreports/tr04-09.pdf

[5] Kim, J., & Kim, S. (2009). *Bilateral Filtered Shadow Maps*. ISVC '09 Proceedings of the 5th International Symposium on Advances in Visual Computing: Part II, 49-58.

[6] Lloyd, B., Govindaraju, N., & Quammen, C. (2007). *Logarithmic Shadow Maps*. Retrieved November 27, 2013, from http://gamma.cs.unc.edu/LOGPSM/

[7] Martin, T., & Tan, T. (2004). *Anti-aliasing and Continuity with Trapezoidal Shadow Maps*. Retrieved December 27, 2013, from http://www.comp.nus.edu.sg/~tants/tsm.html

[8] Reeves, W. T., Salesin, D. H., & Cook, R. L. (1987). *Rendering antialiased shadows with depth maps*. ACM SIGGRAPH Computer Graphics*, 24*(4), 283-291. Retrieved from http://maverick.inria.fr/Members/Cyril.Soler/DEA/Ombres/Papers/Reeves.Sig87.pdf

[9] Stamminger, M., & Drettakis, G. (2002). *Perspective shadow maps*. ACM Transactions on Graphics, 21(3), 557-562. Retrieved from http://www.cs.berkeley.edu/~ravir/6160-fall04/papers/p557-stamminger.pdf

[10] Williams, L. (1978). Casting curved shadows on curved surfaces. *ACM SIGGRAPH Computer Graphics, 12*(3), 270-274. Retrieved from http://design.osu.edu/carlson/history/PDFs/shadowmaps.pdf

[11] Wimmer, M., Scherzer, D., & Purgathofer, W. (2004). *Light Space Perspective Shadow Maps,* Institut für Computergraphik und Algorithmen – Arbeitsgruppefür Computergraphik. Retrieved  27, 2013, from http://www.cg.tuwien.ac.at/research/vr/lispsm/