Rectilinear Texture Warping for Fast Adaptive Shadow Mapping

Bjarke V. Grøn* Jon K. Sorensen†

Lund University
Sweden
December 11, 2012

Abstract

This report describes a project being a result of having followed the course *High Performance Computer Graphics* taught by Michael Doggett at Lund University, Sweden. It focuses on a technique called *Rectilinear Texture Warping (RTW) for Fast Adaptive Shadow Mapping*, which is a way to create shadows in 3D scenes that produces less aliasing artifacts than when using conventional shadow mapping.

Shadows are a very important factor in computer graphics and contribute largely to the level of realism. The quality of shadows produced with RTW shadow maps are noticeably better than that of conventional shadow maps while at the same time being very fast.

1 Introduction

We found the idea of using RTW for generating adaptive shadow maps very interesting after having read Paul Rosen's article "Rectilinear Texture Warping for Fast Adaptive Shadow Mapping" [Rosen 2012], and decided to focus on that topic as the project for the course.

The demand for higher quality, realistic images continue to be an important challenge in the area of real-time computer graphics. Shadows created using conventional shaddow mapping capture the data on a uniform grid, independent of the scene and the part of it that is inside the canonical view volume, thus usually producing "unnatural" edges on the shadows, unless the resolution of the shadow map in increased or a blur is applied.

RTW produces single-layer textures with importance-based sampling, and images are composed of a rectilinear warping map on top of a conventional texture. The sampling rate can be varied because of the warping map, and so RTW shadow maps are dynamic, adapting to view, scene and lighting conditions, and can produce shadows that are noticeably better, without a significant cost in performance.

2 Algorithms

We implemented the algorithm for producing an adaptive shadow map, as it is described in [Rosen 2012]. We used the Forward Analysis approach to create the importance map, and our importance function is checks whether a point is inside the canonical view volume or not. After having created a two-dimensional importance map, we convert it into two one-dimensional importance maps – one for the x-axis and one for the y-axis. This was done by saving the largest value in a column or row to the respective one-dimensional importance map. We implemented the two maps as the red and green channels respectively in a 1D texture.

Next, we blurred the two 1D importance maps with a standard Gaussian blur.

Then we calculated a warp map for both the x- and y-axis. This was done using equation 1 in [Rosen 2012], and implemented by looping over the corresponding 1D importance map for each pixel in the warp maps. Note that it is necessary to convert the values in the warp maps from $[-1, 1]$ to $[0, 1]$ before storing them in the warp map.

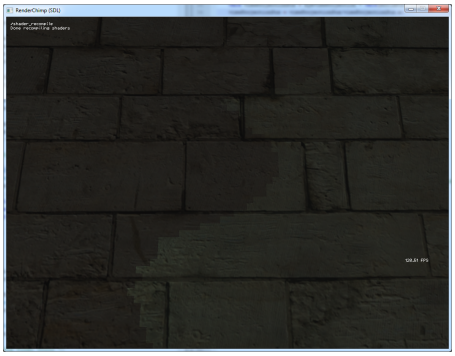

Figure 1: Conventional shadow mapping.

Figure 2: RTW shadow mapping.

Next we rendered the RTW shadow map. This was done by shifting the vertices, in the shadow map vertex shader, according to a lookup in the warp maps. Note that we shift the vertices around in the canonical view volume, thus their coordinates range from $[-1, 1]$. This means the values from the warp map had to be converted from the $[0, 1]$ interval, in which they were stored, to $[-2, 2]$, as they must be able to warp a vertice from one end to the other of the $[-1, 1]$ interval. After this is done we proceed as conventionally in the shadow map fragment shader.

In order to do a lookup in the RTW shadow map in the spotlight shader, we did an offset on the texture coordinate according to the value looked up in the warp map. Here the values had to be converted to a $[-1, 1]$ interval, since texture coordinates are in $[0, 1]$.

After the offset is applied to the texture coordinate, the lookup in the shadow map is done as conventionally.

3 Results

As can be seen when comparing conventional shadow mapping (figure 1) with the RTW implementation (figure 2), there can be a considerably large difference in effective shadow map resolution when

*e-mail: bjarkevg@gmail.com
†e-mail: jon.k.sorensen@gmail.com

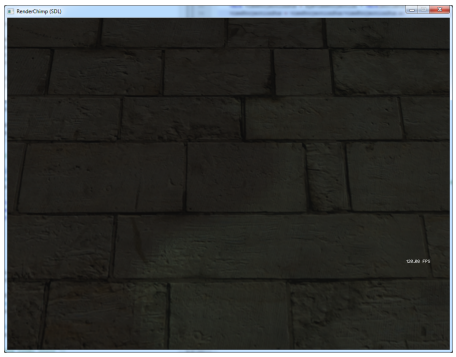

Figure 3: Conventional shadow mapping with Percentage Closer Filtering applied.

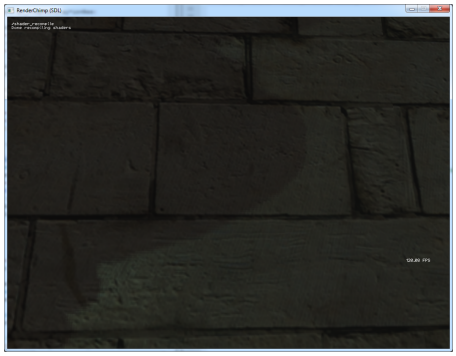

Figure 4: SFW shadow mapping with Percentage Closer Filtering applied.

zoomed in on the shadow edges. The RTW implementation also works nicely when applying a Percentage Closer Filter (figure 3 vs. figure 4). Notice that we have used a box filter that only blurs with the neighbors, so its easier to see the difference in resolution.

The advantage of using the RTW technique is noticeable (with respect to increased shadow resolution) when comparing this technique to conventional shadow mapping techniques 5.

4 Discussion

As big an improvement as the increase in shadow map resolution may be, our imlementation works rather poorly when moving the camera around. The constant movement of the geometry in the RTW shadow map shader makes the edges of the shadows shift

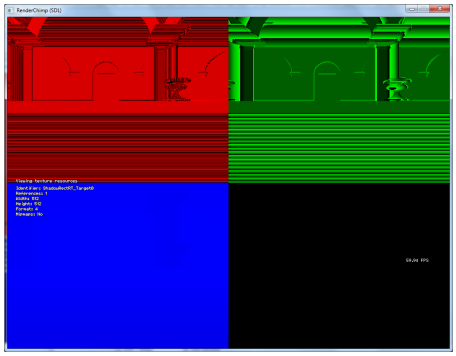

Figure 5: Conventional shadow map.

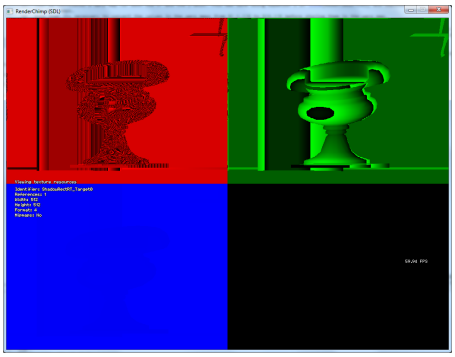

Figure 6: SFW shadow map.

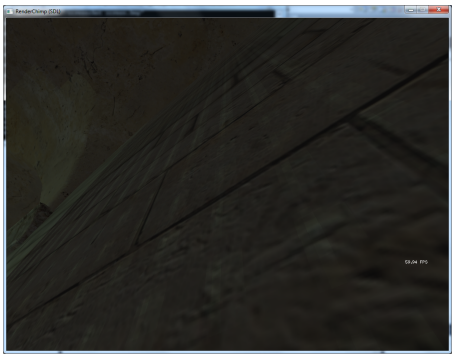

Figure 7: A bug in the shadows.

around quite noticeably whenever the camera is moved. There is the possibility that this would be less fo an issue if a better importance function was used, or maybe the 1D importance maps simply need to be blurred even more, to make the transitions in the warp map more smooth.

There is also some problems with the shadows at certain angles (figure 7). This is entirely due to our importance function, as can be seen in figure 8, where the pillar that is the most important part of the shadow map, is only a few pixels wide. When we use only what the camera sees in the forward analysis mode, objects close to the light can block entire rows/columns in the 2D importance map, when the camera is behind these objects.

One thing we noticed that were crucial for this technique to yield good results, was the need for using bilinear texture filtering, when doing lookups in the warp map. If just using nearest, all texture

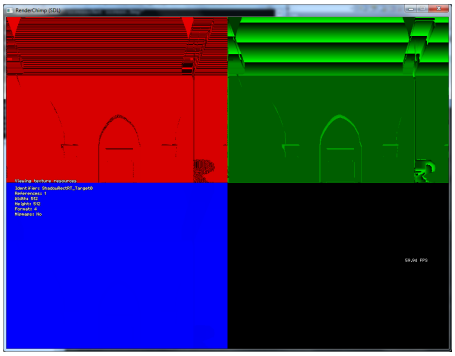

Figure 8: Shadow map when the shadows bug.

coordinates that fall within the same pixel in the warp map will be offset by the same amount. As a consequence the sampling from the RTW shadow map will not be smooth, and lead to very blocky shadows.

We experimented with implementing an importance function that also detected the edges of the shadows, but we found that the specific parameters of such an implementation were quite hard to get ”just right”.

We suspect that an implementation using the backwards analysis mode would yield better results. In backwards analysis you do the importance functions looking out from the camera, contra the forward analysis where you do it looking out from the light source. And this is the crux of the matter really, to determine what parts of the shadow map is important from the camera's viewpoint.

5 Conclusion

All in all we feel that with some time and care spend on perfecting the importance function(s) this technique can be really useful for producing high quality shadows. The biggest advantage is the ability to maintain a low resolution shadow map, while being able to retain high details in the shadows, thus saving a lot of memory on the graphics card.

References

ROSEN, P. 2012. Rectilinear texture warping for fast adaptive shadow mapping. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, 151–158.