Adaptive Shadow Maps

Randima Fernando Sebastian Fernandez Kavita Bala Donald P. Greenberg *

Program of Computer Graphics
Cornell University

Abstract

Shadow maps provide a fast and convenient method of identifying shadows in scenes but can introduce aliasing. This paper introduces the Adaptive Shadow Map (ASM) as a solution to this problem. An ASM removes aliasing by resolving pixel size mismatches between the eye view and the light source view. It achieves this goal by storing the light source view (i.e., the shadow map for the light source) as a hierarchical grid structure as opposed to the conventional flat structure. As pixels are transformed from the eye view to the light source view, the ASM is refined to create higher-resolution pieces of the shadow map when needed. This is done by evaluating the contributions of shadow map pixels to the overall image quality. The improvement process is view-driven, progressive, and confined to a user-specifiable memory footprint. We show that ASMs enable dramatic improvements in shadow quality while maintaining interactive rates.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shading,Shadowing;

Keywords: Rendering, Shadow Algorithms

1 Introduction

Shadows provide important information about the spatial relationships among objects [8]. One commonly used technique for shadow generation is the shadow map [9]. A shadow map is a depth image generated from the light source view. When rendering, points in the eye view are transformed into the light source view. The depths of the transformed points are then compared with the depths in the shadow map to determine if the transformed points are visible from the light source. A transformed point is considered lit if it is closer to the light than the corresponding point in the shadow map. Otherwise, the point is considered to be in shadow. This information is then used to shade the eye view image.

One of the major drawbacks of shadow maps is aliasing, which is depicted in Figure 1. Here we see views from two different locations in a simple scene: Viewpoint A and Viewpoint B. The grid in each picture shows the projected pixel area of the shadow map as seen by the current viewpoint. Since the projected area of the light source pixels in Viewpoint A is roughly equal to the area of the shadow map pixels in the shadow map, aliasing is minimal in this case. In contrast, Viewpoint B is quite close to the scene geometry, resulting in large projected areas for the shadow map pixels in this view. This means that large areas of Viewpoint B are shaded using relatively small amounts of information from the shadow map. The result is significant aliasing artifacts, depicted here as jagged edges.

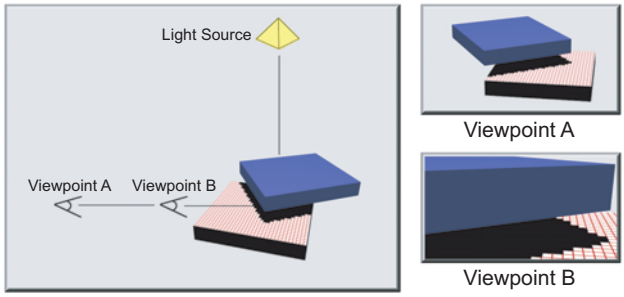

Figure 1: Aliasing in Shadow Maps.

In this paper we present the Adaptive Shadow Map (ASM), an extension to the conventional shadow map that addresses a primary cause of shadow map aliasing: inadequate shadow map resolution. We use a hierarchical grid structure to improve image quality on a view-driven and progressive basis by generating higher-resolution pieces of the shadow map when needed. A combination of least recently used (LRU) and priority-based memory management schemes constrains the algorithm's memory usage.

Lights are often placed in contrived locations to ensure adequate shadow quality because traditional shadow maps perform poorly when a light is far away or has a wide field of view. ASMs allow users to avoid such constraints by allowing realistic placements and fields of view. The view-driven nature of ASMs also produces high-quality shadows during interactive walkthroughs without careful tweaking of conventional shadow map parameters or the use of excessively large shadow maps.

2 Previous Work

A complete review of shadow generation algorithms is beyond the scope of this paper, but [6] and [11] together present a comprehensive review of the most popular methods. Introduced in [9], conventional shadow mapping is a common technique used to generate shadows. Since it is an image-space technique, it confers advantages in terms of generality, speed, and ease of implementation. However, removing aliasing in shadow maps has been a difficult problem to resolve. Percentage-closer filtering [7] is a solution to the aliasing problem, but in cases of severe aliasing it can only mask the aliasing by blurring. The light buffer [3] resolves the aliasing problem within the context of a non-interactive ray tracer by using a flat shadow map with analytical testing of shadows. Most recently, deep shadow maps [5] address aliasing by using jittered samples and pre-filtering them. However, they do not deal with aliasing caused by insufficient shadow map resolution.

*Email: {randy, spf, kb, dpg}@graphics.cornell.edu.
Web page: http://www.graphics.cornell.edu/pubs/papers.html

ACM SIGGRAPH 2001, 12-17 August 2001, Los Angeles, CA, USA

3 The Adaptive Shadow Map

ASMs are based on the observation that a high-quality shadow map need not be of uniform high resolution; only regions that contain shadow boundaries need to be sampled densely. In those regions, the resolution of the shadow map should be at least as high as the corresponding region in the eye view to avoid aliasing artifacts.

An ASM hierarchically subdivides an ordinary shadow map, providing higher resolutions in visually important regions. Like a conventional shadow map, an ASM takes as input a set of transformed eye view points and allows shadow queries on this set, returning true if a particular point is lit and false otherwise. In software systems, ASMs can seamlessly replace conventional shadow maps.

An ASM has three main characteristics:

- It is view-driven, meaning that the hierarchical grid structure is updated based on the user's viewpoint.
- It is confined to a user-specifiable memory limit. Memory is managed efficiently and the ASM avoids the explosive growth in memory usage that would be required by a conventional shadow map of the same visual quality.
- It is progressive, meaning that once a particular viewpoint is established, image quality continues to improve until the user-prescribed memory limit is reached.

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. Two operations may be performed on a cell in the tree. An empty cell may have a new node assigned to it when it is determined that the resolution of the shadow map region corresponding to the cell is not high enough to provide the required image quality. A cell containing a node may also have that node and its descendants deleted. This is done in response to the user-specified restrictions on memory usage.

3.1 Creating Nodes

At any time there are many cells that require new nodes to be assigned to them. In an interactive application, it is not always possible to fulfill all of these requirements. Therefore, we use a cost-benefit metric to determine which cells to satisfy. It is only beneficial to create a new node (and hence a higher resolution shadow map piece) if it causes a perceived improvement in shadow quality. We quantify this perceived benefit by counting the number of transformed eye pixels within the cell that straddle a depth discontinuity *and* whose shadow map resolution is lower than the eye view resolution. We use the mip-mapping capability of commodity graphics hardware to estimate the resolution in the eye view and in the shadow map. Section 4.1 explains this in detail.

The cost of creating a new node is the amount of time required to generate its new shadow map. Using the ratio of eye view pixel size to shadow map pixel size, the resolution required for the new shadow map to match the eye view resolution is:

$$N_{required} = N_{current} \frac{Eye\ View\ Pixel\ Projected\ Area}{Shadow\ Map\ Pixel\ Projected\ Area}$$

where N is the number of pixels in a shadow map cell.

The cost of generating a new shadow map can be approximated by:

$$cost = aN_{required} + b$$

This cost model is based on the fact that hardware read-back performance varies roughly linearly with the size of the buffer being read back, becoming increasingly inefficient as the read-back size gets very small. We perform a calibration test as a preprocess to evaluate a (the per-pixel rendering cost) and b (the constant overhead) for a given hardware setup.

The benefit of a new shadow map is the number of resolution-mismatched eye pixels that could be resolved. Once the cost-benefit ratio is computed for all prospective cells, the cells are sorted according to this ratio. To maintain consistent frame rates, new shadow map nodes are only generated for the most profitable cells from this list until a given time limit has passed.

3.2 Removing Nodes

Since the ASM only uses a fixed amount of memory, a particular node's memory might need to be reclaimed. In order to do so, we use a LRU scheme on all nodes that were not used in the last frame, removing the least recently used nodes. If there are no such nodes, then all nodes are currently visible. In this case, we remove an existing node only if a new node that needs to be created has a greater benefit than the existing node.

4 Implementation

This section discusses additional techniques and optimizations.

4.1 Mip-mapping

To determine when resolution mismatches occur, the algorithm must calculate the projected area of a pixel (i.e., the area that the pixel covers in world-space). Performing this calculation in software would be too expensive for interactive rates, so we approximate this calculation using mip-mapping.

Mip-mapping [10] is traditionally used to avoid aliasing artifacts associated with texture-mapping. Current graphics hardware implements perspective-correct mip-mapping, which interpolates between textures of different resolutions based on the projected area of the pixels being rendered. We use this feature to quickly approximate the projected pixel area as follows: the algorithm places the resolution of each mip-map level in all the texels of that level. Thus, the smallest level has a 1 in every texel, the second smallest level has a 2, the third smallest level has a 4, and so on. Texture coordinates are set up so that world-space texel sizes are uniform. Every polygon is then drawn with this mip-mapped texture. When the frame is read back, each pixel contains its trilinearly interpolated mip-map level, which is a reasonable approximation of its projected area. Anisotropic filtering is used to improve the accuracy of the approximation.

As a further optimization, the mip-map level is encoded only in the alpha channel while the rest of the mip-map is white. This allows the read-back to be done simultaneously with the polygon ID read-backs described below, eliminating an extra rendering pass.

4.2 Combining ID and Depth Comparisons

Conventional shadow maps commonly use a depth comparison with a bias factor to check if transformed pixels are lit. However, this approach exhibits artifacts on surfaces near edge-on to the light and has difficulties in scenes with varying geometric scale. Using per-polygon IDs to determine visibility instead of depth comparisons was proposed in [4]. This approach is better in many cases but results in artifacts along mesh boundaries. Therefore, we use a combination of per-polygon IDs and depth comparisons to perform the visibility determination for transformed pixels. If the ID test

fails, the conventional depth comparison allows us to avoid artifacts along polygon boundaries. Our results show that this simple modification is more robust than using just per-polygon IDs or depth comparisons, although the bias problem persists.

4.3 Optimizations

It is possible to accelerate ASM queries using a depth culling technique similar to that described in [2]. A cache storing the most recently queried leaf node further accelerates ASM lookups. Low-level optimizations such as using Pentium family SSE/SSE2 instructions could be used to speed up pixel reprojection from the eye view to the shadow map.

Our approach requires frequent renderings and read-backs as cells are refined. Since it is inefficient to redraw the whole scene for each grid cell during refinement, we use frustum culling for each cell in the topmost level of the hierarchy.

Since analysis of all pixels in the image can be expensive, our algorithm performs the cost-benefit analysis only on a fraction of the transformed pixels. This choice might result in slower convergence to an accurate solution. However, in our implementation, we found that analyzing one-eighth of the pixels gives good performance without significantly affecting the rate of convergence.

5 Results

Figures 2, 3, and 4 present our results for an interactive walkthrough of a 31,000-polygon scene at an image resolution of 512×512 pixels. Our timings were performed on a 1 GHz Pentium III with a NVIDIA GeForce2 Ultra graphics board. The scene features three different objects designed to test different aspects of our algorithm. The light source is a point light with a 122° field of view. It is placed on the room ceiling, far from the objects. The first object is a 20,000-polygon bunny model, which illustrates the algorithm's ability to deal with small triangles and frequent variations in polygon ID. The other two objects are a robot and a sculpture with a fine mesh, which demonstrate the algorithm's ability to find and refine intricate shadow details of varying scale.

During the walkthrough, a conventional 2,048×2,048 pixel shadow map (using 16 MB of storage with 32 bits of depth per pixel) averaged 8.5 frames per second, while our algorithm (also using 16 MB of memory) averaged 4.9 frames per second. Figures 2 and 3 illustrate the dramatic improvement in image quality achieved by the ASM. For close-ups of objects, the equivalent conventional shadow map size is very large (65,536×65,536 pixels in Figure 2 and 524,288×524,288 pixels in Figure 3). Creating such a shadow map in practice for an interactive application would be infeasible not only because of the long creation time, but also because of the enormous storage requirements.

Our results also demonstrate the ASM's ability to accommodate a wide field of view. Because of the ASM's view-driven nature, the starting shadow map size can be relatively small and its field of view can be relatively large. In our walkthrough, the starting resolution of the ASM was 512×512 pixels.

Figure 4 illustrates the algorithm's memory management. From left to right, we show images generated with a 2,048×2,048 pixel conventional shadow map, an ASM using 8 MB of memory, and an ASM using 16 MB of memory. The differences between the two ASM images are small, but both show considerable improvement in image quality when compared to the image on the left. To highlight the improvement in image quality from an 8 MB ASM to a 16 MB ASM, we have magnified two sections of each image.

The ASM used approximately 203 ms per frame, while a conventional 2,048×2,048 pixel shadow map used 117 ms for the same total memory usage (16 MB). The extra time was spent on cost-benefit analysis (30 ms), node creation (5 ms), traversals through the hierarchy for queries (35 ms), and an extra rendering and read-back of the scene to gather per-polygon ID information (16 ms).

6 Conclusions

This paper presents a new technique, the ASM, which uses adaptive subdivision to address aliasing in shadow maps caused by insufficient shadow map resolution. ASMs are view-driven, progressive, and run in a user-specifiable memory footprint. They interface with applications in the same way as conventional shadow maps, allowing them to be easily integrated into existing programs that use software shadow mapping. Since ASMs automatically adapt to produce high-quality images and do not require experienced user intervention, they should be useful in interactive modeling applications and in off-line renderers.

The algorithm presented in this paper can be extended to prefilter and compress refined shadow map cells using techniques described in [5]. In addition, perceptual masking [1] can be used to refine less in heavily masked areas.

7 Acknowledgements

We would like to thank Bruce Walter, Eric Haines, Parag Tole, Fabio Pellacini, and Reynald Dumont for their insights and comments, Linda Stephenson for her administrative assistance, and Rich Levy for his help in making the video. We would also like to thank the anonymous reviewers for their valuable comments. This work was supported in part by the National Science Foundation Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219). We also gratefully acknowledge the generous support of the Intel Corporation and the NVIDIA Corporation.

References

[1] J. A. Ferwerda, S. N. Pattanaik, P. Shirley, and D. P. Greenberg. A Model of Visual Masking for Computer Graphics. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 143–152, August 1997. T. Whitted, editor.

[2] N. Greene and M. Kass. Hierarchical Z-Buffer Visibility. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 231–240, August 1993. J. T. Kajiya, editor.

[3] E. A. Haines and D. P. Greenberg. The Light Buffer: a Shadow Testing Accelerator. *IEEE Computer Graphics and Applications*, 6(9):6–16, September 1986.

[4] J. C. Hourcade and A. Nicolas. Algorithms for Antialiased Cast Shadows. *Computer and Graphics*, 9(3):259–265, 1985.

[5] T. Lokovic and E. Veach. Deep Shadow Maps. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 385–392, July 2000. K. Akeley, editor.

[6] T. Möller and E. A. Haines. *Real-Time Rendering*. A. K. Peters, Massachusetts, 1999.

[7] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering Antialiased Shadows with Depth Maps. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):283–291, July 1987. M. C. Stone, editor.

[8] L. R. Wanger, J. A. Ferwerda, and D. P. Greenberg. Perceiving Spatial Relationships in Computer-Generated Images. *IEEE Computer Graphics and Applications*, 12(3):44–58, May 1992.

[9] L. Williams. Casting Curved Shadows on Curved Surfaces. *Computer Graphics (Proceedings of SIGGRAPH 78)*, 12(3):270–274, August 1978. R. L. Phillips, editor.

[10] L. Williams. Pyramidal Parametrics. *Computer Graphics (Proceedings of SIGGRAPH 83)*, 17(3):1–11, July 1983. P. Tanner, editor.

[11] A. Woo, P. Poulin, and A. Fournier. A Survey of Shadow Algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.

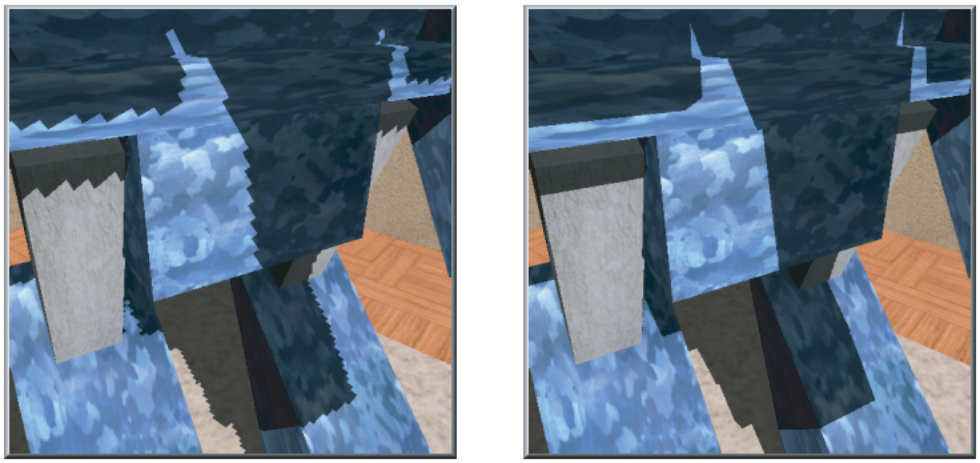

Figure 2: A conventional 2,048×2,048 pixel shadow map (left) compared to a 16 MB ASM (right). EFFECTIVE SHADOW MAP SIZE: 65,536×65,536 PIXELS.

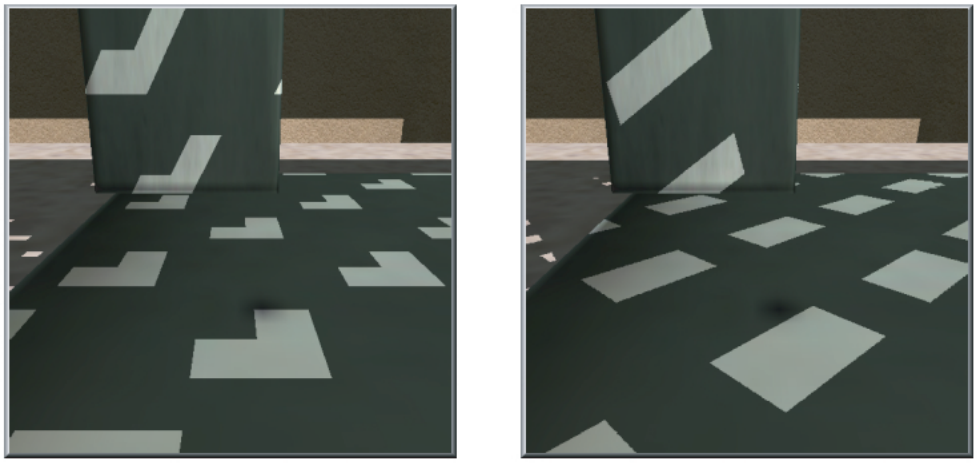

Figure 3: A conventional 2,048×2,048 pixel shadow map (left) compared to a 16 MB ASM (right). EFFECTIVE SHADOW MAP SIZE: 524,288×524,288 PIXELS.

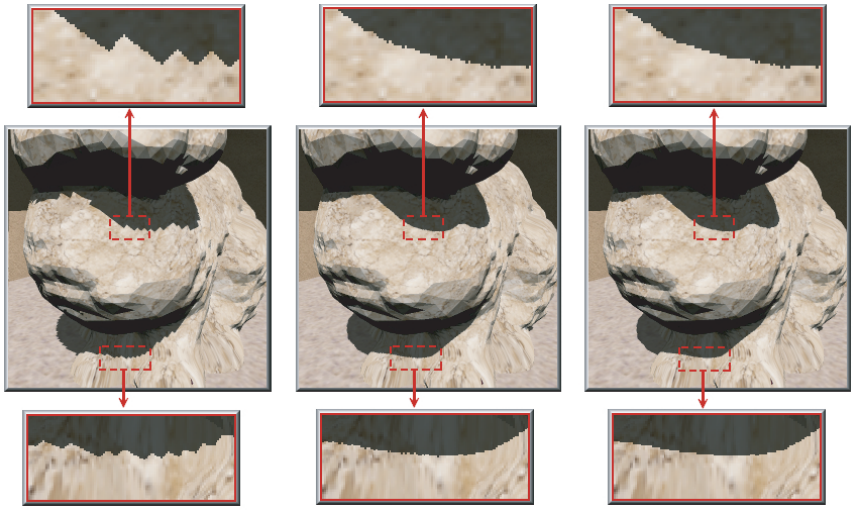

Figure 4: A conventional 2048×2048 pixel shadow map (left), an 8 MB ASM (center), and a 16 MB ASM (right).