Multi-View Soft Shadows

Louis Bavoil
lbavoil@nvidia.com

March 2011

Document Change History

Version	Date	Responsible	Reason for Change
1.0	March 16, 2011	Louis Bavoil	Initial release

Overview

The Multi-View Soft Shadows (MVSS) algorithm renders contact-hardening shadows by averaging hard shadows from multiple point lights evenly distributed on an area light. This is similar to accumulation-buffer rendering [Haeberli and Akeley 90], the main difference being that with MVSS the scene is rendered a single time from the eye's point of view. First, the scene geometry is rasterized into multiple shadow maps with one shadow map per point light. Second, soft shadows are rendered by averaging the hard shadows from each shadow map in a pixel shader. In this DirectX 11 code sample, the shadow maps are implemented as a depth texture array, they are generated with one depth-only pass per shadow map, and they are fetched with one bilinear hardware-PCF sample per shadow map.

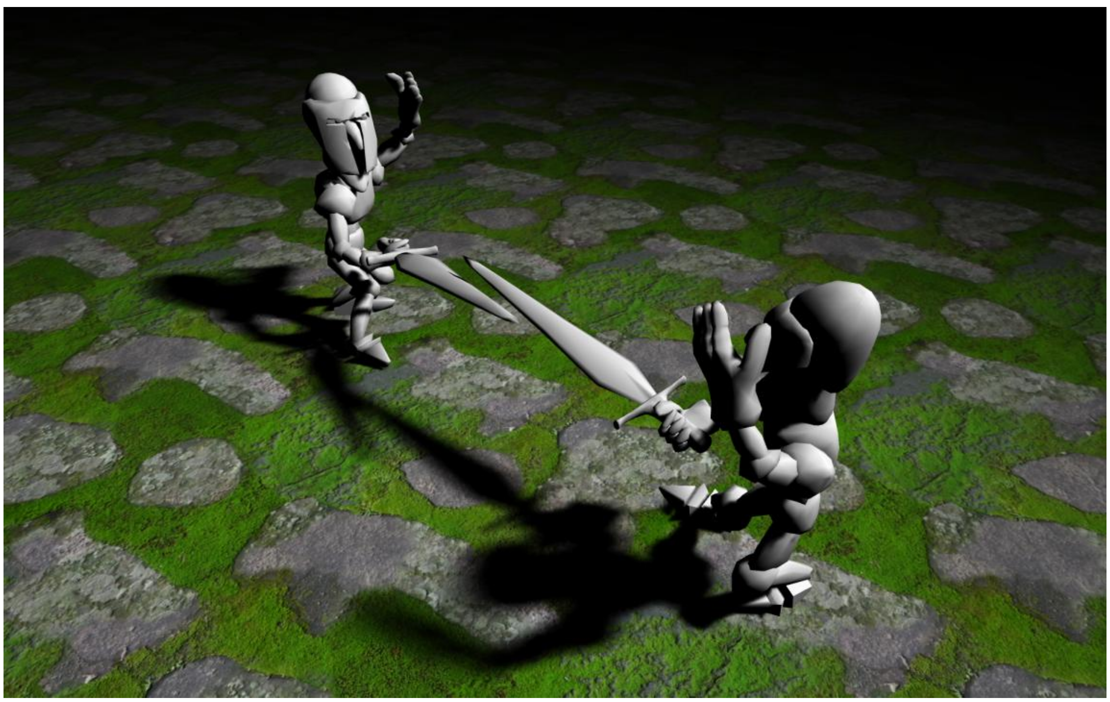

Figure 1. Multi-View Soft Shadows (MVSS) with 28 shadow maps. Rendered in 2.7 ms in 1920x1200 4xMSAA on GeForce GTX 580.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Introduction

The purpose of the Multi-View Soft Shadows algorithm is to render realistic contact-hardening soft shadows in real time. Contact-hardening soft shadows are particularly important for realistic rendering because the human visual system uses the variations of penumbra size as a cue for evaluating distances between shadow-casting and shadow-receiving objects [Fernando 05] [Myers et al. 08].

Among the vast literature of soft-shadow rendering algorithms (see [Eisemann et al. 09] for a recent survey), two of them are commonly used for generating ground-truth images: distributed ray tracing [Cook et al. 84] and accumulation-buffer rendering [Haeberli and Akeley 90]. Multi-View Soft Shadows takes the latter approach, sampling the area light with point lights and using one shadow map per point-light sample.

Percentage Closer Filtering (PCF) [Reeves et al. 87] is a popular algorithm for rendering dynamic shadows in games. Shadows rendered with PCF can be made softer by increasing the footprint of the PCF kernel but they do not harden on contact. The Percentage-Closer Soft Shadows (PCSS) algorithm [Fernando 05] is an extension of PCF which can render physically-plausible contact-hardening shadows.

Like PCSS, Multi-View Soft Shadows (MVSS) uses a depth bias to avoid self-shadowing artifacts (surface acne), by pushing the shadow-map fragments away from the light source. For PCF kernels, the larger is the kernel, the larger the depth bias must be to avoid surface acne. Since MVSS uses a smaller PCF kernel than PCSS (2x2 for MVSS, dynamic size for PCSS), MVSS can use a smaller depth bias without introducing artifacts, which makes it more robust with regard to self-shadowing.

Implementation

Shadow-Map Passes

In this SDK sample, the light source is a disk and point-light samples are distributed over the area of the light by using a Poisson-disk sampling pattern [Bridson 07].

The shadow maps are implemented as a depth texture array with one slice per shadow map. Before drawing into it, the depth texture array is cleared at once by performing a hardware Clear on a depth-stencil view encompassing the entire texture array.

For each point-light sample, the depth-stencil view is set to the corresponding slice in the depth texture array, and the hardware depths of the shadow-casting geometry are rendered into the current shadow map. To avoid surface acne, the hardware depth bias is enabled and both a constant depth bias and a slope-scaled depth bias are used.

Shading Pass

For each shaded fragment, the 3D position of the fragment needs to be projected onto each shadow map (4x4 matrix multiply followed by a perspective division), and the shadow contributions from each shadow map need to be computed and averaged together.

The vertex shader performs the 4x4 matrix multiply for each point light, and the resulting (u,v,z,w) coordinates (one float4 per shadow map) are passed down to the pixel shader as interpolated attributes with perspective-correct interpolation.

Performance

Table 1 presents a performance analysis of this SDK sample, for scenes with 2 shadow-casting characters (12,086 triangles) and 6 shadow-casting characters (36,258 triangles), using 28 512^2 shadow maps. In these results, the performance is limited mainly by the shadow-map filtering pass, not by the shadow-map generation passes.

Increasing the shadow-map resolution from 512^2 to 1024^2 may cause a 1-10% performance hit on the shadow-map generation and a 1-2% hit on the shadow-map filtering, assuming that all the shadow maps fit in video memory (28 MB for 28 512^2 32-bit shadow maps).

Table 1. Performance in 1920x1200 1xMSAA on GeForce GTX 580.

GPU Time	12,086 Triangles	36,258 Triangles
Shadow-Map Generation	0.3 ms (14%)	0.9 ms (27%)
Forward Rendering	1.9 ms (86%)	2.4 ms (73%)
Total	2.2 ms	3.3 ms

Code Organization

In MultiViewSoftShadows.h, Scene::Render() is the main rendering method, which renders the shadow maps and performs the shading pass.

MultiViewSoftShadows.cpp contains the code for the GUI and DXUT callbacks.

The HLSL source code of the shaders is located in the "shaders" subdirectory.

Running the Sample

The left button controls the light position and the right button the camera position. The mouse wheel zooms in and out.

The light source is a disk area light. The "Light Size" slider changes the radius of the light.

The "Visualize Depth" check box outputs the shadow-map depths on the screen. The visualized depths are the one from the last-rendered shadow map.

In the GUI, the sample displays the GPU time spent generating the 28 shadow maps, and performing the shading pass, as well as the number of shadow-casting triangles rendered in each shadow map, and the total amount of allocated video memory for the shadow maps.

Conclusion

With previous-generation GPUs, rendering the geometry of the shadow-casting objects multiple times was typically impractical for games. With NVIDIA's Fermi GPUs, Multi-View Soft Shadows (MVSS) have become more practical for casting realistic contact-hardening soft shadows from a limited number of triangles, such as in-game characters.

References

[Eisemann et al. 09] E. Eisemann, U. Assarsson, M. Schwarz, M. Wimmer, "Casting Shadows in Real Time", ACM SIGGRAPH ASIA 2009 Courses, 2009.

[Myers et al. 08] K. Myers, R. Fernando, L. Bavoil, "Integrating Realistic Soft Shadows into Your Game Engine", NVIDIA Whitepaper, 2008.

[Bridson 07] R. Bridson, "Fast Poisson disk sampling in arbitrary dimensions", ACM SIGGRAPH 2007 Sketch Program, 2007.

[Fernando 05] R. Fernando, "Percentage-Closer Soft Shadows", ACM SIGGRAPH Sketch Program, 2005.

[Haeberli and Akeley 90] P. Haeberli , K. Akeley, "The Accumulation Buffer: Hardware Support for High-Quality Rendering", Proceedings of ACM SIGGRAPH, 1990.

[Reeves et al. 87] W. Reeves, D. Salesin, R. Cook, "Rendering Antialiased Shadows with Depth Maps", Proceedings of ACM SIGGRAPH, 1987.

[Cook et al. 84] R. Cook , T. Porter , L. Carpenter, "Distributed ray tracing", Proceedings of ACM SIGGRAPH, 1984.