To appear in ACM TOG 30(6).

Supplemental Results for ACM SIGGRAPH Asia 2011 Paper: Genetic Programming for Shader Simplification

Pitchaya Sitthi-amorn Nicholas Modly Westley Weimer Jason Lawrence
University of Virginia

Shader	Add	Replace	Swap	Remove	Cross Over
V. Shadow Map	64	74	50	68	364

Table 1: *Distributions of mutation operators use to obtain the set of results along the Pareto frontier for each non-trivial test shader.*

This document contains results of our simplification system applied to a technique for approximating soft shadows.

1 Variance Shadow Map (Multiple Pass)

Variance Shadow Maps (VSM) is a technique for approximating soft shadow edges developed by Donnelly and Lauritzen [2006]. This shader is comprised of three passes. The first pass computes the per-pixel depth and square of the depth from the light source's vantage point. The second pass blurs this depth information using a 7×7 Gaussian filter. The final pass combines these previous computations to approximate a soft shadow boundary using Chebychev's inequality. We measured the performance of this shader using a mobile device with NVidia's Tegra 2 chipset. This further illustrates the ability of our system to optimize a shader for a wide range of hardware platforms.

Figure 1 shows the set of shaders our simplification system discovered. The shader variant at the red dot removes the perspective divide from the step that computes the square of the per-pixel depth and reduces the number of samples in the blur kernel from 7 to 6. This variant reduces the rendering time by 18% with no visible error.

The second variant corresponding to the orange dot reduces the complexity of the analytical formula of the Gaussian blur filter while retaining all 7 samples. Although this produces slightly sharper shadow boundaries, it does not cause aliasing. Using this variant reduces the rendering time by 33% compared to the original. The final variant, marked by a purple dot, reduces the shader to a standard shadow map algorithm. The fact that our system was able to automatically identify this algorithm as an approximation to a more sophisticated soft shadow algorithm undermines the ability of our GP search to uncover profitable areas within a highly non-linear optimization terrain.

Finally, Table 2 gives the full statistics for this shader (compare this to Table 1 in the main paper), and Table 1 shows the distributions of the different mutation operators that were used during our GP search to obtain the set of results along the Pareto frontier. Altogether, our system generated 229 variants of this shader in 4.3 hours.

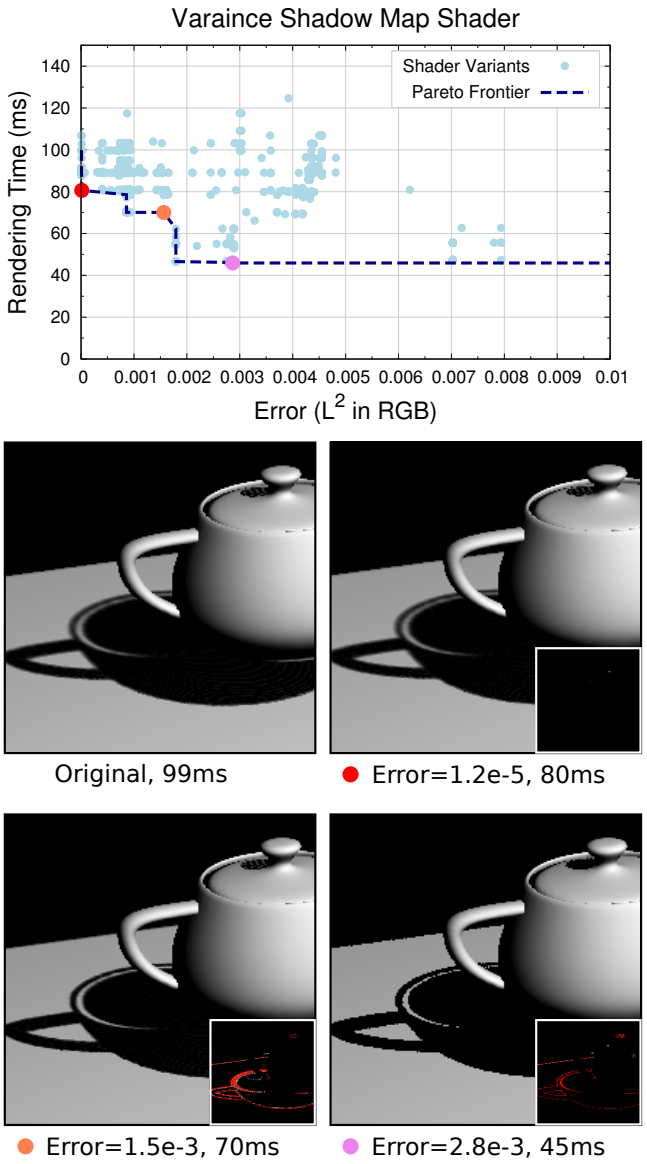

Figure 1: **Top:** *Scatter plot of error and performance of different shader variants generated by our system on the Variant Shadow Map shader.* **Bottom:** *Selected rendering results as indicated in the graph above. The inset contains a visualization of the per-pixel error.*

References

DONNELLY, W. and LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, New York, NY, USA. ACM, pages 161–165. ISBN 1-59593-295-X. URL `http://doi.acm.org/10.1145/1111411.1111440`.

	Lines of Code		Time (hours)			Shader Variants			Speedup @ ≤ 0.075 L^2 error
Shader	Source	Asm	Compiling	Testing	Total	Generated	Unique	On Frontier	Our Approach
V. Shadow Map	129	67	0.2	4	5	7,035	3,911	228	2.00x

Table 2: *Two shaders used in our supplemental experiments. "Lines of code" counts non-comment, non-blank lines. "Compiling time" gives the total time taken by the shader compiler over all variants. "Testing time" reports the time spent evaluating the error and performance of variants. "Total time" includes the total time spent performing GP bookkeeping, compiling time and testing time. "Generated Variants" counts all distinct shader source codes produced; "Unique Variants" counts the number of unique assembly produced. "Variants on Frontier" gives the number of shaders on the Pareto frontier: the size of the final set of error-performance tradeoffs produced by our algorithm. The "Speedup @* ≤ 0.075 L^2 *error" columns give the best speedup obtained (original rendering time divided by optimized rendering time) by a variant with* L^2 *RGB error at most* 0.075 *for our approach.*