HAL
open science

Practical morphological antialiasing on the GPU

Venceslas Biri, Adrien Herubel, Stéphane Deverly

To cite this version:

Venceslas Biri, Adrien Herubel, Stéphane Deverly. Practical morphological antialiasing on the GPU. SIGGRAPH 2010, Aug 2010, United States. pp.45. hal-00681574

HAL Id: hal-00681574

https://hal.science/hal-00681574v1

Submitted on 21 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Practical morphological antialiasing on the GPU

Venceslas Biri*
Institut Gaspard Monge

Adrien Herubel†
Institut Gaspard Monge

Stephane Deverly‡
Duran Duboi

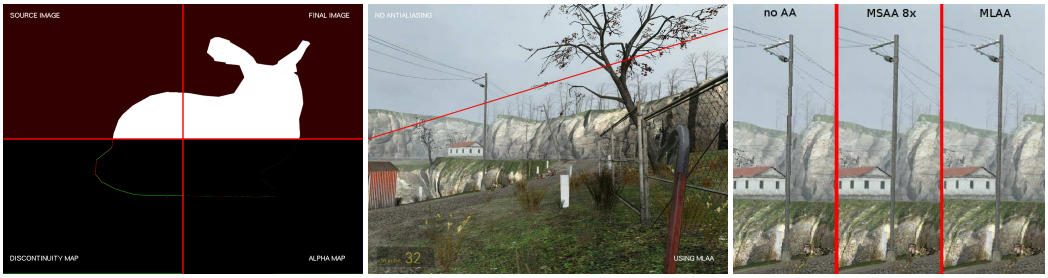

Figure 1: *Left. 4 layers used. Middle : Result on a test image. Right : Comparison between MSAA 8x and our MLAA*

Keywords: antialiasing, gpu, real-time

1 Introduction

The subject of antialiasing techniques has been actively explored for the past 40 years. The classical approach involves computing the average of multiple samples for each final sample. Graphics hardware vendors implement various refinements of these algorithms. Computing multiple samples (MSAA) can be very costly depending on the complexity of the shading, or in the case of ray-tracing. Moreover, image-space techniques like deferred shading are incompatible with hardware implementation of MSAA since the lighting stage is decorrelated from the geometry stage. A filter based approach called Morphological Antialiasing (MLAA) was recently introduced [2009]. This technique does not need multiple samples and can efficiently be implemented on CPU using vector instructions. However, this filter is not linear and requires deep branching and image-wise knowledge which can be very inefficient on graphics hardware. We introduce an efficient adaptation of the MLAA algorithm running flawlessly on medium range GPUs.

2 MLAA on the GPU

Overview In the MLAA algorithm, L-shaped edges are identified in the final image and samples are blended along the formed triangles. The value of the blended samples depends on the area of the trapeze covering each sample.

Line length detection We start by identifying the discontinuities between the color samples and write them in a discontinuity map where red and green components store respectively vertical and horizontal discontinuities. To detect discontinuity we switch to CIE L*a*b* color space, and compute a color difference which is a practical way to control the quality of the blending while avoiding the caveats of luminance-based discontinuity. Then we compute line and column lengths in two textures. This algorithm is based on the recursive doubling technique used in SAT generation [2005] and results are obtained in $log(width) + log(height)$ passes. Each sample of the two resulting textures contains distances to the discontinuity both ways.

*e-mail: biri@univ-mlv.fr
†e-mail: herubela@esiee.fr
‡e-mail: sdeverly@quintaindustries.com

Computing areas The area computing pass is the most costly since it is not linear. For each sample we have to identify its position in any L-shape formed by a column and a line using distances to discontinuities previously computed. This relative position allows us to estimate the area α of the sample, in one particular direction, covered by the resulting trapeze. We use a precomputed area table texture which is a 512x512 floating point texture with one component. It implies that we only filter lines with a maximum of 512 pixels which is enough for real world scenes. Therefore, in the shader, we identify any of the 8 possible L-shape and evaluate the area of the trapeze needed to blend sample with its 4-neighbours.

Blending In the final pass of the algorithm we blend the value of the sample with its 4-neighbours using the α values computed in the previous stage.

3 Results and discussion

Our implementation adds a total cost of 34ms (3.49ms) to the rendering at resolution 1248x1024 on a NVidia Geforce 8600 GT (295 GTX). The GPU version tends to scale very well since the cost at 1600x1200 resolution is only 67.5ms (5.54ms) which represents a cost 98% (65.3%) higher for 144% more pixels. We can compare our results to a standard CPU implementation which runs in 67ms at 1024x768 and in 128ms at 1600x1200 on a Core2Duo 2.20Ghz. Note that it does not include the costly GPU/CPU/GPU transfers in case of real time rendering. We provide an open source OpenGL/GLSL implementation of our method at http://igm.univ-mlv.fr/~biri/mlaa-gpu/.

Further works will consist in handling artifacts introduced by filter approaches in animation using techniques such as temporal coherence and auto determination of the discontinuity factor.

References

HENSLEY, J., SCHEUERMANN, T., COOMBE, G., SINGH, M., AND LASTRA, A. 2005. Fast summed-area table generation and its applications. In *Computer Graphics Forum*, vol. 24, Citeseer, 547–556.

IOURCHA, K., YANG, J., AND POMIANOWSKI, A. 2009. A directionally adaptive edge anti-aliasing filter. In *Proceedings of the 1st ACM conference on High Performance Graphics*, ACM, 127–133.

RESHETOV, A. 2009. Morphological antialiasing. In *Proceedings of the 2009 ACM Symposium on High Performance Graphics*.