Computer Graphics - Lab 5

Current developments in Computer Graphics

Analysis of modern anti-aliasing methods

Jan 26, 2017

Introduction

Computer graphics is today a big topic and widely applied in different kind of situations. When we look at a computer screen, it is not uncommon for it to display 3D rendered objects. The data that is used for modelling these objects have most often gone through a pipeline of calculations and computing. The reason for this is to achieve a visually good representation of the object.

Aliasing is a well-known problem today in computer graphics. This symptom appears from rasterization of a scene to a finite amount of pixels (a screen). For instance, if a diagonal line is rendered, the aliasing effect causes a stair-like jagged look to the line. To prevent this effect from occurring, different types of anti-aliasing methods can be applied.

That is why this report will be about aliasing and how modern methods and algorithms are used to combat it. There will be a brief explanation of each method at the beginning and at the end there will be comparisons between these methods and a discussion.

Problem Description

The chosen methods are of PPAA-algorithms (Post-Processing Anti-Aliasing), which means that the anti-aliasing takes place after the rasterization is complete, unlike MSAA (Multisample Anti-Aliasing) or SSAA (Super Sampling), which is not a PPAA-algorithm. These algorithms are run during the rasterization-process by sampling the geometry of an image at a higher resolution and then blending the pixels together. Due to the how these algorithms impact the performance of a system, several other algorithms has been introduced, mostly PPAA-algorithms.

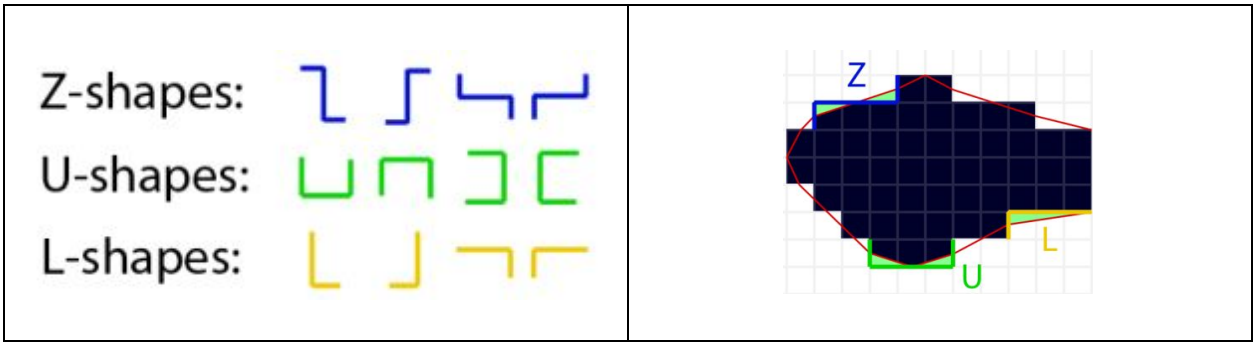
These PPAA-algorithms takes the rendered image and blurs the edges. This doesn't impact the performance of a system as much, since it is processed very fast. However PPAA has it's downsides as well since the only data it has is pixel data, and image detail cannot be gained since nothing is rendered at a higher resolution. That's why in this report, some of the well-known algorithms which will be looked into. SMAA and FXAA are well-known by now, but a new addition to the PPAA-algorithm family will be summarized and compared as well, called AXAA.

Anti-aliasing methods

In this section brief explanation of each algorithm which will be discussed and compared later, will be explained. This is to give the reader a small understanding of what each algorithm does and where in the process the enhancements are made and what causes the different image differences.

MLAA - MorphoLogical Anti-Aliasing

Jorge Jimenez's MLAA estimates the pixel coverage of the original geometry. It first finds vertical and horizontal edges by comparing the colour of a pixel and two neighbour pixels (bottom and right). The edges are categorized according to three predefined patterns in the MLAA-algorithm. These patterns consists of the following formations: L, Z, U. Further information can be found in the study by Jorge Jimenez et al. but the following figures, from the same study, may explain visually what is meant by patterns around detected edges.



From the pattern-classification of the edges, re-vectorization creates a coverage area for the pixels on these edges. The estimated coverage for a pixel is decided by where in the edge-pattern it's located. The coverage area is then blended with the right or bottom neighbour to calculate the final colour of the pixel.

SMAA - Sub-pixel MorphoLogical Anti-Aliasing

This subsection will only explain how the SMAA method works. It is not a PPAA technique however a short explanation of how this algorithm works, will aid in understanding when comparisons are made in the coming sections.

SMAA is a derivate of MLAA, also proposed by Jorge Jimenez et al. Furthermore, this method is similar to MLAA, but includes another edge detection technique through local contrast adaptation and new introduced shape patterns for diagonal edges, accurate searches for patterns and subpixel features through the use of multisampling or temporal supersampling in SMAA x2 and up. [3]

FXAA - Fast approXimate Anti-Aliasing

FXAA is currently the fastest PPAA (Post Processing Anti Aliasing) algorithm available. The general aim for this method is to with high speed, produce an anti-aliasing solution that looks good enough, rather than to produce a perfect graphical solution (like many other PPAA techniques). This technique uses a single render pass, calculates and produces an estimate colour (blur) of the edges through edge-detection. Using the method early-exit for pixels, it is able to skip pixels that are not in need of anti-aliasing. This is done by fetching four luma values from the neighbouring pixels, storing the maximum value and comparing it to the contrast range of these pixels, which is the difference between the highest luma value and the lowest luma value. If the contrast range is greater than the maximum luma value (multiplied with a set threshold constant) it will be anti-aliased, otherwise no anti-aliasing will occur. [1]

FXAA can sometimes cause thin lines and geometry to become excessively blurry and undetailed. There have recently been done experiments to improve this algorithm and provide better visualisation by Jae-Ho Nah et al. in the AXAA: Adaptive approXimate Anti-Aliasing (2016) paper. [3]

AXAA - Adaptive approXimate Anti-Aliasing

This algorithm is an enhancement of the previously mentioned FXAA algorithm. It consists of three additional steps compared to the FXAA algorithm. From the already pre-existing early-exit in the FXAA algorithm, an additional early-exit condition was introduced to avoid duplicate anti-aliasing of pixels. This was done by comparing the median luma values of the current pixel and its neighbours. By checking whether the luma values of the current pixel or its neighbours are within the range of the median value plus minus a constant, it is possible to avoid unnecessary anti-aliasing filtering of non-edge pixels.

Another step included conserving contrast of thin lines by comparing the difference of the current pixel and other pixels in one direction, to a constant value. If the difference is higher than this constant value, the sub pixel offset is set to zero to avoid bilinear filtering, to prevent small fonts and other thin geometry to become blurred. To prevent overhead from these steps and to increase performance, the number of iterations is adaptively limited compared to FXAA - where more iterations means higher quality. The iterations are limited by observing the luma contrast of the pixel and is set accordingly.

Comparisons and conclusions

Performance

Performance is of high importance when it comes to real time rendering of a video game or some other kind of simulation. Using anti-aliasing -methods that performs bad, can result in lower frame rate, which makes the simulation to run slowly or laggy. This can give the player/user/observer a bad or unrepresentative experience.

MSAA is one of the most well-known anti-aliasing methods. In most modern video games or other applications using computer graphics, MSAA is an option, with different sample rates (2X, 4X, 8X etc). [2]

Since the wide usage of this method, in this report it will be used for comparison for some of the modern methods that has been explained previously.

SMAA

According to experimental tests done by Jimenez et al., MSAA 2x, 4x and 8x that is applied to a 1080p image has an average computation time of 1.57, 2.3 respectively 4.3 ms.
The same scene was rendered, but using SMAA 1x (single-pass), S2x (SMAA 1x + spatial multisampling), T2x (SMAA 1x + temporal supersampling) and SMAA 4x (SMAA 1x + spatial multisampling + temporal supersampling). It gave an average computation time of 1.02, 2.04, 1.32 respectively 2.34 ms.

The measurements was done on a NVIDIA Geforce GTX 470. [3]

The following table displays the computation time ratios between the SMAA and MSAA versions (SMAA / MSAA).

SMAA version	**MSAA 2x**	**MSAA 4x**	**MSAA 8x**
SMAA 1x	0.65	0.44	0.24
SMAA S2x	1.30	0.89	0.47
SMAA T2x	0.84	0.57	0.31
SMAA 4x	1.49	1.02	0.54

AXAA

According to tests that is presented by Jae-Ho Nah et al., AXAA performs 1.7x - 2.8x faster than SMAA (1x). These tests were run on four different platforms: Intel HD Graphics 4600, NVIDIA Geforce 840M, AMD Radeon 7650M and NVIDIA Tegra K1.
The AXAA computation times for a 1080p image, using the different platforms (same order as above) were 5.3, 2.5, 4.2 and 2.1 ms.
Same setup with SMAA 1x, computation times resulted in 10.6, 4.2, 9.6 respectively 5.9 ms.
AXAA compared to FXAA 3.11 were almost identical. [2]

The following table displays the computation time ratios between AXAA, FXAA and SMAA 1x on the different graphic platforms.

Anti-aliasing methods which time ratio is to be presented	**Intel HD Graphics 4600**	**NVIDIA Geforce 840M**	**AMD Radeon 7650M**	**NVIDIA Tegra K1**
AXAA / SMAA 1x	0.50	0.60	0.44	0.36
FXAA 3.11 / SMAA 1x	0.50	0.60	0.44	0.34

The average ratio (AXAA / SMAA 1x) resulted in 0.48.
The average ratio (FXAA 3.11 / SMAA 1x) resulted in 0.47.

These numbers tells that averagely, both AXAA and FXAA takes less than half of the work time of SMAA 1x, which is the simplest and fastest version of SMAA.

In the SMAA subsection above, the test results shows that the (SMAA 1x / MSAA 2x, 4x and 8x) calculation time ratio was 0.65, 0.44 respectively 0.24. If these ratios are applied to the other ratios from the AXAA test results, the following table can show the rough comparisons of AXAA, FXAA and MSAA.

Anti-aliasing algorithm	**MSAA 2x**	**MSAA 4x**	**MSAA 8x**
AXAA	0.31	0.21	0.12
FXAA	0.31	0.21	0.11

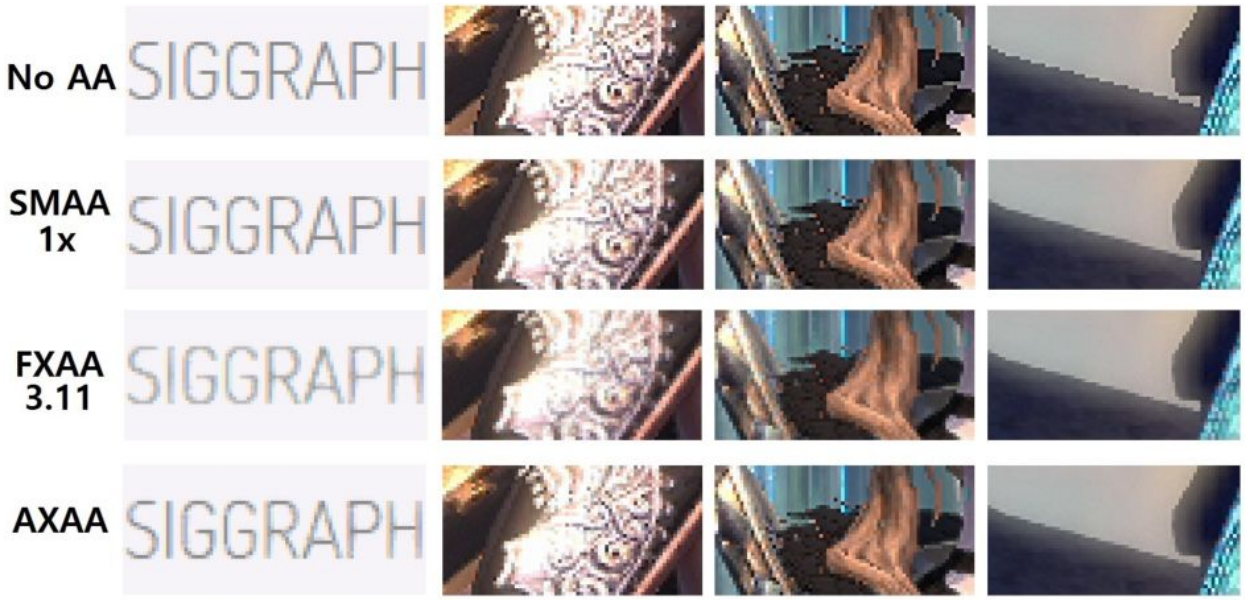
Even though the comparison uses measurement ratios from different hardware and images it still shows that AXAA and FXAA are the fastest anti-aliasing techniques that has been presented in this report.

Image quality

Image quality in the case of speed and performance is also very important to the user. In addition to faster rendering, it's important that what is observed on the screen has a representative image, which is the entire point of doing anti-aliasing in the first place.

Looking at Jae-Ho Nah et al. and their comparisons with AXAA and FXAA, they suggest that because of the three big improvements to the FXAA algorithm, their algorithm will increase the visibility of textures. The main reason for this comes from the early-exit criterion, which prevents duplicate anti-aliasing of the same pixel. By conserving the contrast of thin geometry pixels, it will also render thin geometry better and increase the readability of fonts. Doing these new things, it's able to produce a sharper image while still keeping the same performance as FXAA as proclaimed in the previous subsection. [2]

The following figure is a comparison between AXAA, FXAA and SMAA, done by Jae-Noh et al.



It is clear when looking at the images that all three algorithms does a good job of

removing aliasing. FXAA however, does excessive blurring due to the duplicate anti-aliasing that occurs. It is also clear, when looking at the figure, that this has been somewhat remedied with the changes to FXAA with AXAA and the addition of the early-exit criterion to skip already filtered pixels. The lines are sharper and there are not much blurring on the thin lines in the second and the third image. The SIGGRAPH fonts are, very clearly, extremely blurred with FXAA, but with AXAA it has kept some of its sharpness and improved the visibility. All three algorithms does a good job of smoothing the edges in the fourth image.

Comparing between SMAA and AXAA, there aren't a lot of distinction between the two, unless the pixels are looked at closely. AXAA almost does as good of a job as SMAA, with some minor differences. In the SIGGRAPH font image, one can clearly see that the blurring and the sharpness is more balanced with SMAA, than with AXAA. In the other images it is also noted that AXAA produces an image with slightly more contrast than SMAA, however it is unnoticeable when looked at from a larger distance. Unlike FXAA, the details are kept. The authors have expressed their concerns in some issues that may still come up since this is still a very new algorithm and that they will work more to improve image quality by combining the algorithm with multi-sampling and temporal super-sampling, such as SMAA has done. [2]

AXAA keeps the same performance as FXAA has, and from the results of the experimental tests that Jae-Ho Nah et al. has demonstrated, AXAA is close to 2-3 times faster than SMAA x1, while still producing a very similar image.

Discussion

Upon researching the different PPAA-algorithms, it has occurred to us that there are a lot of different ways to create an algorithm and different ways to make an existing algorithm better by using other algorithms that exists.

In the case of AXAA, it is mostly based on the already existing FXAA-algorithm, which was created by Timothy Lottes in NVIDIA in 2011. This was five years ago and during 2016, a team of people (Jae-Ho Nah et al.) presented an improved version of this algorithm, which produces better images for the same cost, AXAA. It is still experimental and improvements have been promised by the authors, so that it may produce better images for different situations. This may very well be an indication that this algorithm might be the next big algorithm in the market for low-performance renderers, and will possibly take over FXAA as the fastest algorithm in a few years.

Since there is only one paper of AXAA and it is very new in the field of computer graphics, there aren't a lot comparisons out there between other algorithms. For this reason, it was very difficult to write anything about AXAA in larger extensions than what was done in this paper.

When looking at the performance comparison of AXAA, FXAA and MSAA that was stated in this report, one should take into account that the resulting ratios are based on different tests that included different platforms. If the anti-aliasing methods instead were tested on a single platform with the same equipment and setup, these ratios possibly could differ.

Literature discussion

When surveying for literature, we wanted to find as new publications as possible, since we decided to write about different AA-methods, especially new modern ones. When looking for the literature, the DiVA database was used, as well as looking for publications on siggraph.org.

Some of the keywords that we used when searching: “AA”, “FXAA”, “anti-aliasing”, “anti aliasing algorithms”, “anti-aliasing 2016”, “computer graphics anti-aliasing”, “modern anti-aliasing”.

From these searches we were able to filter out publications that we felt were good and relevant for what we wanted to write about. Even though two of the publications are from 2011-2012, we thought it was modern enough for this subject. SMAA and FXAA have been around for a long time and almost all games use these methods for anti-aliasing. AXAA however is one of the newest methods and we thought it was a good opportunity to take a look into how it works and what differences we can find from FXAA and SMAA.

References

[1] Timothy Lottes, NVIDIA. *Filtering Approaches for Real-Time Anti-Aliasing*. 2011.

[2] Jae-Ho Nah, Sunho Ki, Yeongkyu Lim, Jinhong Park & Chulho Shin. LG Electronics. *AXAA: Adaptive approXimate Anti-Aliasing*. SIGGRAPH, 2016.

[3] Jorge Jimenez, Jose I. Echevarria, Tiago Sousa & Diego Gutierrez. Crytek GmbH. *SMAA: Enhanced Subpixel Morphological Anti-Aliasing*. Eurographics (Number 2, Volume 31), 2012.