UNIVERSITY OF SÃO PAULO

INSTITUTE OF MATHEMATICAL AND COMPUTER SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

SCC0251 – IMAGE PROCESSING

Alex Sander Ribeiro da Silva	9779350
Danilo Marques Araujo dos Santos	8598670

Implementing and Comparing Anti-aliasing Algorithms – Final Report

Prof. Dr. Moacir Antonelli Ponti

São Carlos
2019

Index:

1. Introduction

Anti-aliasing is a technique designed to add greater realism to a digital image by smoothing jagged edges on curved lines and diagonals. It is largely used in games. Aliasing manifests itself as jagged or stair-stepped lines (otherwise known as jaggies) on edges and objects that should otherwise be smooth.[1]

The objective of this project is to implement and compare different anti-aliasing algorithms regarding their effectiveness. One of the algorithms that we have decided to implement and analyze is the supersampling.

2. Input Images

Any image in the RGB space can be used by these algorithms, but we are mainly aiming at games. Therefore, most of the images are screen captures of the game itself. To provide better comparisons, multiple images of the same game are taken while looking at the exact same spot, avoiding any moving particles and disabling animations if possible. All the game settings are also maintained, except resolution and built-in AA.

Usually, three to five images are used per game:

- One at the target resolution and without any AA, either as a base for comparison or input for some of our algorithms.
- One at the target resolution and with built-in FXAA, if available, for comparison.
- One at the target resolution and built-in 4x MSAA, if available, for comparison.
- One at the target resolution and both built-in FXAA and MSAA, if available (which is rare), for comparison.
- One at a higher resolution, to be used as input for some of our algorithms.

[1] "What is Antialiasing? - Definition from Techopedia." https://www.techopedia.com/definition/1950/antialiasing. Accessed 27 May. 2019.

3. Algorithms Studied

3.1. Supersampling

Supersampling (SS) or postfiltering is the process by which aliasing effects in graphics are reduced by increasing the frequency of the sampling grid and then averaging the results down. This process means calculating a virtual image at a higher spatial resolution than the frame store resolution and then averaging down to the final resolution. It is called post-filtering as the filtering is carried out after sampling.[2]

The two steps in the post-filtering process are[3]:

1. Sample the scene at n times the display resolution. For example, suppose the display resolution is 512x512. Sampling at three times the width and three times the height of the display resolution would yield 1536x1536 samples.

2. The color of each pixel in the rendered image will be an average of several samples. For example, if sampling were performed at three times the width and three times the height of the display resolution, then a pixel's color would be an average of nine samples. A filter provides the weights used to compute the average.

We are implementing a simulated version of the Supersampling algorithm. It’s “simulated” because we are only implementing the second step of the process, which is the image processing part, and using images already sampled at higher resolutions as the input. This is done mainly because we aim to achieve anti-aliasing on already existent images, which could then be applied to, for example, games that do not have that feature available.

[2] "Antialiasing methods." https://web.cs.wpi.edu/~matt/courses/cs563/talks/antialiasing/methods.html. Accessed 28 May. 2019.

[3] "Overview of Aliasing in Computer Graphics: Part 2." 4 Oct. 1999, https://www.siggraph.org/education/materials/HyperGraph/aliasing/alias2b.htm. Accessed 28 May. 2019.

3.2. Multisample anti-aliasing

Multisample anti-aliasing (MSAA) is a type of spatial anti-aliasing, a technique used in computer graphics to improve image quality. "multisampling" refers to a specific optimization of supersampling.

In multisample anti-aliasing, if any of the multi sample locations in a pixel is covered by the triangle being rendered, a shading computation must be performed for that triangle. However this calculation only needs to be performed once for the whole pixel regardless of how many sample positions are covered; the result of the shading calculation is simply applied to all of the relevant multi sample locations.[4]

With MSAA, instead of sampling one pixel on a much larger scale, two or more adjacent pixels are sampled together while rendering an image at its intended size. Because multiple pixels are being sample together, coverage points can be shared between them.[5]

However, MSAA is an algorithm that requires more resources to be applied, so it is the worst choice in terms of compatibility.

The easiest option to solve this problem would be rendering the screen, at the same time, in two different resolutions. This would have a huge cost in performance, and would still involve some changes in drivers.

A more elaborate option would be to render pixels that are part of one edge at higher resolution, and others at lower resolution. Theoretically, this could result in good performance, but the cost would be to apply the edge detection algorithm on each frame, and only start building the next frame after that, which would also have a very large impact on performance.

[4] "Multisample anti-aliasing - Wikipedia." https://en.wikipedia.org/wiki/Multisample_anti-aliasing. Accessed 24 Jun. 2019.

[5] "How To Choose the Right Anti-Aliasing Mode for Your GPU - Tested." 22 Oct. 2010, https://www.tested.com/tech/pcs/1194-how-to-choose-the-right-anti-aliasing-mode-for-your-gpu/. Accessed 24 Jun. 2019.

3.3. Fast approximate anti-aliasing

Fast approximate anti-aliasing (FXAA) is the least demanding type of anti-aliasing. Rather than running complex calculation depending on the geometry and colors displayed, FXAA simply applies extensive blurring to obscure the jagged edges. The end result is unnoticeable performance impact but a generally blurrier image.[6] It is ideal for low-end computers.

The downsides are that textures may not appear as sharp if they are included in the edge detection, and it must be applied before rendering the HUD elements of a game, lest it affect them too.[7]

The processes of FXAA are listed as follows[8]:

1. Find all edges contained in the image: finding edges is typically a depth-aware search, so that pixels which are close in depth are not affected. This helps to reduce blurring in textures, since edges in a texture have similar depths.

2. Smooth the edges: smoothing is applied as a per-pixel effect. That is, there is no explicit representation of the edges. Rather, the first step is a depth-aware edge filter, which marks pixels as belonging to edges, and the second step filters the color image values based on the degree to which a pixel is marked as an edge.

.

[6] "What is Anti-Aliasing? [Simple Explanation] - GamingScan." 9 Apr. 2019, https://www.gamingscan.com/what-is-anti-aliasing/. Accessed 24 Jun. 2019.

[7] "Fast approximate anti-aliasing - Wikipedia." https://en.wikipedia.org/wiki/Fast_approximate_anti-aliasing. Accessed 24 Jun. 2019.

[8] "Fast approximate anti-aliasing - Wikipedia." https://en.wikipedia.org/wiki/Fast_approximate_anti-aliasing. Accessed 24 Jun. 2019.

4. Results obtained

4.1. Supersampling

We've coded our own simulation of a supersampling algorithm and have applied it to a 1440p image of a game, generating a 720p supersampled image in 1 minute and 37 seconds.

The figures 4.1.1., 4.1.2. and 4.1.3. show the original image (without anti-aliasing), the image generated by our algorithm and the image generated with the Fast Approximate Anti-Aliasing algorithm (FXAA) provided by the game:

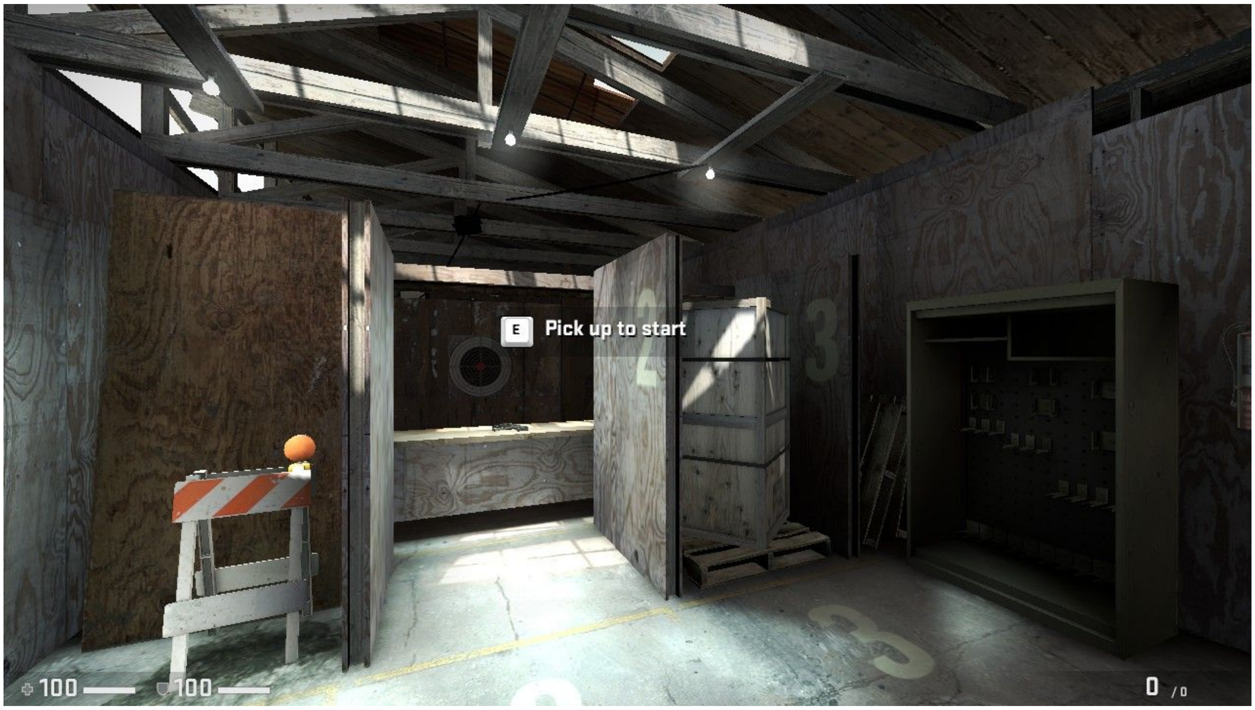

Figure 4.1.1. – original image. CSGO at 720p, no AA, captured from the game.

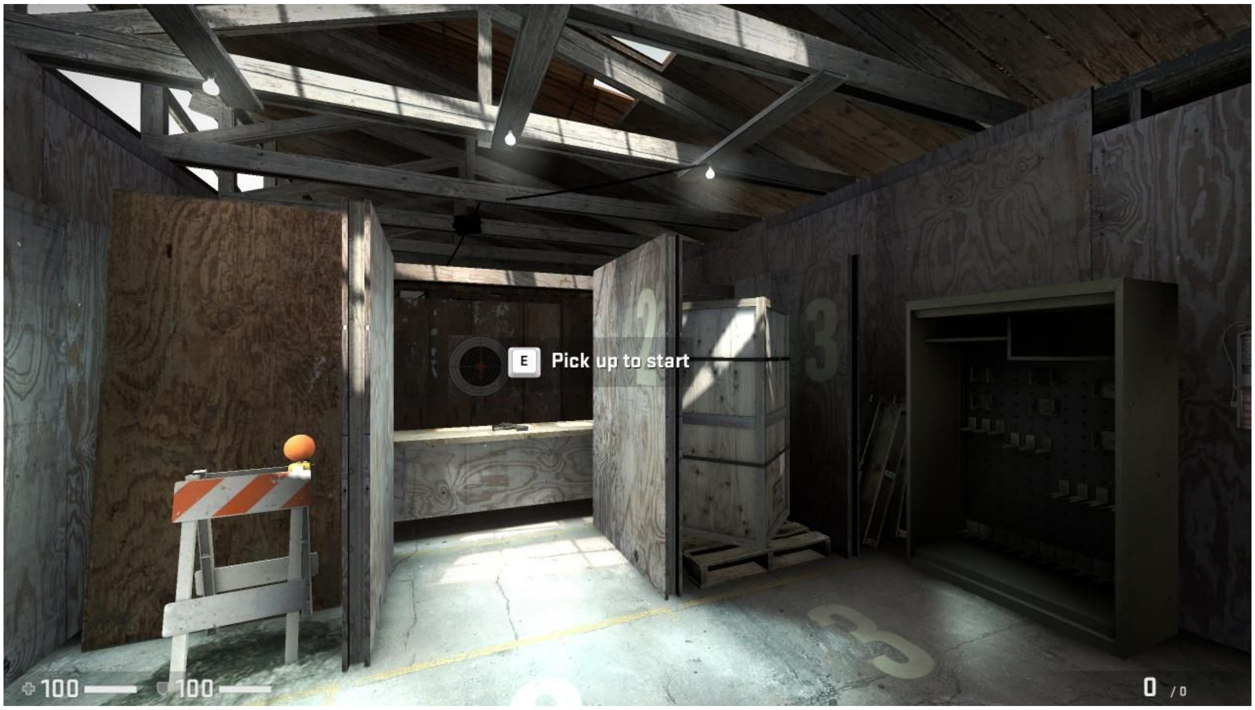

Figure 4.1.2. – image generated by using figure 4.1.1 as input for our SS algorithm (RMSE = 8.45).

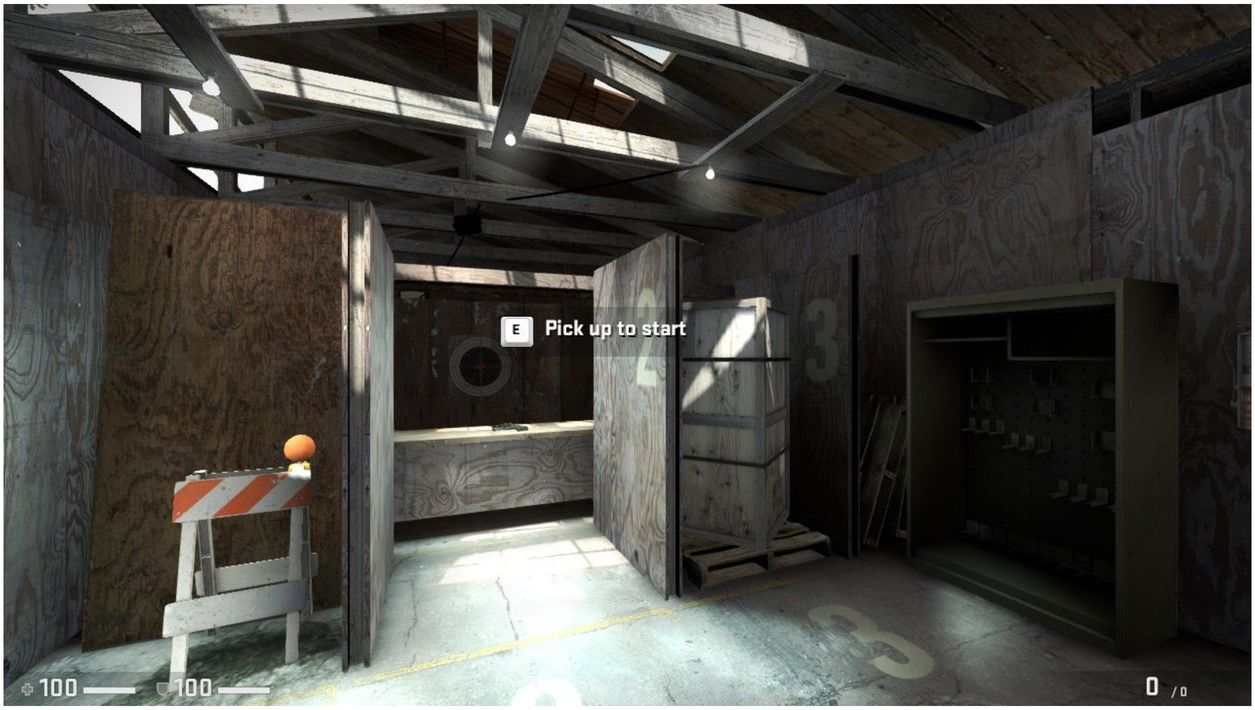

Figure 4.1.3. – comparison image. CSGO at 720p, built-in FXAA, captured from the game (RMSE = 5.012).

Of course, these images have a limited resolution on this page, but some differences can already be seen. The full sized images, as well as the results of two other games, can be found at https://github.com/dsantos-1/DIP_FinalProject/tree/master/Images.

4.2. Multisample anti-aliasing

We've coded our own simulation of a MSAA algorithm and have applied it to a 1440p image of a game, generating a 720p supersampled image in 1 minute and 32 seconds.

The figures 4.2.1. and 4.2.2. show the original image (without anti-aliasing), and the image generated with our implementation of the MSAA algorithm:

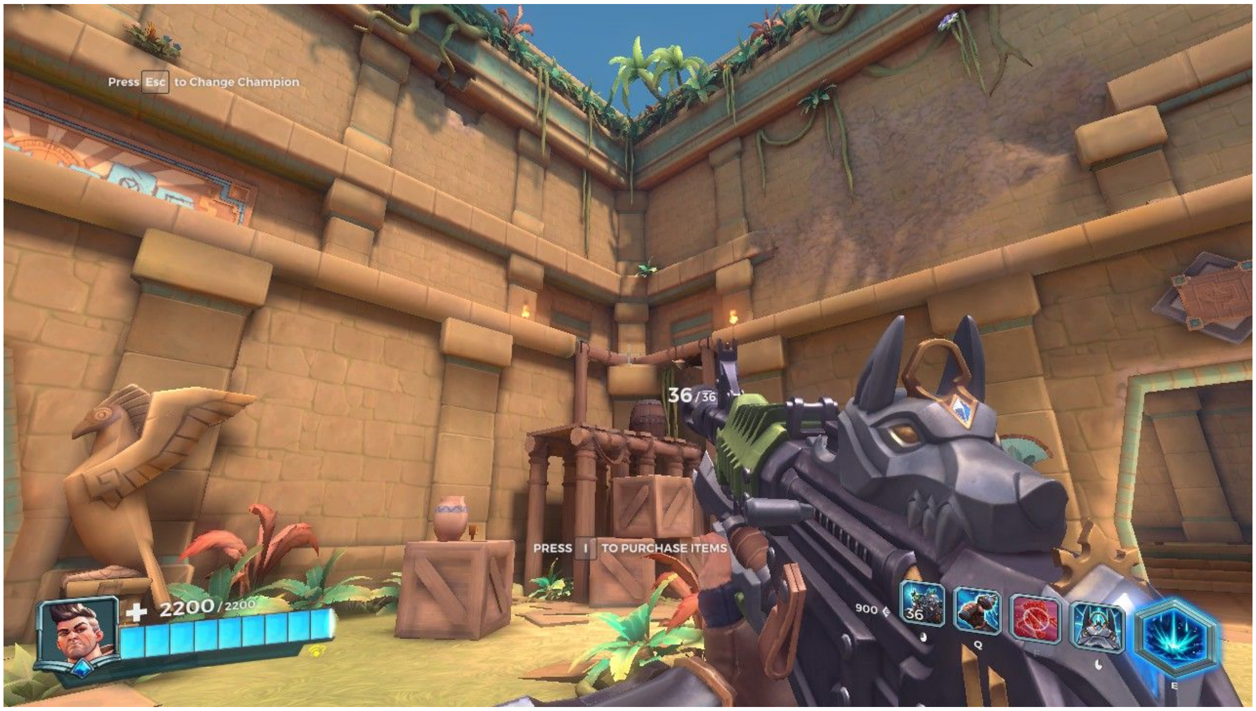

Figure 4.2.1. – original image. Paladins at 720p, no AA, captured from the game.

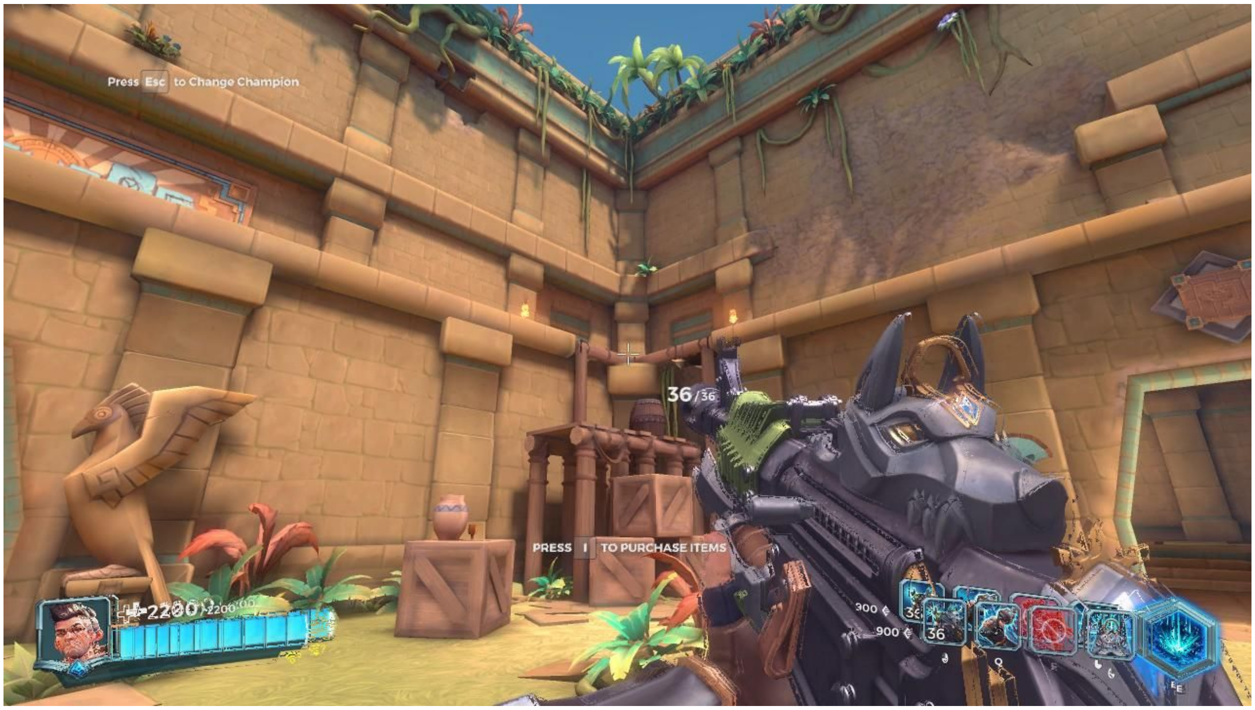

Figure 4.2.2. – image generated by using figure 4.2.1 as input for our MSAA algorithm (RMSE = 24.1627).

The full sized images, as well as the results of two other games, can be found at https://github.com/dsantos-1/DIP_FinalProject/tree/master/Images.

5. Conclusions

Despite achieving promising results in some regions of the image, using the MSAA through image processing is impractical in terms of performance and compatibility.