


Intra-Model Smoothing Using Depth Aware Multi-Sample Anti-Aliasing for Deferred Rendering Pipelines

B. M. Magnussen¹ 

¹FreeSpace 2 Source Code Project

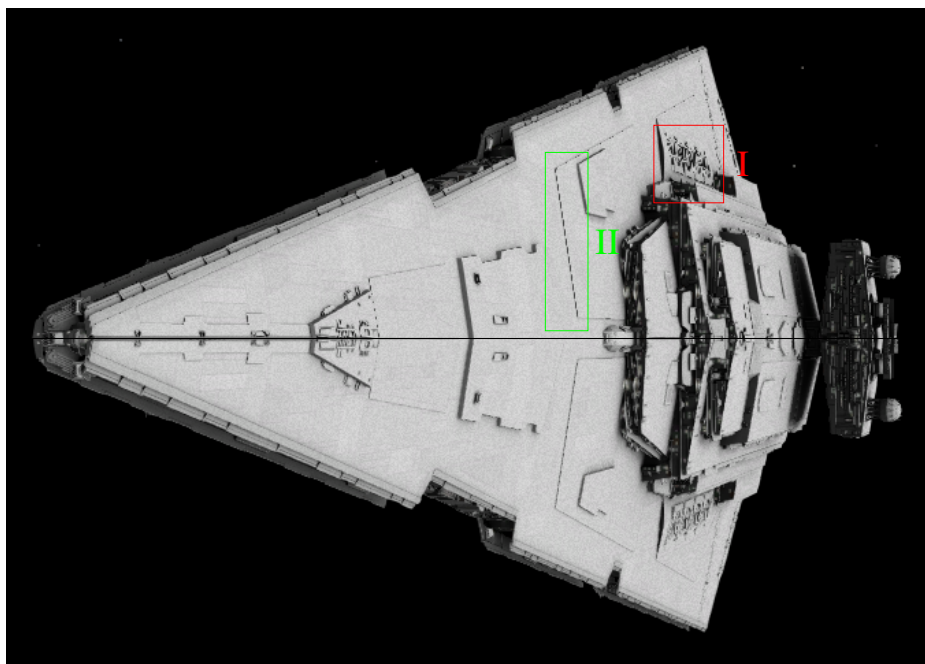


Figure 1: A community-created model by modeller Brand-X for the mod Fate of the Galaxy [Fat], rendered in FreeSpace Open [Fre23]. The top half shows the previous render pipeline, the bottom half has the proposed method enabled. Section I shows exaggerated geometry, and section II shows jagged geometry. The respective mirrored geometry rendered with the proposed method shows significantly reduced issues.

Abstract

Subpixel geometry often causes lighting artifacts. In some cases, post-process anti-aliasing algorithms are not sufficiently able to smooth the resulting image. For forward rendering pipelines, multi-sample anti-aliasing is a powerful tool to avoid such artifacts. However, modern rendering pipelines commonly use deferred shading, which causes issues for multi-sample anti-aliasing. This article proposes a new method of combining a pipeline using deferred shading with multi-sample anti-aliasing while avoiding common pitfalls. The proposed method achieves this by intelligently resolving the geometry buffers with a custom shader based on the depth of samples. This allows the lighting shader to run unchanged on the geometry buffer on a per-fragment basis without additional performance costs. Furthermore, the proposed method is easy to retrofit to existing engines as no changes are required to either the model rendering shader or the deferred lighting shader. The proposed method is demonstrated and implemented on the example of the open-source game engine FreeSpace Open. It is shown that the proposed method is capable of preventing subpixel geometry artifacts, while also avoiding lighting artifacts from resolving geometry buffers and avoiding the performance overhead of calculating lighting per sample.

CCS Concepts

• Computing methodologies → Antialiasing; Rendering;

1. Introduction

Many real-time 3D rendering applications use a technique called deferred shading [PF05] to improve the visual accuracy of the scene. In a pipeline using deferred shading, the scene is first rendered to a so-called geometry buffer (g-buffer) which contains information about the position, normal vectors, and depending on the pipeline other physical properties of the part of a model visible at each pixel of the screen. Only after the full scene is rendered to the g-buffer will the lit scene be rendered from the information available in the g-buffer. Because this causes the lit scene to not be rendered from the actual geometry, multi-sample anti-aliasing (MSAA, [PMH02]) cannot be used to directly render the final image. Using multi-sample anti-aliasing to render the g-buffer causes lighting artifacts. This is because when the multi-sampled g-buffer is resolved, the smoothing of single pixels which contain multiple different models will result in incorrectly blended values in the g-buffer [HH04]. Thus, techniques for performing anti-aliasing in screen space on the rendered scene during post-processing have been developed, such as fast approximate anti-aliasing (FXAA, [Lot09]) or enhanced subpixel morphological anti-aliasing (SMAA, [JESG12]).

Figure 1 shows an example rendered in the open-source FreeSpace Open [Fre23] engine, based on the game FreeSpace 2 [Vol99]. The image's top half is rendered using deferred shading and FXAA, and aliasing artifacts are still visible. These artifacts are caused by noisy, subpixel geometry details which result in aliasing artifacts in the normal buffer, as visible in Figure 2. Other, more advanced techniques for anti-aliasing, such as temporal anti-aliasing (TAA [YLS20]) may be able to improve upon the image quality but present a significant effort to be integrated into existing rendering pipelines, especially if not designed for methods such as TAA. This article proposes a new method of utilizing MSAA by resolving multi-sampled g-buffers manually using depth-based median weighting in order to reduce visual artifacts caused by noisy geometry while avoiding blending unconnected models and thus causing visual artifacts. The proposed method does not require any changes to either the model rendering shader or the deferred lighting shader, which allows for very easy retrofitting into any existing render pipelines using deferred shading. In addition, the proposed method will

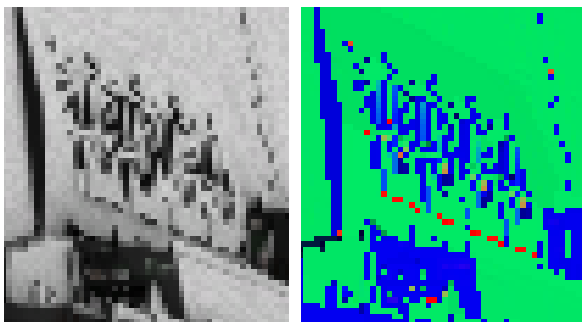


Figure 2: Closeup of section I from Figure 1. The right image shows the normal buffer. Exaggerated geometry lighting caused by aliasing in the normal buffer is clearly visible.

be implemented into and demonstrated on the FreeSpace Open engine, as can be seen on the bottom half of Figure 1, with the visible artifacts significantly reduced. The proposed method will be explained on regions of interest in Figure 3.

2. Related Work

One of the most common approaches for anti-aliasing with deferred shading is to use a post-process anti-aliasing technique, which analyzes visible discontinuities in screen space and applies smoothing respectively. One such implementation is FXAA [Lot09] which performs smoothing based on luminance edge detection. Another similar approach is SMAA [JESG12] which estimates the original geometry from edge detection and corrects the color of pixels on the edge accordingly.

More modern rendering pipelines also often use TAA [YLS20], which blends multiple subsamples of a pixel over multiple frames in order to achieve results similar to MSAA without the issues presented by utilizing MSAA with deferred rendering and without the added cost of rendering multiple samples per frame. However, TAA both introduces a very slight temporal lag and image smear, but more importantly is also more complicated and effort to integrate, especially if used in existing rendering pipelines which are not designed to utilize TAA.

Specifically in the context of deferred shading, it has been suggested to perform similar post-process anti-aliasing, improving the decision of which pixels to blend with the content of the g-buffer [HH04]. Especially utilizing the content of the position and the normal buffer is interesting for such an approach. In a similar manner, there has been research into combining such g-buffer enhanced post-process anti-aliasing with MSAA [CML11]. In this approach, the g-buffer is rendered using MSAA, however, shading is only performed on one of the samples. The post-process anti-aliasing pass however utilizes the available multi-sampled geometry data to improve its accuracy.

Another option to utilize MSAA with deferred shading is to perform the lighting pass per sample instead of per fragment [Thi09, HH04]. In this approach, the multi-sampled g-buffer is never resolved. Instead, it is the task of the deferred lighting shader to compute the final color for each sample, and then blend these to one pixel. While this approach makes use of the multi-sampled g-buffers, it causes the final shading to effectively run like supersampling, where every output pixel is required to be shaded for each possible sample. This drastically increases the computational cost, even if many pixels would not benefit from multisampling. To alleviate this, it is possible to store information about which pixels may require shading from multiple samples, and for which one sample may suffice [NVI]. In this approach, if a pixel consists of samples of only the same polygon, the deferred lighting shader will only compute the lighting once for the pixel, instead of once for each sample. Alternatively, it is possible to utilize information about screen space discontinuities from the g-buffer to decide whether or not a pixel needs to be shaded from all or just from one sample.

Optimization of especially memory usage and memory bandwidth utilization of similar per-sample shading strategies has been

researched as well, as memory usage and memory bandwidth have often been bottlenecks for per-sample shading [FT20].

Utilizing shaders to resolve multi-sampled buffers has been previously investigated [Pet12]. However, the research was focused on resolving color buffers using different types of blending kernels and was not applied to deferred shading. In addition, it was not extended to perform selective blending as is required for the method proposed in this article.

Similarly, utilizing the world position stored in the g-buffer as a source for heuristics has been investigated in the context of noise filtering the g-buffer [MIGMM17]. Application or usage in the context of MSAA has not been studied, however.

3. MSAA in Deferred Render Pipelines

At the core of the problem of utilizing MSAA to render the g-buffer and then resolve it to perform deferred shading per pixel instead of per sample, is that in some cases, blending multiple position or normal samples results in nonsensical values, causing incorrect lighting [NVI]. This can be seen in Figure 4, which is a region of interest of Figure 3. The left image shows the render without any kind of multisampling. The center render is the result of multisampling when rendering to the g-buffer, resolving the g-buffer, and then performing the final shading on the resolved g-buffer. Noticeably, the left edge of the model has a blue highlight. This is caused by a combination of two things. First, a short distance behind the model is a light source. Secondly, on the edge of the model, the position and normal of the model are blended with the position and the normal of the background. As a blended position of background and model results in a position behind the light source, the resulting pixel erroneously reflects the light source behind the model, thus being rendered as lit up. On the right is the result of rendering with the proposed method as introduced in this section.

In order to avoid the kinds of lighting artifacts as seen in Figure 4 when resolving the g-buffer, two samples may only be blended when they stem from the same object. If two samples stem from



Figure 3: A community-created model by modeller Nyctaeus for the FreeSpace Upgrade Project [Fre], rendered in FreeSpace Open [Fre23] in the previous render pipeline without the proposed method. Boxes I to IV highlight regions of interest.

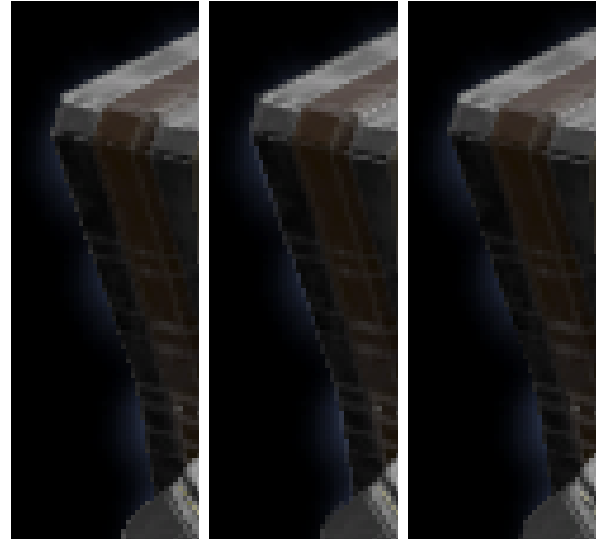


Figure 4: Closeup of section I from Figure 3. The left shows the image in the original pipeline without the proposed method, the center shows a naive MSAA implementation with incorrect edge lighting artifacts, the right image shows the proposed method.

two separated objects, or if there is significant space between the two samples, they must not be blended, and instead, one of the two must be chosen to represent the pixel.

This can be achieved using explicit multisampling, which was added in OpenGL 3.2 and DirectX 10. With explicit multisampling, it is possible to access the different samples of a multi-sampled render target directly from within a shader, not needing to previously resolve the multi-sampled texture. In the proposed method, two sets of g-buffers are created. One is multi-sampled, and the other is not. During rendering, the models are then rendered into the g-buffer using multisampling. Then, a custom multi-sample resolve shader is invoked by drawing a full-screen quad to the non-multi-sampled g-buffer. For each fragment, the shader will read all samples from the multi-sampled g-buffer, and blend them according to a weighted average according to their distance. Two samples close to one another can be blended equally. If the samples are far apart, one must be chosen over the other instead. As the distance of samples in the screen plane is always close to one another, given that the samples stem from the same fragment, the relevant distance between samples is given from the difference in their depth. For discussions on the requirements and different options for the weighting algorithm, see section 4. The resulting non-multi-sampled g-buffer can be utilized for the deferred shading step with per-fragment shading.

As it is shown in the upper images of Figure 5, without the proposed method, the g-buffer exhibits aliasing artifacts. The lower images of Figure 5 show, that with the proposed method, the intra-model aliasing artifacts (most notably, the red edge on the lower right of the image) are properly smoothed. At the same time, the proposed method does not incorrectly blend the blue-green edge on the top left of the image, as this edge is between two parts of the model with significant space between them. While this allows

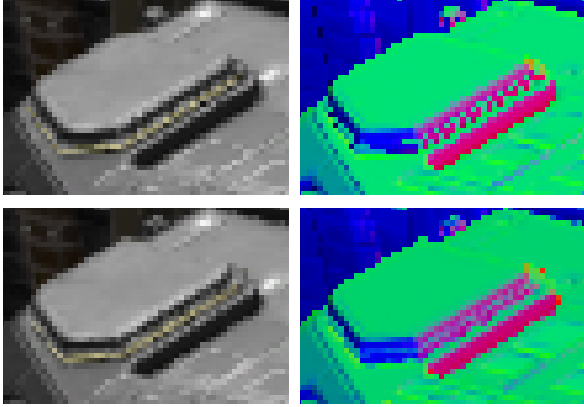


Figure 5: Closeup of section II from Figure 3. The top image set shows the previous pipeline without the proposed method, the bottom image set shows the proposed method. The right image is the normal buffer before the lighting pass. Normal aliasing is clearly visible in the previous pipeline.

proper anti-aliasing of intra-model geometry, it does not allow for anti-aliasing of outer model edges. Hence, while the visual quality of the model itself benefits, post-process anti-aliasing passes are still required to enhance the visual quality of the model edges.

Another advantage of the proposed method is the ease of integration into an existing render pipeline, as shown in Figure 6. While other methods, especially those with per-sample shading, would require changes to the model rendering shader and the main deferred lighting shader, the proposed method does not. This is because the required multi-sampling of the model render pass is done by the graphics library, and the deferred shading can happen identically as

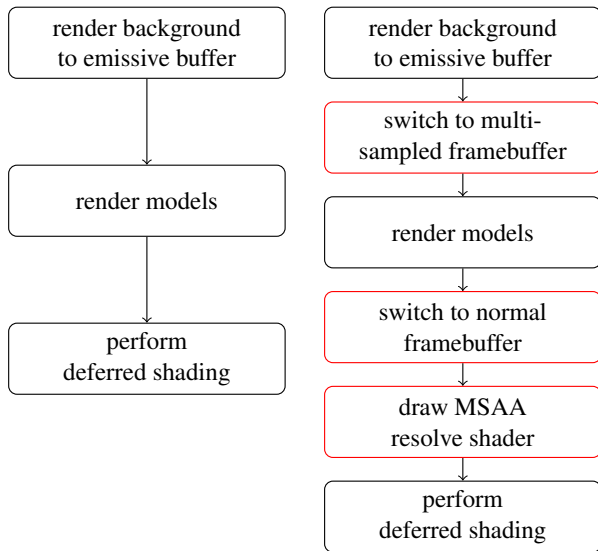


Figure 6: The graphics pipeline of FreeSpace Open before (left) and with the proposed method (right). New pipeline stages are red.

before per fragment once the g-buffer is resolved with the custom multi-sample resolve shader. The only required changes to a rendering pipeline for the proposed method are that the render target needs to be changed to be multi-sampled, and the custom multi-sample resolve shader needs to be invoked. Both the model rendering shader, and the deferred lighting shader can be left unchanged.

The proposed method has a notable performance cost compared to a native resolve of the MSAA buffer, especially when a tile-based GPU such as the Mali GPU [Har14] is used, as the multi-sampled texture texture has to be written back to memory instead of being able to be resolved per-tile. Implemented into the FreeSpace Open engine and tested on an NVIDIA RTX 2070, the additional g-buffer resolve pass takes 0.5 ms for four samples, and 1.1 ms for eight samples per fragment at a resolution of 1920×1080 , which is however an acceptable level of additional performance cost.

4. Multi-Sample Resolve Weighting Algorithms

The weighting algorithms used for the proposed method must fulfill three criteria in order to prevent rendering artifacts.

- Samples close in depth need to have similar weight
- Samples far away from each other in depth may not both have a high weight
- At least one sample must have a non-zero weight

The simplest possible algorithm that satisfies these criteria is to find the smallest depth of all samples, and then weigh a sample with 1 if it is close to this minimum depth, and with 0 if it is not. A slightly improved algorithm could replace the hard cutoff with a smoothstep, in which case we call it the front-first algorithm, as it fills the fragment with the frontmost samples. The definition of close can depend on the setting. If a common scale of objects is known in the context of what will be rendered, it is possible to define close in terms of the scale of details on the objects to be rendered. When this is not the case, for example with the model in Figure 3, close can be defined relative to the scale of a fragment. In the proposed method, two samples are considered close when their difference in depth is within a certain factor of the width of their fragment. Given a perspective projection matrix, this is the case if Equation 1 is fulfilled, with d_1 and d_2 being the respective depths, fov being the horizontal field of view angle, $width$ being the number of fragment columns over which the field of view will be rendered, and c being the aforementioned factor. Experiments show a factor of 5 to be a good compromise between artifacts and overly strict prevention of smoothing.

$$|d_1 - d_2| < \min(d_1, d_2) \cdot \tan\left(\frac{fov}{width}\right) \cdot c \quad (1)$$

Another option, as opposed to the front-first approach, is to invert the selection by finding the largest depth of a sample and assigning a weight of 1 to samples close to it and 0 to others. This algorithm, called back-first will fill the pixel with the sample that has the largest depth and is thus the furthest in the background.

In between those, this method suggests using a median algorithm, where samples close to the median sample depth are assigned a weight of 1. It is to note that this cannot be the true median, the mean between the two median values when an even number of

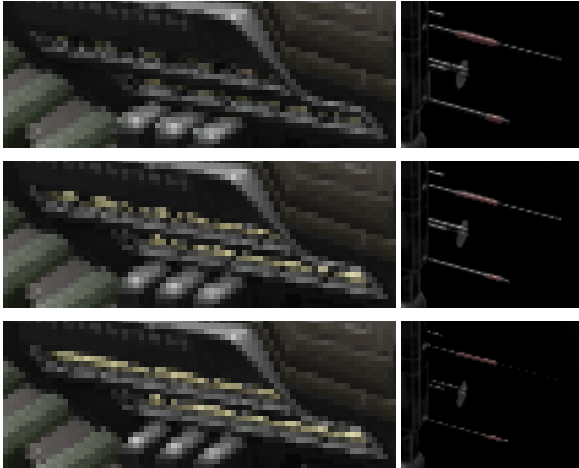


Figure 7: Closeup of section III and section IV from Figure 3. The top image set shows the front-first algorithm, the center image set shows the median algorithm and the bottom image set shows the back-first algorithm.

samples is available. Instead, it must be one of these two sample depths. It is not possible to use the mean instead of the median, as for a case with some samples with a high depth and some samples with a low depth, the mean would result in no sample being close, and all samples having a weight of 0, violating the third condition.

Figure 7 shows the result of applying the different algorithms introduced. As front-first will fill a pixel with the sample with the lowest depth, even if it just takes up one sample in the fragment, the foreground of images will appear up to one pixel thicker. This can be observed in the topmost images, where the antenna appears slightly thicker, and the yellow detail is almost completely obscured by the grey foreground. The opposite, back-first, will prioritize the sample with the highest depth, causing the foreground of images to appear up to a pixel thinner. In the lowermost images, the antenna appears noticeably thinner, and the yellow detail is almost completely visible behind the grey foreground. Median neither overly prioritizes foreground nor background, and thus results in a balanced look closest to the actual model dimensions. There was no significant performance cost for calculating the median of eight samples per fragment in the resolve shader. For this reason, the median algorithm is suggested for use with the proposed method.

5. Conclusion

In conclusion, this article proposes a method to perform multi-sample anti-aliasing for rendering pipelines using deferred shading. The proposed method is capable of reducing lighting artifacts from subpixel geometry, as seen in Figure 8, while avoiding artifacts caused by traditionally resolving multi-sampled g-buffers.

In addition, the proposed method is easy to retrofit into existing pipelines, as changes are neither required for the existing model rendering shader nor required for the deferred lighting shader. Furthermore, as lighting is still performed per fragment and not per sample, the proposed method is capable of avoiding the additional

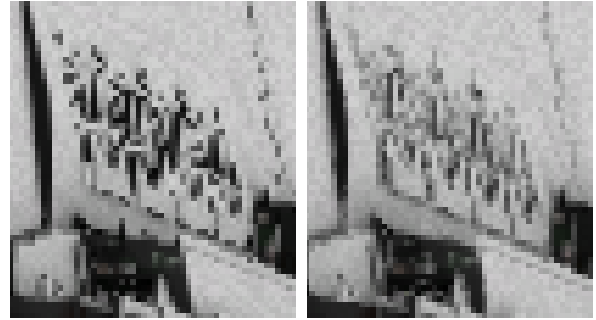


Figure 8: Closeup of section I from Figure 1. The left image shows the previous pipeline without the proposed method (including post-process FXAA), the right image shows the proposed method.

computational cost seen with previous solutions based on calculating lighting for multi-sampled g-buffers per sample.

References

- [CML11] CHAJDAS M. G., MCGUIRE M., LUEBKE D.: Subpixel reconstruction antialiasing for deferred shading. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, Association for Computing Machinery, p. 15–22 PAGE7. doi:10.1145/1944745.1944748. 2
- [Fat] FATE OF THE GALAXY:. Accessed 14 May 2023. URL: <https://www.hard-light.net/forums/index.php?board=143.0.1>
- [Fre] FREESPACE UPGRADE PROJECT:. Accessed 14 May 2023. URL: <https://www.hard-light.net/forums/index.php?board=120.0.3>
- [Fre23] FREESPACE 2 SOURCE CODE PROJECT:, 2002–2023. Accessed 14 May 2023. URL: <https://github.com/scp-fs2open/fs2open.github.com.1,2,3>
- [FT20] FRIDVALSZKY A., TÓTH B.: Multisample Anti-aliasing in Deferred Rendering. In *Eurographics 2020 - Short Papers* (2020), Wilkie A., Banterle F., (Eds.), The Eurographics Association. doi:10.2312/egs.20201008. 3
- [Har14] HARRIS P.: The mali gpu: An abstract machine, part 2 - tile-based rendering, 2 2014. Accessed 25 July 2023. URL: <https://community.arm.com/arm-community-blogs/b/graphics-gaming-and-vr-blog/posts/the-mali-gpu-an-abstract-machine-part-2---tile-based-rendering.4>
- [HH04] HARGREAVES S., HARRIS M.: Deferred shading, 2004. Accessed 12 May 2023. URL: http://download.nvidia.com/developer/presentations/2004/6800_Leagues/6800_Leagues_Deferred_Shading.pdf.2
- [JESG12] JIMENEZ J., ECHEVARRIA J. I., SOUSA T., GUTIERREZ D.: Smaa: Enhanced subpixel morphological antialiasing. *Comput. Graph. Forum* 31, 2pt1 (may 2012), 355–364. doi:10.1111/j.1467-8659.2012.03014.x. 2
- [Lot09] LOTTES T.: FXAA, 2 2009. Accessed 12 May 2023. URL: https://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf.2
- [MIGMM17] MOON B., IGLESIAS-GUITIAN J. A., McDONAGH S., MITCHELL K.: Noise reduction on g-buffers for monte carlo filtering. *Computer Graphics Forum* 36, 8 (2017), 600–612. doi:10.1111/cgf.13155. 3

- [NVI] NVIDIA CORP.: Gameworks library - graphics and compute samples - antialiased deferred rendering. Rev. 1.0.220830, Accessed 12 May 2023. URL: https://docs.nvidia.com/gameworks/content/gameworkslibrary/graphicsamples/d3d_samples/antialiaseddeferredrendering.htm. 2, 3
- [Pet12] PETTINEO M.: Experimenting with reconstruction filters for msaa resolve, 10 2012. Accessed 10 May 2023. URL: <https://therealmjp.github.io/posts/msaa-resolve-filters/>. 3
- [PF05] PHARR M., FERNANDO R.: Deferred shading in S.T.A.L.K.E.R. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005, ch. 9. 2
- [PMH02] PETERSON J., MULLIS R., HUNTER G.: Multi-sample method and system for rendering antialiased images, 10 2002. US Patent Application US09/823,935. 2
- [Thi09] THIBIEROZ N.: Deferred shading with multisampling anti-aliasing in directx 10. In *ShaderX7: Advanced Rendering Techniques*. Charles River Media, 2009, ch. 2.8. 2
- [Vol99] VOLITION, INC.: Freespace 2, 1999. source code published on 25 April 2002. 2
- [YLS20] YANG L., LIU S., SALVI M.: A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* (2020). doi:10.1111/cgf.14018. 2