Vol. [VOL], No. [ISS]: 1–3

The Compact YCoCg Frame Buffer (Supplemental Material)

Pavlos Mavridis and Georgios Papaioannou

Department of Informatics, Athens University of Economics & Business

1. Discussion on Antialiasing

One interesting idea is to take advantage of the multisample buffer format and store in the same pixel both chroma-orange and chroma-green samples, at different sub-pixel positions, effectively applying the mosaic pattern at the sub-pixel level. Although this idea sounds promising, we have rejected it for two reasons. First, the increased variance on the sub-pixel samples (each pixel will contain both YCo and YCg) would reduce the effectiveness of the underlying hardware lossless compression scheme. Furthermore, an implementation on existing GPUs would require the complete fragment shader to be executed at least two times for each pixel, one to write the YCo samples and one more for the YCg samples (assuming the hardware suports write masks in MSAA buffers). This is effectively supersample rasterization, which is prohibitively expensive for most real-time applications and scales linearly with the number of per-pixel samples.

We should also note that when MLAA [Reshetov 09] or similar post-processing antialiasing algorithms are used, then the antialiasing can be applied only on the luminance channel of the frame buffer. In particular, any bandwidth intensive post-processing algorithm can benefit from chrominance sub-sampling, as briefly discussed in [White and Barre-Brisebois 11]. This is another area that benefits from our method.

1086-7651/06 $0.50 per page

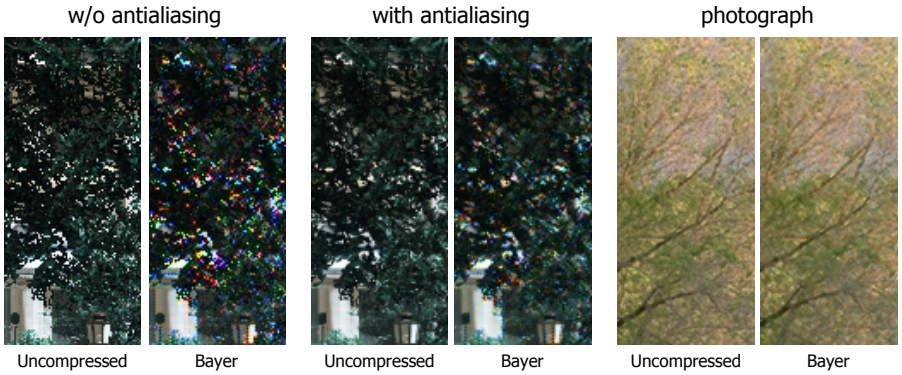

Figure 1. Left: Bayer encoding of a frame buffer without any antialiasing. Middle: Bayer encoding of the same frame buffer with antialiasing (SSAAx16). Right. Bayer encoding of a photograph (kodim22 form the kodak image suite).

2. Discussion on the Bayer Encoding Format

As explained in the main article, we have found that this format is not robust enough to be used for frame buffer compression in real-time rendering. In Figure 1 we examine the reasons of this failure. We observe that the chrominance artifacts are intense when encoding a frame buffer without any antialiasing, but they become much softer when anti-aliasing is used (SSAAx16). When applying the same encoding on a photograph with similar content, we observe that the algorithm works remarkably well, whithout any visible chrominance noise. The reason for this is that real-time rendering is still plagued with high frequencies and aliasing artifacts, due to various simplifying assumptions (pin-hole cameras, point lights, poor filtering, etc.). On the other hand, photographs caputered by real-world lens systems exhibit much lower frequencies. Based on the above observations, we conclude that the single channel bayer encoding is not suitable for general use in real-time rendering. On the other hand, our less aggressive 2-channel YCoCg encoding handles the high frequency color variations without any major artifacts.

3. Deferred Lighting Background

In order to reduce the heavy computational load caused by the shading computations, many interactive applications use the so called *deferred shading* pipeline. In this pipeline, the scene is rendered once, writing the normal vector, diffuse color, specular color, and specular spread factor, among other things, into a deep frame buffer or G-Buffer. In subsequent passes the shad-

ing of the pixels is calculated by reading the G-Buffer channels as textures and evaluating the shading equation for each light. The results are accumulated into a high precision accumulation buffer, using additive blending. The transparent surfaces are rendered in a separate forward pass.This approach reduces the computational load of the GPU, but increases the bandwidth consumption, since all the G-Buffer channels should be read for each light.

As the balance between available computation power and bandwidth has shifted towards the first, most of the modern game engines use a variation of this approach, which reduces the bandwidth requirements. First the scene is rendered writing only the normal and specular spread factor into a buffer. A subsequent pass calculates only the lighting of the pixels, and not the full shading equation as before, and accumulates the lighting results in two high precision accumulation buffers, one for the diffuse and one for the specular lighting. The final pass renders the scene again, reading the diffuse and specular accumulation buffers from textures, modulating them with the diffuse and specular colors of the surfaces and writing the end result into the final color buffer. This approach is commonly called *deferred lighting*, and was used in many modern video games (Naughty Dog's Uncharted: Drake's Fortune, Insomniac's Resistance 2, Crytek's Crysis 2, amongst many other). In order to reduce the consumed bandwidth and storage, most implementations of this pipeline use one 4-channel render-target to store the diffuse and specular accumulation buffers, by completely dropping the specular chrominance information, leading to incorrect specular highlights.

Naty Hoffman has a more in-depth analysis of the deferred pipelines online (http://www.realtimerendering.com/blog/deferred-lighting-approaches/).

References

[Reshetov 09] Alexander Reshetov. "Morphological Antialiasing." In *Proceedings of the 2009 ACM Symposium on High Performance Graphics*, 2009.

[White and Barre-Brisebois 11] John White and Colin Barre-Brisebois. "More Performance! Five Rendering Ideas from Battlefield 3 and Need For Speed: The Run." In *ACM SIGGRAPH 2011: Advances in the real-time rendering course*. ACM, 2011.