Z^2 Traversal Order for VR Stereo Rendering on Tile-based Mobile GPUs

Jae-Ho Nah, Yeongkyu Lim, Sunho Ki, and Chulho Shin*
LG Electronics

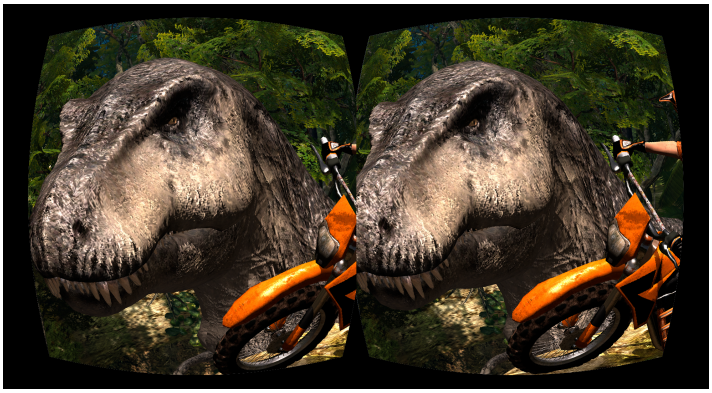

Figure 1: *A captured image of VR stereo rendering in the GFXBench T-Rex scene. By exploiting a similarity between the images of the left and right views, our novel tile traversal order can decrease memory bandwidth requirement for texture mapping by up to 30% and can increase texturing performance by up to 8% when rendering the scene.*

Abstract

With increasing demands of virtual reality (VR) applications, efficient VR rendering techniques are becoming essential because VR stereo rendering requires increased computational costs to separately render views for the left and right eyes. To reduce the rendering cost in VR applications, we present a novel traversal order for tile-based mobile GPU architectures, called the Z^2 traversal order. In tile-based mobile GPU architectures, a tile traversal order that maximizes spatial locality can increase the GPU cache efficiency. For VR applications, our approach improves the traditional Z-curve order; we render two screen tiles in the left and right views by turns or simultaneously, as a result, we can exploit spatial locality between the two tiles. To evaluate our approach, we conducted a trace-driven hardware simulation using Mesa and a hardware simulator. The experimental results show that the Z^2 traversal order can reduce external memory bandwidth requirements and can increase rendering performance.

Keywords: tile traversal order, tile-based GPUs, virtual reality, stereo rendering

Concepts: •**Computing methodologies** → **Graphics processors; Virtual reality;**

*e-mail: {nahjaeho, postrain70}@gmail.com, {sunho.ki, chulho.shin}@lge.com

SIGGRAPH Asia 2016 Technical Briefs, December 5-8, 2016, Macao
ISBN: 978-1-4503-4541-5/16/12
DOI: http://dx.doi.org/10.1145/3005358.3005374

1 Introduction

Recent technical advances of head-mounted displays (HMDs) and GPUs have brought an explosion in the virtual reality (VR) market. As a result, a wide range of VR applications has been developed: Games, 360° video, simulations, social media, and so on. In these VR applications, immersive visual experiences are very important. Thus, VR devices usually need to provide a high resolution screen (e.g., FHD to UHD) with high refresh rates (e.g., 60-120Hz). Additionally, HMDs require separately rendered images of left and right eyes, and this stereo VR rendering can increase the number of draw calls by up to twice. Therefore, efficient VR rendering techniques are required for realistic VR experiences.

We focus on efficient GPU hardware architectures for VR applications. Currently most mobile GPUs (e.g., Qualcomm Adreno, ARM Mali, and Imagination Technologies PowerVR) are based on tile-based GPU architectures to minimize off-chip memory accesses. These architectures divide the entire screen into multiple tiles by adding the tiling stage between vertex and fragment shading, and each shader core performs fragment shading of geometry in each screen tile with a tile buffer. Because the number of tiles is much higher than the number of shader cores, there can be various tile traversal orders and are needs for efficient tile traversal orders which increase rendering performance by utilizing spatial locality. That also means there is a room to improve VR rendering performance by choosing a tile traversal order dedicated to VR applications.

In this paper, we present a novel tile traversal order called the Z^2 traversal order. This traversal order is based on the traditional Z-curve order [Morton 1966], but we improve that for VR stereo rendering in two ways. The left-right tile assign version (Z^2 LRTA) assigns tiles in the left and right screens to a shader core by turns, and the simultaneous tile access version (Z^2 STA) assigns two tiles in the left and right screens to a shader core simultaneously and performs interleaved access to the primitive lists of the two tiles. Thanks to similarity of the left and right screens, the Z^2 traversal order can increase spatial locality compared to traditional traversal orders. In the point of view of simultaneous rendering of the

left and right views, the idea of our traversal order was inspired by Hasselgren et al.'s multi-view rasterization architecture [Hasselgren and Akenine-Möller 2006]. However, there is a difference between them; Hasselgren et al. focused on how to efficiently rasterize triangles to multiple views, in contrast, we focus on how to efficiently map shader cores to screen tiles.

We have built a GPU simulation environment on Mesa 11.0.3 [Paul and Whitwell 2015] to evaluate the effect of the Z^2 traversal order. This simulation environment provides an analysis of texture cache access patterns. This analysis includes texture cache hit rates, memory bandwidth requirements for texture mapping, and the utilization of a texture mapping unit on the simulator. According to the experimental results on the GFXBench T-Rex scene (Figure 1), the Z^2 traversal order can reduce memory bandwidth requirements for texture mapping by up to 30% and can increase texturing performance by up to 8%, compared to the traditional Z-curve order.

2 Background and Related Work

2.1 Tile-based GPU Architectures and Tile Traversal Order

Tile-based GPU architectures have been widely adopted to bandwidth-limited mobile platforms. The detailed hardware architectures are differently implemented by each GPU vendor, but those architectures have a common feature: splitting the entire rendering stage by dividing the screen into small tiles and redistributing primitives into multiple shader cores using the primitive list in each tile. This redistribution is performed between vertex and pixel processing, so the tile-based GPU architectures are also known as the sort-middle architectures [Molnar et al. 1994]. Because depth and color accesses for fragment shading can be performed using a small tile buffer in the architectures, the architectures can reduce the amount of power-hungry DRAM accesses and that is why the architectures are suitable for mobile devices. A great overview of tile-based rendering is described in Harris [2014].

After the tiling stage, tiles and their primitive lists are distributed into multiple shader cores. If a tile traversal order is cache friendly, then it can increase rendering performance because GPU architectures usually include multi-level cache hierarchies. A scanline order is the simplest form, the Z-curve order [Morton 1966; Clarberg et al. 2013] is a more sophisticated form to increase spatial locality, and a zig-zag pattern [Ellis et al. 2015] is an alternative of the Z-curve order.

2.2 VR Acceleration Techniques

A brute-force approach for VR stereo rendering is to separately render scenes twice for the left and right views. This approach is simple but can increase the number of draw calls by twice. Thus, recent studies try to reduce the redundant CPU/GPU workload in that case. An alternative approach is shader multiview (also known as stereo instancing) [Reed and Sancho 2015; Wilson 2015; Johansson 2016]; by exposing a ViewID variable to shaders, a GPU can separately handle shader threads for each view without increased draw-call overhead. This method can be implemented on current generation GPUs with recent OpenGL/OpenGL ES extensions (GL_OVR_multiview2, GL_NV_viewport_array2, and GL_ARB_shader_viewport_layer_array). More aggressive, effective approach is shading reuse [Hasselgren and Akenine-Möller 2006; Reed and Sancho 2015]; by reusing fragment shading results on the left view for the right view, this approach can reduce fragment shading cost by up to half. However, it can degrade image quality because pixel values on the right view are approximately evaluated on the texture space and it is particularly problematic on view-dependent shading. Another approach is to broadcast draw calls across multiple GPUs [Reed and Sancho 2015; Vlachos 2016; AMD 2015; NVIDIA 2016]. This approach can utilize full power of multiple GPUs connected by the SLI or Crossfire interfaces.

Another research direction for VR rendering is to reduce the number of shaded fragments of each view. Vlachos [2015] presented a stencil mesh to cull hidden area after warping in advance. Foveated rendering [Guenter et al. 2012] is a gaze-contingent multi-resolution rendering technique. By using eye trackers, this technique lowers image quality in the periphery (outside of the fovea) to increase rendering performance. The image quality of the peripheral area can be improved by additional blur or sharpen post-processing [Patney et al. 2016]. NVIDIA multi-resolution shading [NVIDIA 2016] allows multiple scaled viewports in a single pass, as a result, the edges of the screen distorted by warping and lens distortion can be rendered at reduced resolution without apparent loss of image quality. It can also be used for fixed foveated rendering [Vlachos 2016].

3 Z^2 Tile Traversal Order

In this section, we describe our novel Z^2 tile traversal order. An important point when selecting the tile traversal order is how much increase spatial locality; if data from similar texture addresses are referenced again within a short period time, there will be a high possibility of maintaining the texture data in cache hierarchies.

Our observation is that the images of the left and right views in VR stereo rendering usually look similar as illustrated in Figure 1. This is because a same scene is rendered with two slightly different views. Therefore, if we are able to render two screen tiles in the left and right views by turns or simultaneously, it will increase cache locality. Our Z^2 traversal order utilizes this feature for VR stereo rendering. Expect for that, the traversal order in each view is fundamentally the same as the Z-curve order.

We introduce two different traversal orders as depicted in Figure 2: The left-right tile assign version (Z^2 LRTA) and the simultaneous tile access version (Z^2 STA). Z^2 LRTA traverses the tiles in the left and right screens by turns. When multiple shader cores share an L2 cache, this traversal order can increase the L2 cache hit rates by assigning different shader cores to the left and right screen tiles, respectively. This traversal order can be simply implemented on the Z-curve-order-based architectures without increase of hardware complexity.

In contrast to Z^2 LRTA, Z^2 STA fetches two tiles in the left and right screens simultaneously, so two triangle lists of the two tiles are passed to a single shader core. After that, triangles in the left and right triangle lists are rendered by turns; in other words, after a triangle in the left screen tile is rendered, a triangle in the right screen tile is rendered. If the two screen tiles consist of very similar triangles (e.g., regions in the far distance), Z^2 LRTA can increase not only spatial locality but also temporal locality because there is a high possibility of the same triangle is fetched again while the two tiles are rendered. However, this traversal order has a disadvantage compared to Z^2 LRTA; Z^2 STA requires double-sized tile memory because two screen tiles should be rendered concurrently, and this double-sized tile memory will decrease area efficiency of a GPU.

Note that both Z^2 STA and Z^2 LRTA require the use of the shader multiview techniques and extensions mentioned in Section 2.2. If the brute-force approach with duplicated draw calls is used, there is no clue how to obtain the geometry lists of the left and right views simultaneously. Thus, our traversal order can be enabled only if the

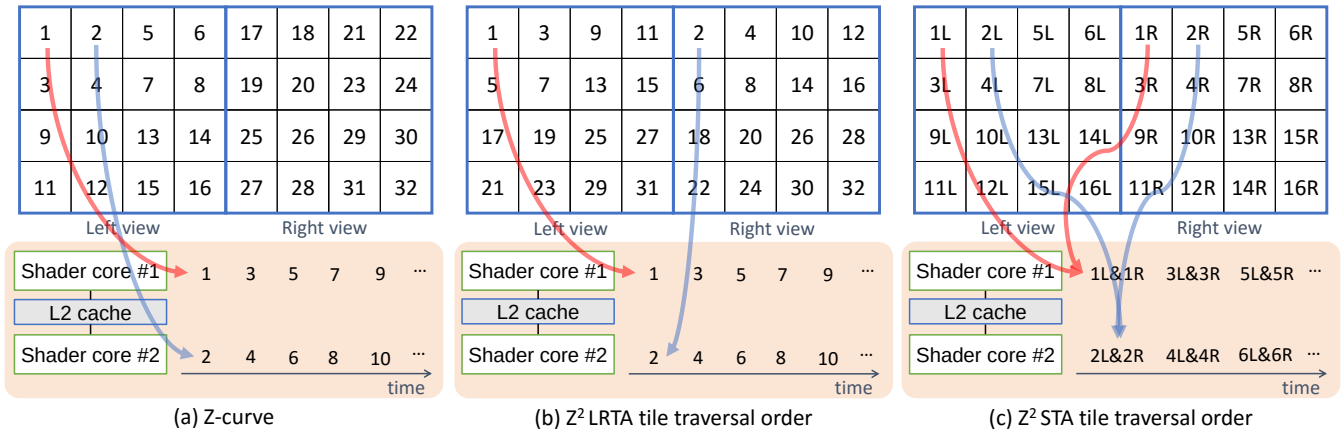

Figure 2: *Traversal order examples of 32 tiles in VR stereo rendering (top) and tile assignment examples between the tiles and a two-core GPU (bottom).*

tiling stage can sort all geometry of the left and right views together in a single render pass.

4 Experimental Results

4.1 Experimental Setup

We have built a texture-mapping simulation environment on Mesa 11.0.3 [Paul and Whitwell 2015] to verify the effectiveness of the Z^2 order. Because Mesa is a software OpenGL renderer based on immediate-mode rendering, we make a virtual grid to generate texture access traces on tile-based GPU architectures. After that, we store texture addresses into the corresponding grid cells whenever texture accesses occur. The size of a grid cell is 16×16 because 16×16 or 32×32 tile sizes are usually used in modern tile-based GPUs (e.g., Mali and PowerVR). After a frame is rendered, the trace file generated in the modified Mesa is fed to our in-house cache simulator in order to measure cache hit/miss rates and memory bandwidth requirements. We have also built a simple hardware simulator to measure utilization of a texture mapping unit (TMU). In this simulation, the texture pipeline is stalled when a cache miss occurs, and the miss penalty of L1 and L2 caches is 20 and 200 cycles, respectively.

For hardware configurations, we assume that two GPU shader cores share an L2 cache as illustrated in Figure 3; we believe this configuration is reasonable because usually two to four GPU cores are connected with an L2 cache in modern tile-based GPUs (e.g., Mali T600-T800 series and PowerVR 6-7 series). The size of each L1 cache is configured to 8 and 16 KB, and the L2 cache size is configured to 128 and 256 KB. Additionally, both caches are commonly configured to four-way set associativity with a 64-byte block size. If the two shader cores concurrently request the same memory address, we assume that the texture data from the requests are broadcast to the shader cores later.

The rendered scene is GFXBench T-Rex as depicted in Figure 1. For VR stereo rendering, we used Oculus VR library with a 100-degree field-of-view (FOV). All textures in the scene were compressed by DXT1. The screen resolution of each view is 960×1080.

4.2 Results and Analysis

Table 1 summarizes the experimental results. This table includes L1 cache miss rates, L2 cache miss rates, and TMU utilization.

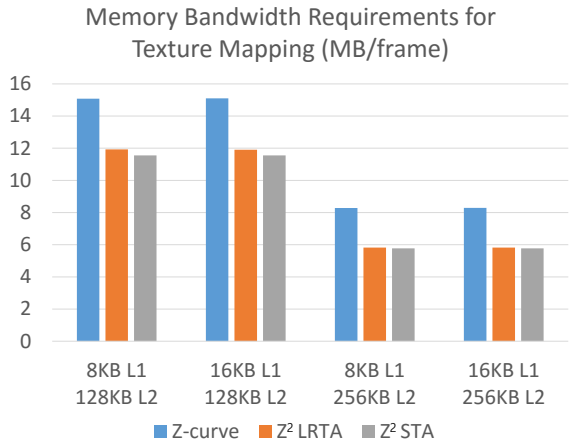

Figure 3: *Bandwidth comparison between the traditional Z-curve order, the Z^2 LRTA order, and the Z^2 STA order in the T-Rex scene. Lower values are better.*

According to the results in the table, Both Z^2 LRTA and Z^2 STA reduce L1 and L2 cache misses compared to the Z-curve order, as a result, TMU utilization also increases. In a texture-heavy scene, overall GPU performance is bounded by texture mapping performance. In this case, Z^2 LRTA and Z^2 STA can increase overall rendering performance by up to 7% and 8%, respectively. Regardless of the cache configurations, Z^2 STA always achieves slightly higher TMU utilization than Z^2 LRTA.

Figure 3 compares external memory bandwidth requirements of each tile traversal order. Note that the values in Figure 3 represent only bandwidth requirements for texture mapping because a tile traversal order is not directly related to off-chip memory bandwidth for fetching other types of data, such as geometry, depth, and color data. According to the results in the graph, Z^2 LRTA and Z^2 STA can reduce the memory bandwidth requirements for texturing by 21-30% and 23-30%, respectively. This is possible by decreasing miss rates of the L1 and L2 cache hierarchies.

The most optimal tile traversal order in a scene with a perfect coherence between the left and right views will not generate any cache misses in rendering the right view, and in this ideal case, a reduction in memory bandwidth requirements will be 50%. Because our sim-

Table 1: *Experimental results in the T-Rex scene. The traditional Z-curve order, the Z^2 LRTA order, and the Z^2 STA order are compared in this table, and lower cache miss rates and higher TMU utilization are better results. TMU is an abbreviation of a texture mapping unit.*

Cache Size		L1 Cache Miss Rate (%)			L2 Cache Miss Rate (%)			TMU Utilization (%)		
L1	L2	Z-curve	Z^2 LRTA	Z^2 STA	Z-curve	Z^2 LRTA	Z^2 STA	Z-curve	Z^2 LRTA	Z^2 STA
8 KB	128 KB	2.39	2.37	2.38	11.68	9.31	8.98	50.49 (1.00×)	53.43 (1.06×)	53.75 (1.06×)
16 KB	128 KB	1.70	1.65	1.61	16.46	13.39	13.27	54.27 (1.00×)	57.95 (1.07×)	58.59 (1.08×)
8 KB	256 KB	2.39	2.37	2.38	6.41	4.54	4.49	57.01 (1.00×)	59.94 (1.05×)	59.95 (1.05×)
16 KB	256 KB	1.70	1.65	1.61	9.03	6.55	6.63	61.89 (1.00×)	65.66 (1.06×)	66.03 (1.07×)

ple interleaved tile traversal orders achieve a memory-bandwidth reduction up to 30% in a common game-like scene, we believe that our approach is quite efficient.

5 Conclusions and Future Work

We have presented two variations of the traditional Z-curve order which are specially designed for VR stereo rendering. Z^2 LRTA traverses the left and right screen tiles by turns and is advantageous in terms of hardware complexity. Z^2 STA fetches the two tiles simultaneously in a single shader core and can maximize cache locality. We built a simulation environment on Mesa to evaluate the proposed tile traversal order, and the experimental results show that both approaches can decrease cache miss rates and can increase TMU utilization compared to the traditional Z-curve order. Because our tile traversal order can be easily implemented on existing tile-based GPU architectures and are orthogonal to other VR acceleration techniques, we believe it can be useful for a wide range of VR applications.

As future work, we would like to experiment more with various scenes and hardware architectures. Disparity manipulation techniqeus for specific graphics effects (e.g., gloss depiction [Templin et al. 2012]) will affect coherences between the left and right views, so we would like to analyze the effect of our approach in those cases. Additionally, we believe that our interaved Z^2 traversal order can be used in not only rasterization GPUs but also ray-tracing GPUs. In case that camera rays are shot using the Z-curve order in a ray tracer [Nah et al. 2014], our approach can be applicable to the ray tracer for VR extension.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments, Byeongjun Choi for his proofreading, and Jinhong Park for his good guide in the early stages of this work. GFXBench T-Rex is courtesy of Kishonti Ltd.

References

AMD, 2015. Virtual reality with AMD LiquidVR^{TM} technology. http://www.amd.com/en-us/innovations/software-technologies/technologies-gaming/vr.

CLARBERG, P., TOTH, R., AND MUNKBERG, J. 2013. A sort-based deferred shading architecture for decoupled sampling. *ACM Transactions on Graphics 32*, 4 (July), 141:1–141:10.

ELLIS, S., ENGH-HALSTVEDT, A., AND NYSTAD, J., 2015. Graphics processing systems. US Patent 9122646 B2.

GUENTER, B., FINCH, M., DRUCKER, S., TAN, D., AND SNYDER, J. 2012. Foveated 3D graphics. *ACM Transactions on Graphics 31*, 6 (Nov.), 164:1–164:10.

HARRIS, P., 2014. The Mali GPU: An abstract machine, part 2 - tile-based rendering. https://community.arm.com/groups/arm-mali-graphics/blog/2014/02/20/the-mali-gpu-an-abstract-machine-part-2.

HASSELGREN, J., AND AKENINE-MÖLLER, T. 2006. An efficient multi-view rasterization architecture. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, 61–72.

JOHANSSON, M. 2016. Efficient stereoscopic rendering of building information models (BIM). *Journal of Computer Graphics Techniques (JCGT) 5*, 3 (August), 1–17.

MOLNAR, S., COX, M., ELLSWORTH, D., AND FUCHS, H. 1994. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications 14*, 4 (July), 23–32.

MORTON, G. M. 1966. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York.

NAH, J.-H., KWON, H.-J., KIM, D.-S., JEONG, C.-H., PARK, J., HAN, T.-D., MANOCHA, D., AND PARK, W.-C. 2014. Raycore: A ray-tracing hardware architecture for mobile devices. *ACM Transactions on Graphics 33*, 5, 162:1–162:15.

NVIDIA, 2016. NVIDIA VRWorks^{TM}. https://developer.nvidia.com/vrworks.

PATNEY, A., KIM, J., SALVI, M., KAPLANYAN, A., WYMAN, C., BENTY, N., LEFOHN, A., AND LUEBKE, D. 2016. Perceptually-based foveated virtual reality. In *ACM SIGGRAPH 2016 Emerging Technologies*, 17:1–17:2.

PAUL, B., AND WHITWELL, K., 2015. The Mesa 3D graphics library version 11.0.3. http://www.mesa3d.org/.

REED, N., AND SANCHO, D. 2015. VR Direct: How NVIDIA technology is improving the VR experience. In *Game Developer Conference 2015*, GDC '15.

TEMPLIN, K., DIDYK, P., RITSCHEL, T., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2012. Highlight microdisparity for improved gloss depiction. *ACM Transactions on Graphics (SIGGRAPH 2012) 31*, 4 (July), 92:1–92:5.

VLACHOS, A. 2015. Advanced VR rendering. In *Game Developer Conference 2015*, GDC '15.

VLACHOS, A. 2016. Advanced VR rendering performance. In *Game Developer Conference 2016*, GDC '16.

WILSON, T., 2015. High performance stereo rendering for VR. San Diego Virtual Reality Meetup.