Design of Low Power Ray Triangle Intersection Accelerators

Krishna Rajan

Submitted on: 4-30-2019

Submitted in partial fulfillment of the requirements of the degree of Bachelor of Science *with Honors* in Electrical Engineering

School of Engineering, Brown University

Prepared under the direction of

Prof. Sherief Reda, Advisor

Prof. Jacob Rosenstein, Reader

By signing below, I attest that the undergraduate thesis listed above meets the criteria for Honors, and has been successfully presented to the faculty at the Undergraduate Research Symposium.

_____

Advisor's Signature

_____

Reader's Signature

BROWN

_____

Honors Chair's Signature

Contents

1. Abstract: 3

2. Introduction: 4

3. Previous Works: 7

4. Methodology 9

4.1 Precision Design Exploration 9

4.2 Duel Precision Methodology 12

4.3 Hardware Design 15

5. Results 17

5.1 Experimental Setup 17

5.2 Impact of Precision Scaling on Hardware Metrics 18

5.3 Evaluation of Duel-Precision Methodology 19

5.4 Hardware Cost Evaluation 22

6. Conclusions 24

7. Appendix 25

7.1 Sponza Scene Images at Fixed Point: Bit Precisions 28-40 25

References 29

1. Abstract:

Ray-Triangle intersection is a fundamental computation in most ray tracing algorithms. The prohibitive cost of the ray-triangle test algorithms, however, limits the utilization of these algorithms in settings with low power budgets, such as mobile systems. In this thesis, we analyze the precision requirements for ray-triangle intersection. We observe that for most rays, a low precision algorithm is sufficient, and only a small fraction of rays require higher precision computations. Accordingly, we propose a dual-precision hardware accelerator for ray-triangle intersection, targeting low-power systems. In this architecture, the higher resolution is activated only for tests deemed critical by our algorithm. Towards this goal, we develop a thresholding technique that autonomously switches between the lower and higher precisions. The lower precision unit is used for the majority of the tests, resulting in significant benefits in power consumption. We evaluate our methodology on a representative set of scenes, and implement our proposed methodology in hardware. The proposed methodology offers benefits of up to 86% in energy consumption compared to a baseline full-precision (floating-point) design.

2. Introduction:

Ray tracing is the basic algorithm with which movie special effects are created (Fascione, 2018), and is approaching the general consumer market with the recent advent of NVIDIA's RTX GPUs (NVIDIA, 2018), providing advanced effects for real-time games. It has also been shown to enable efficient algorithms for lens distortion, wide field of view, foveation, and depth of field for use in virtual and augmented reality (Hunt, July 2018). Making real-time ray tracing available for low power hardware will bring the richness of physically based rendering and efficient algorithms to mobile devices. The cost for providing higher accuracy and realism, however, is that ray-tracing algorithms require significantly higher computational costs, which limit their application under both tight power budgets and real-time applications. Such scenarios are increasingly abundant in both mobile and virtual/augmented reality platforms, which include wearable devices, with no external computation platform. A high-performance ray tracer requires optimization of several components, including ray traversal through acceleration structures to find intersections, and computation of intersection coordinates. There exists a long history of research into these real-time algorithms for software systems (Tomas Akenine-Möller, 2018) and research into potential dedicated hardware systems (Hanika & Keller, 2007) (Keely & Fussell, 2018) (Vaidyanathan, 2016). In real-time systems, triangles are the dominant surface primitive, and ray triangle intersection is the core operation that benefits most from custom hardware design (NVIDIA, 2018). In this thesis, we focus on reducing the power usage of the ray-triangle intersection computations, moving towards the goal of making ray tracing possible on low-power systems.

To enable the use of ray tracing algorithms under tight power constraints, we propose a dual-precision mechanism where an intersection test can be performed in either low or high

precision. This is dependent on the criticality of the intersection, which is classified using a thresholding scheme. Overall, the contributions of our work are as follows.

First, we perform a detailed and thorough evaluation of the precision requirements of ray triangle intersect algorithms. Using an industrial ray tracing package, we evaluate a large range of fixed-point arithmetic precisions, and compare the accuracy against the gold standard floating-point precision. From our evaluation, we observe that the majority of the ray-triangle tests can be properly classified with a reduced number of fixed-point bits. Only a minority of critical ray intersections require higher precision to resolve correctly. To provide realistic results compatible with the requirements of current applications, we modify HVVR (Hierarchical Visibility for Virtual Research), an open source industrial strength ray-tracing implementation, produced by Facebook (Facebook, 2017). HVVR by default performs its computations on a GPU in single-precision floating point; we modify it to utilize fixed-point computations, allowing for profiling and precision analysis. We also investigate the impact of reduced precision on hardware metrics.

Enabled by our evaluation analysis, we devise an analytical formulation that can identify the optimal low precision and high precision requirements to minimize total power consumption. Accordingly, we designs and implement a dual-precision ray triangle intersection accelerator, where the majority of intersections are performed in low precision and the higher precision is only activated when necessitated for accuracy.

To enable switching between low precision and high precision accelerator units during runtime, we formulate a thresholding methodology where the criticality of a test is assessed and the higher-precision unit is activated if necessary. Our methodology detects thresholds in which a delicate balance between accuracy and computation is struck.

We fully implement our proposed dual-precision architecture as a hardware accelerator using a full industrial design tool chain using 7 nm technology, and report the improvements in power consumption and other design metrics using our methodology.

The rest of the thesis is organized as follows: First in *Previous Work,* we briefly overview the recent work relevant to our study. Next, *Methodology* describes the precisions studied and the detailed discussion of the dual-precision methodology. *Results* summarizes our experimental results and highlights the benefits that this methodology provides. Here, we also provide the results obtained from the hardware accelerator. Finally, *Conclusions* concludes the thesis by summarizing the main components of the thesis.

3. Previous Works:

In recent years, many works have studied efficient realizations for ray-tracing algorithms on custom as well as traditional hardware (Keely S. , 2014) (Keely & Fussell, 2018) (Lee, 2013) (Keller, 2013) (Vaidyanathan, 2016). In these work, the underlying principle is either relying on software for performing the intersection tests (Keely S. , 2014), or on adding extra full-precision floating-point units (FPUs) (Lee, 2013) (Hwang, 2015).

Many studies have also investigated different approaches to reduce the power and computation demands of ray-tracing algorithms. Methodologies based on compression of triangle data structures (Cline, 2006) (Kim, Moon, Kim, & Yoon, 2009) (Mahovsky, 2006) (Segovia, 2010), reducing memory footprint using implicit indexing (Bauszat, 2010), reducing the number of stored planes per node (Eisemann, 2008) (Fabianowski, 2009), or transformations aiming at increasing floating-point precision (Kim, Nah, & Park, 2016) have been explored. Studies of reduced precision hardware accelerators have also been proposed (Hanika & Keller, 2007) (Heinly, 2009) (Vaidyanathan, 2016). While utilization of lower bit-width fixed-point arithmetic can significantly reduce the complexity as well as memory and computation footprint of the ray-tracing algorithm, careful consideration is required to minimize the accuracy impact. In contrast to these work, we propose a dual-precision methodology, where two different precisions are utilized within the same hardware accelerator. Enabled by the principal observation that the majority of the ray-triangle tests are correctly classified using reduced precision and lowering the precision only results in intersection error in *critical* cases. This key observation enables a scheme, in which the higher resolution circuitry is only enabled if the test is deemed *critical* by our thresholding methodology.

A hybrid approach has recently been proposed by Hwang *et al*. (Hwang, 2015) where both fixed-point and floating-point representations are utilized. Our proposed hardware differs from theirs as we utilize two lower-cost fixed-point precision instead of a fixed-point component along with a full-precision floating-point component. Furthermore, in our methodology the higher precision is only activated when required for accuracy. Next, in *Methodology*, we describe our methodology and hardware implementation in detail.

4. Methodology

4.1 Precision Design Exploration

While floating-point arithmetic is typically restricted to follow specific precision, namely, half-precision (16 bits), full-precision (32 bits), or double-precision (64 bits), fixed-point arithmetic offers the flexibility to customize the bit-width as required by the application. Furthermore, fixed-point computations are inherently less complex than their floating-point counterparts are. As such, fixed-point arithmetic provides a more fine-grain and efficient approach to computations as long as the operands do not exhibit a large dynamic range. Due to the larger design space of possible fixed-point precision, an effective exploration of the possible precision space is required to quantify, both the resulting accuracy and hardware metrics, as the precision is changed. In this work, we first examine the impact of precision scaling in ray-triangle intersection applications. This thesis both inspires the proposed dual-precision methodology and guides the choice of the higher precision (HP) and lower precision (LP) computing units.

To evaluate the impact of precision scaling in a realistic setting, we modify the well-established HVVR tool from Facebook Reality Labs (Facebook, 2017) to use fixed-point computations. The tool uses the Kensler (Kensler, 2006) approach to determining ray-triangle intersections. Through modification of the HVVR code, we explore the impact of reduced fixed-point precision on three scenes.

To decide the range of bit-widths required for our evaluations, through modification of the HVVR code, we perform a study of the numerical range of coordinates on a variety of 3D scenes. Here, first, we aim to quantify the minimum number of integer bits required to fully capture the scenes. We then fix the required number of integer bits for all the evaluated precision values, as

larger values do not result in improvements in accuracy, while lower values degrade the accuracy significantly. While there exists scenes, which span a larger numerical range, such scenes can be scale down to fit within a smaller range.

Similarly, in order to determine the upper bound of the precision design space, in our evaluations, we increase the number of fraction bits as long as such higher bit-widths result in higher test accuracy.

precision	Correct	Missed Hits	False Hits	Error (%)
(20,4)	4460217	2177591	32248725	88.53
(22,6)	4146136	840485	10186524	72.67
(24,8)	4460133	577123	2144162	37.89
(26,10)	4631573	428424	624543	18.52
(28,12)	4895939	173242	179073	6.71
(30,14)	5011778	59361	49909	2.13
(32,16)	5052235	18003	13957	0.63
(34,17)	5059811	10005	7397	0.34
(34,18)	5064065	5560	4003	0.19
(36,20)	5067989	1449	1030	4.89E-02
(38,22)	5069066	356	251	1.20E-02
(40,20)	5067990	1448	1049	4.92E-03
(40,24)	5069341	79	59	2.72E-03

Table 1: The impact of precision scaling on the accuracy of the Crytek Sponza scene.

Table 1 summarizes the results from our evaluations. Here, we report the results for the Crytek Sponza scene. We will provide more details about the scenes in Setup. In this table, (w,f) indicate the fixed-point bit width, and the number of fraction bits, respectively. In addition, we

report the number of correct test classifications, missed hits (hits that where misclassified as a miss due to reduced precision), and false hits (defined as the number of hits classified as a hit by mistake due to lower precision). Lastly, we also report error, which quantifies the percentage of wrong classifications in reference to a 32-bit floating-point design.

In our experiments, we observed that 16 integer bits, provides enough range to correctly capture the entire scene. Therefore, we fix the integer bits to equal 16, and explore the design space by increasing the number of fraction bits.

For fraction bits, we cover a large range starting from only four, and stopping when increasing the number of fraction does not increase the accuracy. In our evaluations, 26 fraction bits failed to increase the accuracy beyond what 24 fraction bits offered. As a result, in our design space exploration, we evaluate a range of bit-widths from 20 (16+4) and up to 40 (16+24) bits. Having a large number of fractional bits is essential in order to preserve depths accurately. If a triangle is directly behind another, such as a rug on a floor, then precision errors can make the underlying triangle appear on top otherwise.

The hit classification accuracy for these precision values range from an error rate of 88% for 20-bit fixed point to a negligible 0.003% for the highest precision. These results highlight the significance of fraction bits in rendering quality. Such a large range enables the dual-precision methodology described in the next section.

4.2 Duel Precision Methodology

From Table 1, we derive one of the main observation of this work, where we find that the majority of the ray-triangle tests can be correctly classified with reduced precision, whereas a minor subset of the tests here referred to as critical tests will be misclassified with reductions in precision.

For determining the criticality of a ray-triangle intersection test, we observe that the overwhelming majority of the misclassifications occur in cases where (1) the barycentric coordinates of the ray-plane hit is in proximity to one of the edges, or (2) when comparing the depths between two closely stacked triangles. The latter case is a result of the limited precision of the reciprocal operation required to compute depth, which amplifies precision errors in preceding computations. We observed that attempting to reduce the precision of depth calculations left too many rays misclassified, and made thresholding to determine their critically unfeasible. However, we observed that computation of the barycentric coordinates could be successfully thresholded and reran.

The most damaging barycentric coordinate errors are rays that hit triangles in full-precision implementation but miss with reduced precision. These errors can cause gaps between neighboring triangles and at corners. Algorithms that guarantee that this does not happen along shared edges/corners are known as *watertight*. To maximize the effectiveness of our threshold, we only classify rays as critical if a barycentric coordinate is both near zero and is *negative*. We do this rather than a symmetric threshold around zero to keep with the ultra-low power design principles. This thresholding scheme does not consider recomputing rays that could be potential *false positives*, as these generally are less damaging to the overall scene.

In this light, we propose a dual-precision architecture where for the dominant *non-critical* tests the low-precision computing units are utilized whereas the high-precision component is activated if a test is deemed *critical*. Here, we base the choice of the two precisions on the study performed as described in the previous section.

Specifically, after the computation of the intersection's barycentric coordinates (U, V, and W), we can threshold against value theta:

$$U + \theta > 0 \; and \; V + \theta > 0 \; and \; W + \theta > 0 \tag{1}$$

$$U < 0 \; and \; V < 0 \; and \; W < 0 \tag{2}$$

If both of these conditions are met, then the ray is determined to be critical. We determine theta empirically from scene characterization.

In our proposed thresholding scheme, the low precision unit will simultaneously compute the depths at the high fixed-point precision, while computing the barycentric coordinates using a lower precision. After computation, if the ray is determined to be critical, we rerun the barycentric coordinates at high precision at a cost of E_{High}. Thus, the effective energy cost per ray is:

$$E(p) = E_{Low} + error(p)E_{High} \tag{3}$$

Where p is the low precision, E_{Low} and E_{High} are the energy costs of the low and high precision intersector units respectively, and error(p) is the error rate of the low precision unit. One hand, the lower the precision p, the higher the error, and as a result more rays will need to be executed on the high precision unit to maintain image integrity, increasing energy costs. On the other hand, the higher the precision p, the lower will be the error rate; however, the baseline low precision unit will use more energy as it will be larger. This yields an optimization problem, where we search for p_{Min} the saves the total energy.

In our test cases, we evaluated our threshold choice through observation of the absolute number of rays still misclassified, as well as the SSIM (Structural Similarity Index) of the image compared to full precision. For our best design (32-16), a threshold of ~0.07 provided a good balance between striving for the greatest accuracy and hardware costs.

4.3 Hardware Design

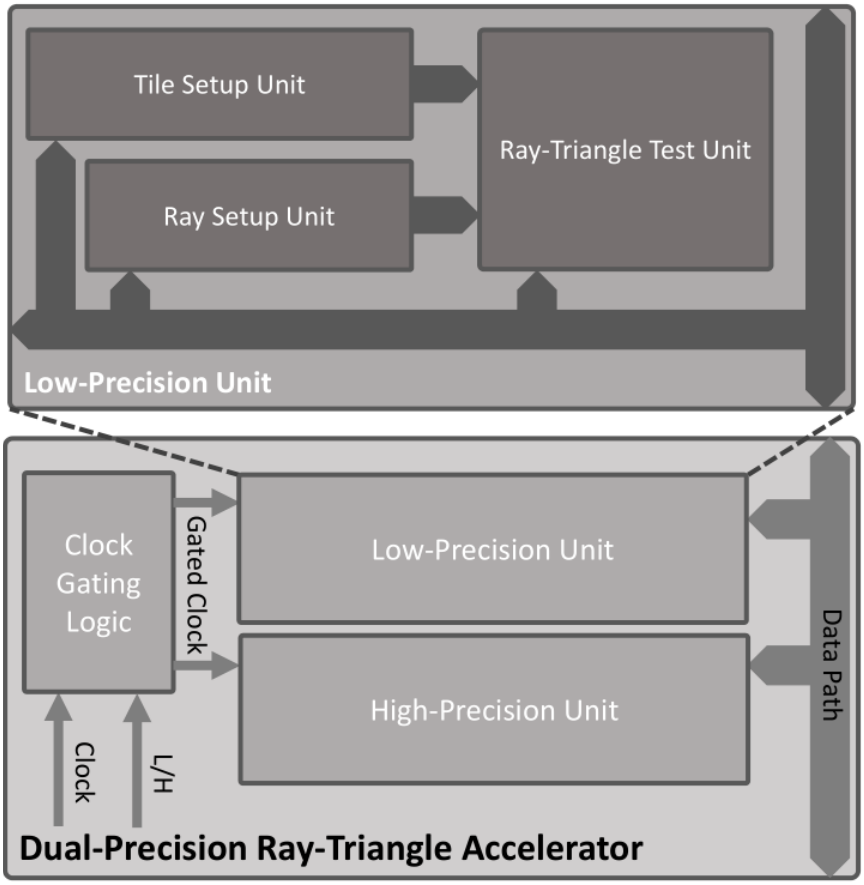

Figure 1: The architecture of the proposed hardware accelerator

We also fully design and implement our proposed methodology in hardware and evaluates the hardware metrics. Figure 1 illustrates the main components of the proposed hardware accelerator and their relationship to each other. As illustrated in the figure, we implement two data path pipelines, a low precision pipeline path and a high precision pipeline path, in conjugation with each other, where at any given time only one of the two precision paths is enabled, while the other is clock gated to reduce its power consumption. In this work, we implement a coarse-grain approach to clock gating, where the entire modules are clock gated, whereas alternatively, one can

implement a fine-grain clock gating scheme where the higher bits of a single computing unit are clock gated to achieve the same dual-precision effect.

Furthermore, each computing pipeline consists of three staged units, mimicking the architecture of the HVVR CUDA Implementation (Facebook, 2017). These three modules are: tile setup unit, where the triangles are first loaded, ray setup unit, where pre-computation is performed to determine the depth of the triangle compared to the ray origin, and ray-triangle test unit, where the actual intersection test in performed. Within our hardware implementation, it is assumed that for a given frame, the intersection accelerator will be computing sets of rays with a shared and known origin as is the case in augmented and virtual reality applications.

Given the triangle setup unit's computations all contribute directly towards the depth calculations, we perform all of its computations at high precision. Based on our profiling analysis of system calls (quantified in *Results* Section), we see that tile computations are between 5-10% as frequent as ray tests, so running these computations exclusively at high precision does not contribute significantly to the total power cost.

In our current implementation, the hardware computes the intersection using low-precision unit and returns the results back to software, where in accordance to the thresholding methodology previously described, the software activates the accelerator using the high-precision computing pipeline as required.

Finally, we note that we base our hardware design on the HVVR intersector, which inherently is not a watertight algorithm. We chose to avoid implementing a watertight algorithm as we are aiming for a design that operates within ultra-low-power budgets. We leave investigations into adding water-tightness to future work.

5. Results

5.1 Experimental Setup

We evaluate our methodology in terms of scene, accuracy, and hardware metrics (i.e. power, energy, design area). To measure accuracy we use hit accuracy, where we quantify the percentage of rays that have been misclassified. Furthermore, as a more perception-based metric, we also report the structural similarity index (SSIM) for all our experiments. We implement the proposed hardware designs in Verilog, and synthesize using Synopsys Design Compiler using the 7nm Arizona State Predictive PDK (ASAP7) in nominal processing corner.

For our tests, we use three well-recognized open-source scenes commonly used in the field, namely, Crytek Sponza, Conference Room, and Stanford Bunny (McGuire, 2017). Table 2 gives the number of objects and vertices, for each of our three test cases.

Scenes		
Crytek Sponza	Conference Room	Stanford Bunny
Triangles: 262267	Triangles: 331179	Triangles: 144046
Vertices: 184330	Vertices: 216862	Vertices: 72378

Table 2: The characteristics of the three evaluated scenes along with a representative image capture

5.2 Impact of Precision Scaling on Hardware Metrics

We implement and evaluate the hardware metrics for the precisions considered in the previous section. Since reducing the precision to 26-bits or below that significantly reduced the scene accuracy, moving forward, we only consider fixed-point precisions with more than 26-bits. Here we evaluate each of these hardware designs separately, and leave the evaluation as part of the dual-precision accelerator to a later section. Table 3 summarizes the hardware metrics obtained from our hardware accelerator.

As shown in the table, reducing the precision significantly reduces the hardware footprint and energy consumption. However, as discussed in the previous subsection, such reductions also result in depredations in quality of results. Therefore, the proposed dual-precision design can strike a balance between the two scenarios. In our work, as suggested by our experiments, we choose (40,24) for high precision unit (HPU) and (32,16) for the low precision unit (LPU). Next, we demonstrate, empirically, how the proposed thresholding scheme can be utilized to enable our methodology.

5.3 Evaluation of Duel-Precision Methodology

precision	Unit	Area (um^2)	Power (mW)	Energy (pJ)
32-bit Float Baseline-Precision	Tile Setup	74246.49	2.34	48.96
	Ray Setup	93447.53	2.84	61.07
	Ray Test	22681.81	1.01	18.13
(28,12)	Ray Setup	57180.66	1.23	6.16
	Ray Test	36872.70	0.52	2.62
(30,14)	Ray Setup	59868.28	1.36	6.82
	Ray Test	37112.75	0.53	2.65
(32,16)	Ray Setup	68434.79	1.49	7.46
	Ray Test	36955.75	0.53	2.65
(34,18)	Ray Setup	70844.34	1.75	8.73
	Ray Test	36818.12	0.53	2.64
(36,20)	Ray Setup	73846.88	1.97	9.83
	Ray Test	36724.10	0.53	2.63
(38,22)	Ray Setup	80182.53	2.13	10.66
	Ray Test	37211.43	0.53	2.65
(40,24) High-Precision Unit	Tile Setup	54470.18	1.58	7.92
	Ray Setup	86803.95	2.32	11.62
	Ray Test	39779.14	0.56	2.80

Table 3: The impact of precision scaling on the hardware cost

# of Bits	% Tests Flagged	% Base Error	% Final Error	Base SSIM	Final SSIM
(28,12)	44.04%	6.713%	0.865%	0.7403	0.9911
(30,14)	20.70%	2.134%	0.323%	0.8628	0.9914
(32,16)	6.62%	0.629%	0.100%	0.9336	0.9915
(34,18)	0.68%	0.188%	0.029%	0.9654	0.9916
(36,20)	0.04%	0.049%	0.010%	0.9857	0.9916
(38,22)	0%	0.012%	0.005%	0.9910	0.9915
(40,24)	0%	0.003%	0.003%	0.9916	0.9916

Table 4: The number of flagged tests, and misclassifications caught for different precision of LPU (HPU is fixed to (40,24)).

As described in *Methodology*, in order to find a balance between the number of flagged ray-triangle tests and the number of flagged misclassifications, we perform a study based on the distribution of the error for different precision. Table 4 shows the effect of changing threshold on the percentage of rays flagged for re-examination, as well as the percentage of misclassified tests corrected with the higher precision.

As indicated in the figure, more conservative thresholds result in a higher number of tests flagged for reevaluation, while also correcting a higher percentage of the low-precision errors. As evident from the table, the dual-precision methodology can effectively flag the significant portion of the tests misclassified in lower precision to be re-evaluated using the HPU. Next, we explore the hardware cost of such re-evaluations and justify the use of such methodology.

Figure 2 visualizes the restoration of the scene accuracy using the dual-precision methodology proposed in this work. Here, as demonstrated, a rendering significantly degraded by lower precision can be recovered to acceptable scene quality. We observe that most of the inaccuracies at that low precision are due to issues in depth computation rather than barycentric coordinate error. By running all depth computations at a high precision, we can correct the remaining errors through flagging rays closely outside of the edges of the triangles.

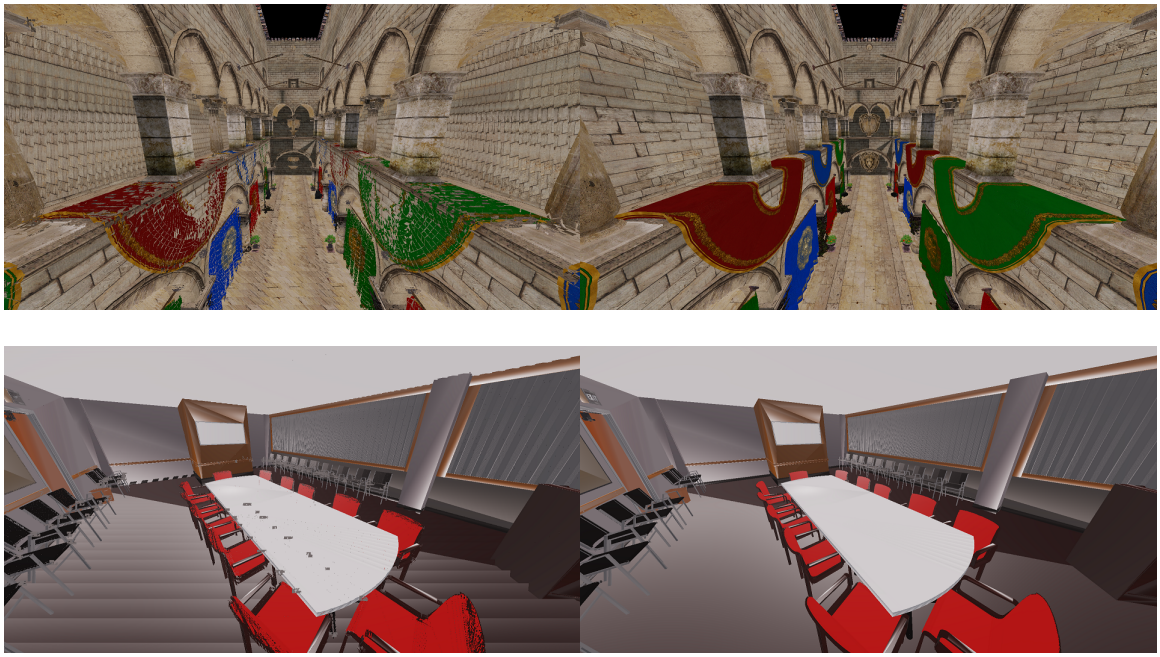

Figure 2: Low Precision (28 Bit) Figures from scenes,

without (left) and with (right) correction.

5.4 Hardware Cost Evaluation

Scenes	Tile Load	Ray Setup	Ray Test
Sponza	3,451,947	53,198,322	51,261,952
Conference Room	3,110,966	38,390,016	36,437,376
Stanford Bunny	554,268	10,281,976	9,971,200

Table 5: The profiled numbers of accelerator calls for different scenes.

In order to examine the number of computation required, we profile the HVVR CUDA implementation, therefore extracting the number of calls to each of hardware unit accelerators. Table 5 summarizes our results. As discussed in previous sections, millions of calls to each of the accelerators in required for the rendering of a single scene. Such significant numbers, results in consequential benefits when lower precision is utilized.

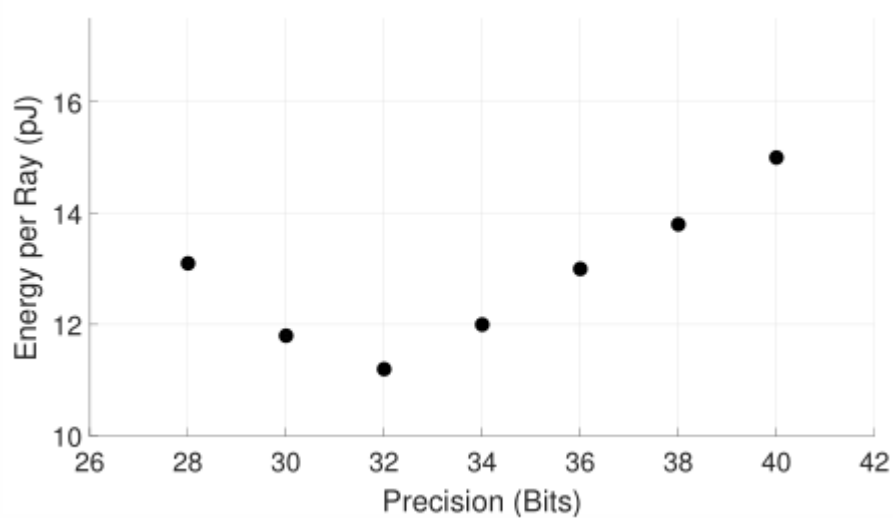

Figure 3: Energy per test for the proposed dual-precision design.

Floating Point Energy is 82.5pJ

Finally, Figure 3 plots the energy per ray as a function of the number of low precision bits as given by Equation 3. The Energy per Ray calculation assumes the percentages of rays flagged though thresholding determined in Table 4. The benefits demonstrated in this figure clearly highlight the significant benefits achieved using our dual precision methodology.

6. Conclusions

In this work, we proposed a dual-precision fixed-point ray-triangle intersection accelerator, where the majority of tests are performed on the lower precision, whereas only tests that are deemed *critical* utilize higher precision. To enable such a framework, we investigated a broad range of fixed-point arithmetic precision, and identify the optimal low precision that minimize total power consumption. We also developed a thresholding scheme where the criticality of the tests is determined. We fully implemented our proposed design in hardware using industrial tool flows and libraries and evaluated the accuracy and the hardware metrics offered by our methodology. As discussed, our methodology enables energy benefits of up to 86% with a minimal scene accuracy degradation of 0.1%, translating to an SSIM difference of 0.0001.

7. Appendix

7.1 Sponza Scene Images at Fixed Point: Bit Precisions 28-40

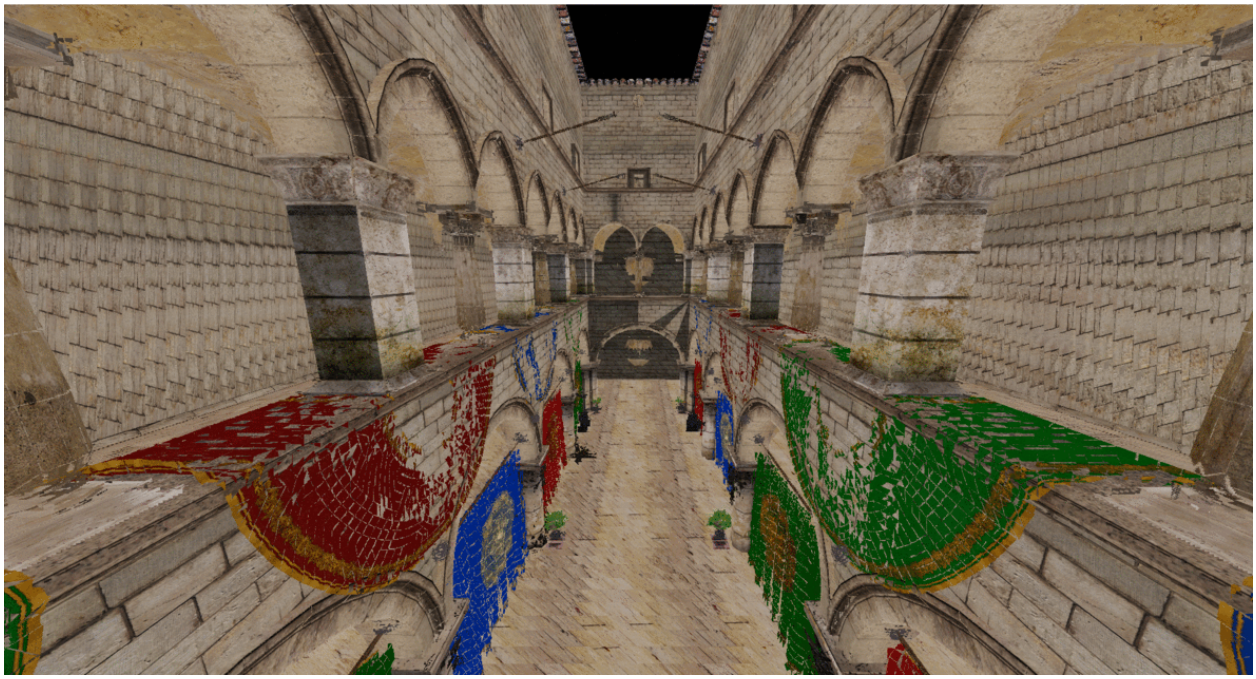

Figure 4: Sponza Scene: Fixed point (28,12) Bits

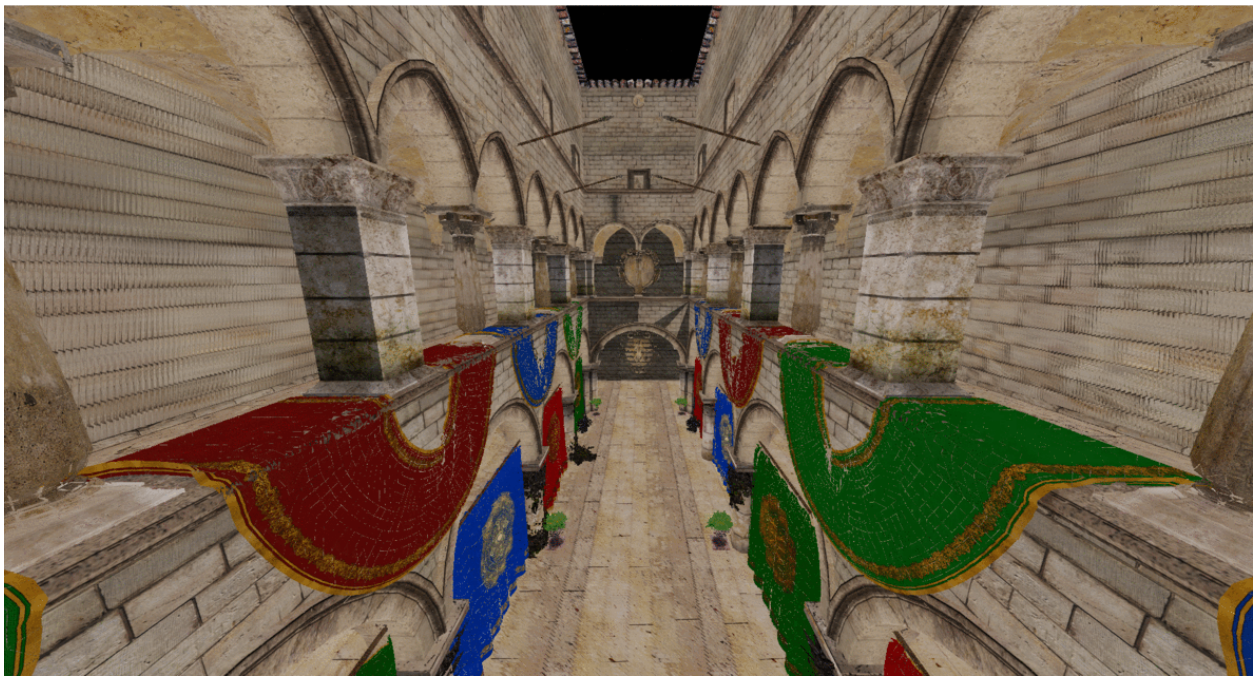

Figure 5: Sponza Scene: Fixed point (30,14) Bits

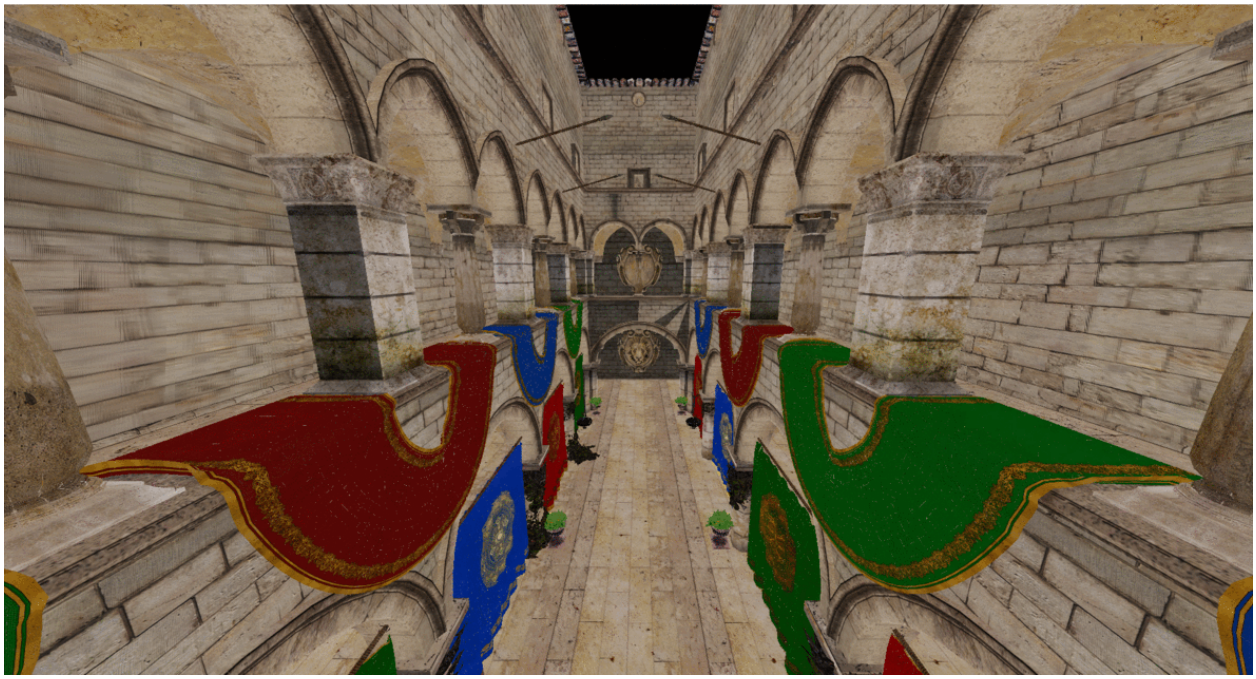

Figure 6: Sponza Scene: Fixed point (32,16) Bits

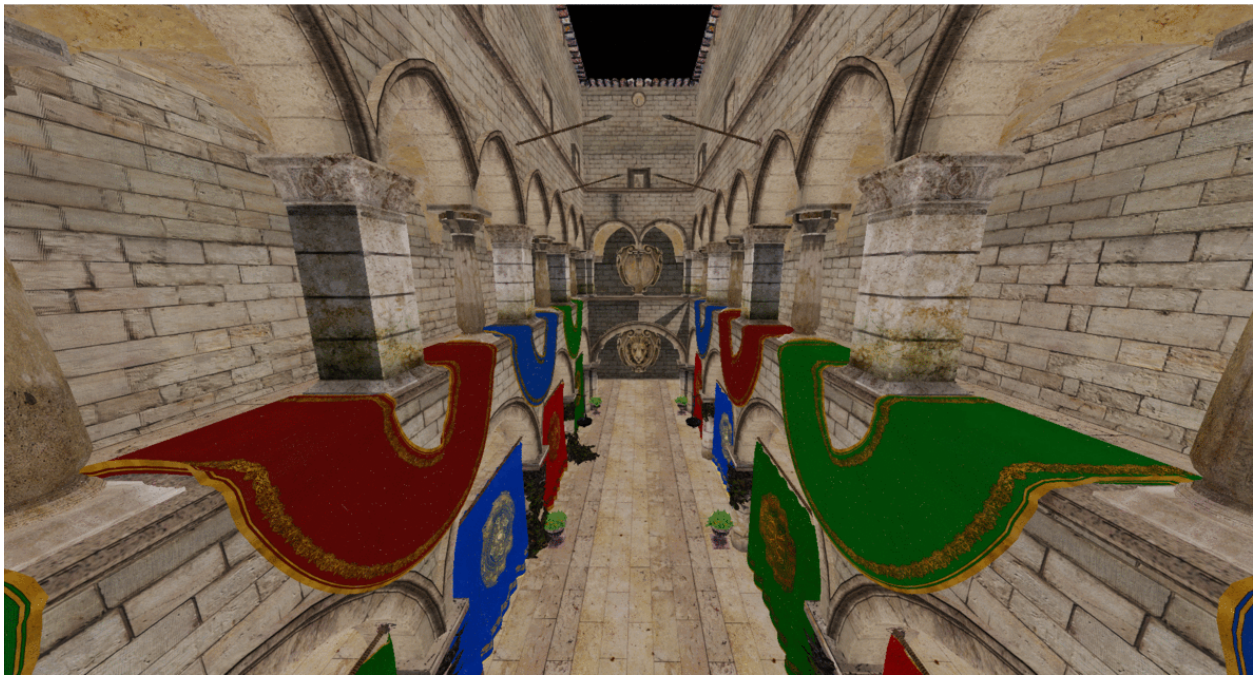

Figure 7: Sponza Scene: Fixed point (34,18) Bits

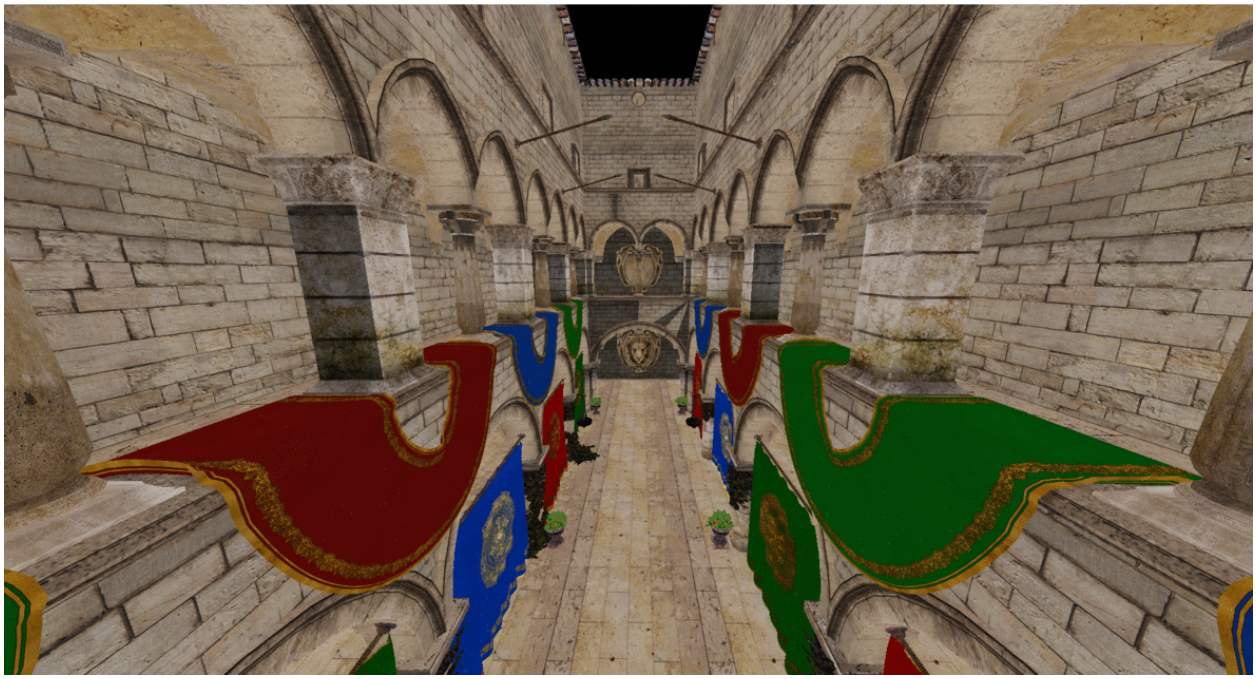

Figure 8: Sponza Scene: Fixed point (36,20) Bits

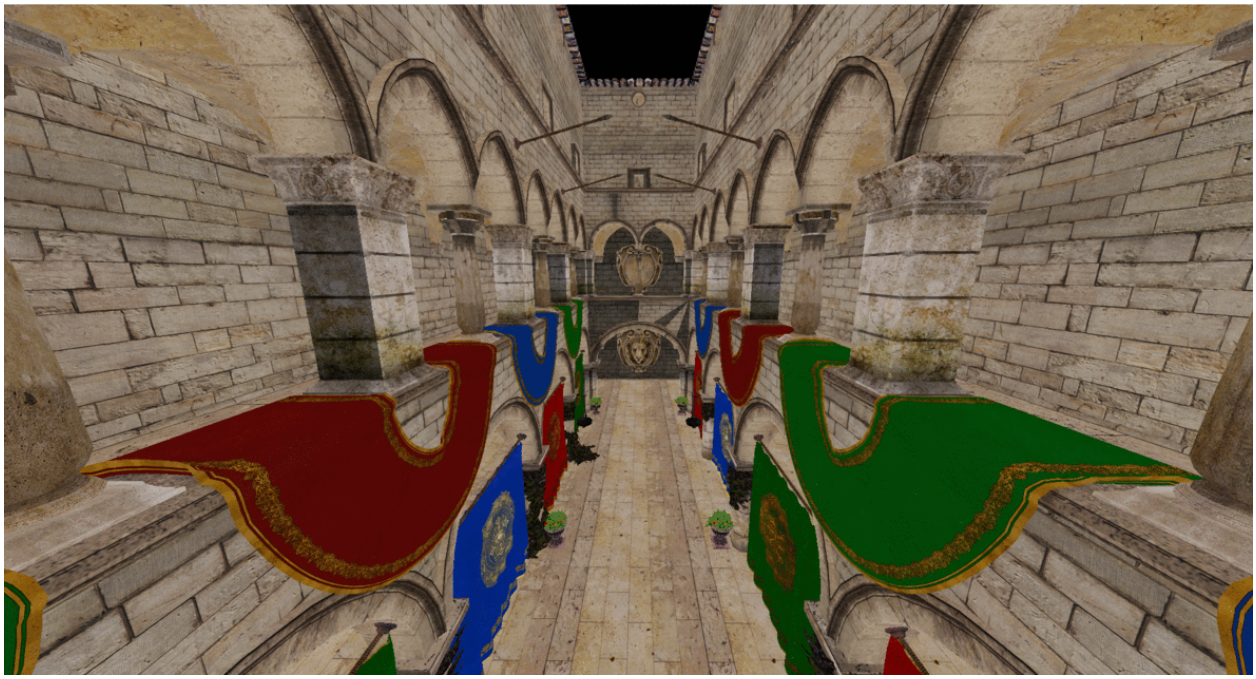

Figure 9: Sponza Scene: Fixed point (38,22) Bits

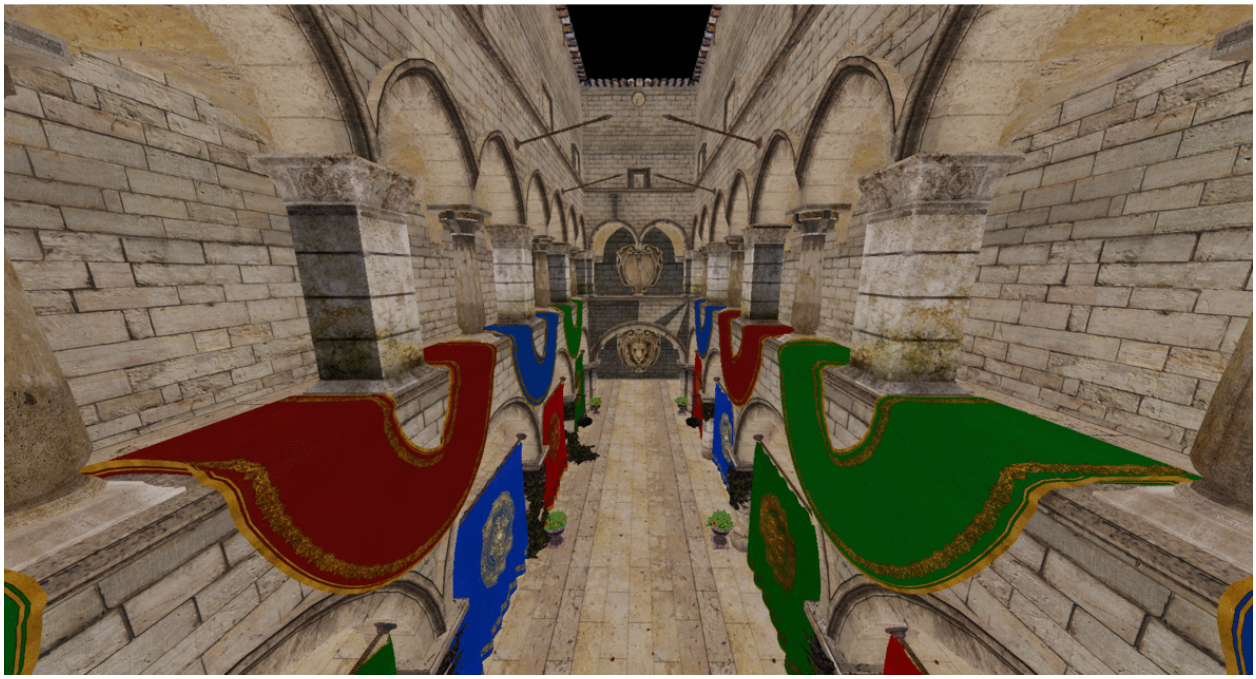

Figure 10: Sponza Scene: Fixed point (40,24) Bits

References

Bauszat, P. a. (2010). The Minimal Bounding Volume Hierarchy. *VMV*, (pp. 227--234).

Cline, D. a. (2006). Lightweight bounding volumes for ray tracing. *Journal of Graphics Tools*, 61--71.

Eisemann, M. a. (2008). Ray Tracing with the Single Slab Hierarchy. *VMV*, (pp. 373--381).

Fabianowski, B. a. (2009). Compact BVH storage for ray tracing and photon mapping. *Proc. of Eurographics Ireland Workshop*, (pp. 1--8).

Facebook. (2017). HVVR: Hierarchical Visibility for Virtual Reality. GitHub. Retrieved from https://github.com/facebookresearch/HVVR

Fascione, L. a. (2018). Path Tracing in Production. *ACM SIGGRAPH 2018 Courses* (pp. 15:1--15:79). Vancouver, British Columbia, Canada: SIGGRAPH.

Hanika, J., & Keller, A. (2007). Towards Hardware Ray Tracing using Fixed Point Arithmetic. *2007 IEEE Symposium on Interactive Ray Tracing*, (pp. 119-128).

Heinly, J. a. (2009). Integer Ray Tracing. *J. Graphics, GPU, and Game Tools*, (pp. 31-56).

Hunt, W. a. (July 2018). Hierarchical Visibility for Virtual Reality. *Proc. ACM Comput. Graph. Interact. Tech.*, 8:1--8:18.

Hwang, S. J.-J. (2015). A mobile ray tracing engine with hybrid number representations. (p. 3). ACM.

Keely, S. (2014). Reduced Precision for Hardware Ray Tracing in GPUs. *Proceedings of High Performance Graphics* (pp. 29--40). Goslar Germany, Germany: Eurographics Association.

Keely, S., & Fussell, D. (2018). Reduced Precision Ray-Triangle Intersection Filtering.

Keller, A. a.-J. (2013). Ray Tracing is the Future and Ever Will Be... *ACM SIGGRAPH 2013 Courses* (pp. 9:1--9:7). New York, NY, USA: ACM.

Kensler, A. a. (2006). Optimizing Ray-Triangle Intersection via Automated Search. *Symposium on Interactive Ray Tracing*, 33-38.

Kim, D., Nah, J.-H., & Park, W.-C. (2016). Geometry Transition Method to Improve Ray-tracing Precision},. *Multimedia Tools Appl.*, 5689--5700.

Kim, T.-J., Moon, B., Kim, D., & Yoon, S.-E. (2009). RACBVHs: Random-accessible Compressed Bounding Volume Hierarchies. *SIGGRAPH 2009: Talks* (pp. 46:1--46:1). New York, NY, USA: ACM.

Lee, W.-J. a.-W.-H.-S.-D. (2013). SGRT: A Mobile GPU Architecture for Real-time Ray Tracing. *Proceedings of the 5th High-Performance Graphics Conference* (pp. 109--119). New York, NY, USA: ACM.

Mahovsky, J. a. (2006). Memory-conserving bounding volume hierarchies with coherent raytracing. *Computer Graphics Forum* (pp. 173--182). Wiley Online Library.

McGuire, M. (2017, July). *Computer Graphics Archive*. Retrieved from https://casual-effects.com/data

NVIDIA. (2018). *NVIDIA TURING GPU ARCHITECTURE, Graphics Reinvented.* Retrieved from https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

Segovia, B. a. (2010). Memory Efficient Ray Tracing with Hierarchical Mesh Quantization. *Proceedings of Graphics Interface 2010* (pp. 153--160). Toronto, Ont., Canada, Canada: Canadian Information Processing Society.

Tomas Akenine-Möller, E. H. (2018). *Real-Time Rendering 4th Edition.* Florida: A K Peters/CRC Press.

Vaidyanathan, K. a.-M. (2016). Watertight Ray Traversal with Reduced Precision. *Proceedings of High Performance Graphics* (pp. 33--40). Goslar Germany, Germany: Eurographics Association.