Energy-Precision Tradeoffs in Mobile Graphics Processing Units

Jeff Pool, Anselmo Lastra, and Montek Singh
Dept. of Computer Science
University of North Carolina-Chapel Hill
{jpool, lastra, montek}@cs.unc.edu

***Abstract*—In mobile devices, limiting the Graphics Processing Unit's (GPU's) energy usage is of great importance to extending battery life. This paper focuses on the first stage of the graphics processor pipeline – the *vertex transformation stage* – and introduces an approach to lowering its switching activity by reducing the precision of arithmetic operations. As a result, the approach enables a tradeoff between energy efficiency and the quality of the rendered image.**

This paper makes the following specific contributions: 1) a transition-based energy model for quantifying energy consumed as a function of arithmetic precision, and 2) detailed simulation results on several real-world graphics applications to evaluate the tradeoff between energy and precision. In most examples, over 23% of the energy can be saved by lowering arithmetic precision while still maintaining a faithful reproduction of the full-precision image. Pushing the idea further, over 36% energy can be saved by further lowering the precision while preserving acceptable result accuracy. We assert that this represents a significant energy savings that warrants further investigation and extension of our approach to the remaining stages of the graphics processor pipeline.

I. Introduction

This work explores an approach for trading off computational precision in return for energy savings in mobile graphics architectures. We exploit the fact that, owing to the limited number of pixels in mobile graphics devices, computational precision can be significantly reduced without any perceptible loss of resulting image quality. Moreover, the precision can be further reduced for even greater energy savings with an acceptably low degradation in image quality. Fig. 1 illustrates the key idea by rendering a frame from a popular video game.

This work has significant implications for the graphics displays of both mobile devices, where battery life is determined by energy usage, and desktop computers, where graphics cards can require large amounts of power to be delivered and heat to be removed. By reducing the complexity of computation in the graphics pipeline, our approach seeks to lower energy usage of all devices.

This paper makes the following specific contributions: 1) an energy model for quantifying energy consumed as a function of arithmetic precision, and 2) detailed simulation results of real-world applications to evaluate the tradeoff between energy and precision.

While there has been some prior work on evaluating the minimum precision required for error-free rendering [1, 2], this work significantly extends and generalizes the concept in three ways. First, our approach allows precision reduction to be pushed into a regime where some graceful image quality reduction may be acceptable in energy-critical scenarios (e.g., perilously low battery in fieldwork). Second, our approach provides a unified framework for evaluating energy savings together with quality loss as a function of computational precision. Thus, unlike past work, our approach aims to allow the user to dynamically choose the operating precision. Third, our interval arithmetic approach is applied to any arbitrary vertex shader, which was not considered in [2].

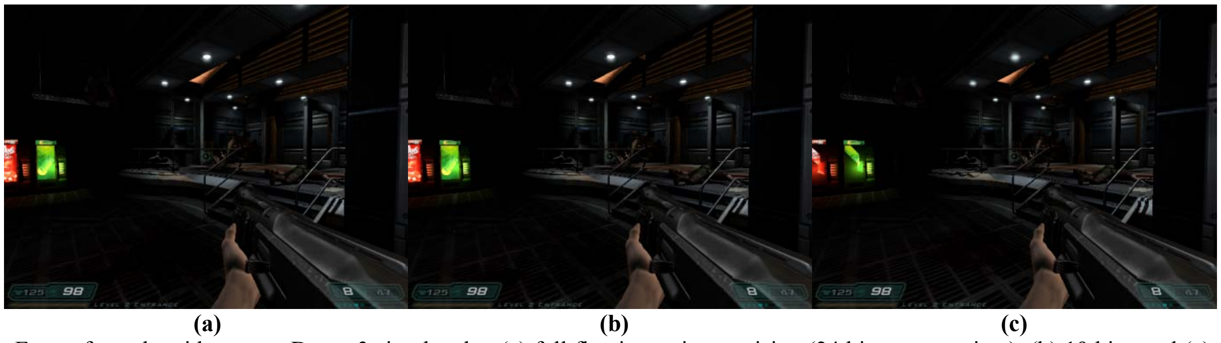

(a) **(b)** **(c)**

Fig. 1. Frame from the video game Doom 3 simulated at (a) full floating-point precision (24 bits per mantissa), (b) 19 bits, and (c) 16 bits of precision. The first two images are nearly visually identical, yet (b) saved 23% of the energy in the vertex transformation stage. While (c) exhibits a slight degradation in image quality, it saved 36% of the energy. Note that the slight loss of quality in 1(c) is due to errors in depth computation (known as "z-fighting errors"). These errors are sometimes present even in full-precision applications. When knowingly designing for a variable-precision video game, an artist could construct geometric models with care to avoid such errors.

In this paper, we have focused on the vertex transformation stage of the graphics pipeline to evaluate the feasibility of the concept of energy-precision tradeoff. The transformation stage converts vertex coordinates from user-defined floating-point representation to screen-space fixed point numbers. This transformation involves translation, rotation, scaling and perspective projection [3]. In addition, this stage computes the contributions of lights to the illumination at the vertices (again a floating-point computation). Ideally, our approach will be extended to other parts of the pipeline that also consume significant amounts of energy, such as memory, texture units, rasterization, and the fragment shader [3]. Moreover, our work so far has been at the behavioral level to focus on the viability of the idea; detailed architecture and circuit-level implementation is part of ongoing and future work.

Our approach was successfully validated using a hardware simulator. Several frames of real-world applications (e.g., Doom 3, Quake 4) were rendered at varying precisions, and the image quality and energy consumption quantified. In most cases, 23% energy savings was possible with little or no errors in resulting images. Further, with acceptable errors that would not affect application usage, even higher (36%) energy savings were obtained.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 details our analytical error model, and Section 4 presents the energy model used to determine savings with reduced precision. Our simulation approach and results are given in Section 5, followed by a discussion of these results in Section 6. Section 7 gives our conclusions and future work.

II. Related Research

Varying the precision of computations has been explored in prior work. Hao and Varshney [1] have given a thorough treatment of the sources of errors inherent in reduced precision rendering and the number of bits of precision required for a desired output. Akeley and Su [2] have used an interval arithmetic model to find the minimum triangle separation in eye space to ensure correct occlusion. Alalusi and Victor [4] have shown how a coarse-grained approach to reducing precision in hardware can be applied to integer applications. Tong et al. [5] and Jain [6] have reported similar findings for floating-point applications. Varying the quality of results in favor of faster or more reliable operation has also been examined, from Lafruit et al.'s [7] work on graceful degradation of 3D algorithms, to Yasuura et al.'s [8] treatment of energy-performance tradeoffs in computational systems, to Yeh et al.'s [9] exploration of error tolerance in physics-based animation. None of these approaches, however, has systematically explored energy-precision tradeoffs in a graphics pipeline.

There has also been research directed towards low power arithmetic. Liu and Furber [10] presented a low power multiplier, while Callaway and Schwartzlander detailed the relationship between power and operand size in CMOS multipliers. Phatak et al. [11] presented a low power adder and included a treatment of the adder's power usage dependent on the number of significant bits. Register files have also been the subject of energy efficiency research, Tseng [12], Zhao and Ye [13], Zyuban and Kogge [14], and Kim [15] have all examined limiting the power usage of register files. None of these designs was targeted for use in GPUs; however, we use them as a basis to develop a detailed energy model for a complete transformation stage of a GPU.

There are other low-power techniques as well, such as dynamic voltage scaling [16] and power gating [17], which could be used for energy-efficient graphics. These techniques, however, are lower-level circuit approaches, and are orthogonal to our work, which is an architecture-level approach.

III. Rendering Error Model

We introduce a model for predicting the error in vertex transformations and compare it to prior research conducted in the area. We show that our analytical model correlates with the results presented by Hao and Varshney [1], and will later compare these results to our experimental results. In all steps, "Error" is defined as the Euclidean distance (in pixels) between the screen position of a vertex transformed at full precision and that of the same vertex transformed at reduced precision. Precision, p, is 24 bits of computation per mantissa at full precision, and less than 24 bits for reduced precision.

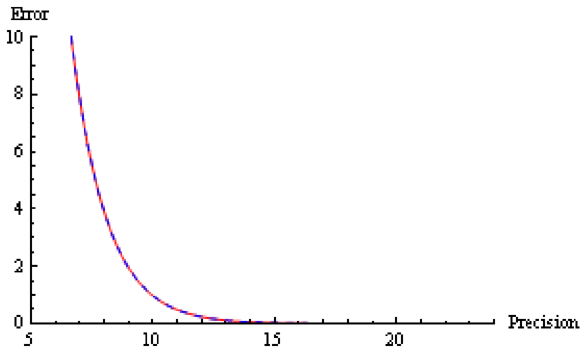


Fig. 2. Plot of our interval arithmetic model (red) and relationship presented in [1] (dashed blue) of error incurred by division as a function of increasing operand precision. The two curves are identical, reinforcing prior work [1] and validating our model.

We modeled the uncertainty imparted by reduced-precision computation using interval arithmetic, where each floating point number's true value is represented as a range with high and low bounds dependent on precision. The error in operations (addition, multiplication, division) was found by operating on interval representations of floating point numbers, and the entire transformation of projecting a vertex in world space (x, y, z) to screen space (x, y) was modeled with these operations.

For each operation involved in vertex transformation—addition, multiplication, and division—we developed a simulation using interval arithmetic, and applied it to a large set of random floating-point numbers. Each computation was performed at several precisions, from 7-bit to full 24-bit mantissas. The errors in the computed results were noted and compared against the errors predicted by the model of Hao and Varshney [1]. In every case, our results were identical to those of [1], which reinforces prior work and validates our model. As an example, Figure 2 shows the errors incurred during division, as a function of precision. Figure 3 shows the expected error incurred for the full transformation stage, which is composed of several instances of adders, multipliers and dividers.

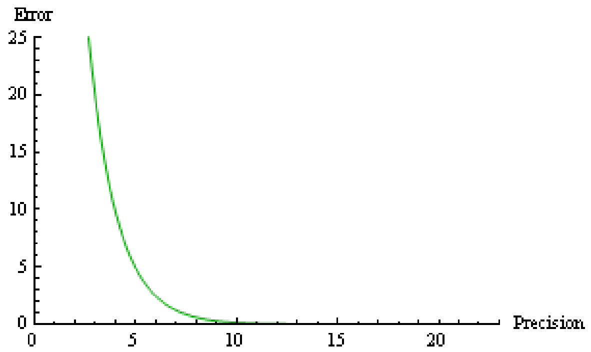


Fig. 3: Expected absolute screen space error (in pixels of a 640x480 pixel display) of a single transformed vertex as a function of precision.

IV. Energy Model

An energy model is presented for key arithmetic blocks in a graphics processing unit. In order to develop an energy model that is independent of process, clock speed, and other variations in implementations, we chose to approximate the energy usage as the number of total signal transitions per operation. Our architectural model uses floating point adders and multipliers as basic building blocks; all other operations are in turn implemented using these two basic operators. Some operations can be implemented in multiple ways; in these cases, we chose low-power, pipelined implementations.

For each operation, the energy usage was estimated by counting the number of transitions involved in that computation. This energy estimate was parameterized with the precision of the operands. The increased energy complexity of the new circuit-level implementations and the runtime energy overheads of dynamic precision switching are both ignored in this preliminary study in order to determine, at a high level, the effectiveness of our approach.

A. Addition

Floating-point additions are computationally expensive operations consisting of several steps. First, the operands' exponents must be made equal, which requires shifting a mantissa. Then, mantissas are added. Another normalization step is needed to align the sum's mantissa, and possibly another after an optional rounding step.

To model the energy spent in an addition, we focus on the mantissa addition and shifting stages. Jain [6] found that the energy consumption of a floating-point adder is highly dependent on the width of the significand, and that limiting the rounding mode to truncation will significantly reduce energy consumption. In practice, most FPUs on graphics hardware are not fully IEEE compliant, especially in rounding modes supported, since in graphical applications, as Tong et al. [5] noted, "delicate rounding modes are not required." We assume here that simple Round toward Zero (truncation) is used.

Mantissa addition was performed using the low-power design for an integer adder reported by Phatak et al. [11]. We generalized their results for energy consumption in adders, from a few distinct sizes to arbitrary sizes, and saw the expected linear relationship. Accounting for the linear energy consumption in a barrel shifter [3], we arrived at the following equation for the expected number of signal transitions as a function of precision *p:*

$$T_{Add}(p) = 13.5 * p \tag{1}$$

B. Multiplication

Multiplication is modeled as integer multiplication at a given precision. Tong et al. [5] found that 81.2% of the energy consumed in floating point multiplication is spent in the mantissa multiplication or over 98% when the rounding unit is disregarded, which is the case for simple truncation. Therefore, we focus on the mantissa multiplication, and chose to base our energy model on a standard array multiplier introduced by Liu and Furber [10], as it is energy-efficient and easily pipelined, which can aid in dynamically varying the precision. Modeling their results as a second-order polynomial to handle both the linear and quadratic operations in a floating-point multiplication operation, we arrived at the following relation:

$$T_{MUL}(p) = 1.74 * p + 0.38 * p^2 \tag{2}$$

C. Reciprocal/Reciprocal Square Root

Historically, several types of iterative reciprocal and reciprocal square root calculations have been used in hardware. SRT division converges upon the correct result linearly, while Newton-Raphson (and others based on Newton's method) [18] and Goldschmidt (and other power series expansions [19]) converge quadratically to the result.

In order to make more use of low-power units, we chose to model reciprocal and reciprocal square root computations with narrow bit-width multiplications introduced by Ercegovac et al. [20], based on Taylor series expansion. This method consists of a reduction step, evaluation step, and post-processing. Several iterations of the evaluation step are needed, for which some operations require only *p*/4 bits of precision. When the energies for all stages are summed, the total transition counts are as follows for a reciprocal (3) and a reciprocal square root (4) operation:

$$T_{RCP}(p) = \log_2 p * \; 5 * T_{MUL} \; \frac{p}{4} \; + T_{ADD} \; \frac{p}{4} \; + T_{MUL}(p) \tag{3}$$

$$T_{RSQ}(p) = \log_2 p * \; 4 * T_{MUL} \; \frac{p}{4} \; + T_{ADD} \; \frac{p}{4} \; + T_{MUL}(p) \; + T_{MUL}(p) \tag{4}$$

D. Dot Product

We modeled the energy consumed in 3- and 4-component dot products as the sum of transitions in the constituent additions and multiplications:

$$T_{DP3}(p) = 3 * T_{MUL}(p) + 2 * T_{ADD}(p) \tag{5}$$

$$T_{DP4}(p) = 4 * T_{MUL}(p) + 3 * T_{ADD}(p) \tag{6}$$

E. Multiply-Add

Similar to dot products, a multiply-add operation can be modeled as a combination of a multiplication and an addition:

$$T_{MAD}(p) = T_{MUL}(p) + T_{ADD}(p) \tag{7}$$

F. MIN/MAX

Comparisons are typically implemented as a subtract operation followed by checking the sign bit of the result. Therefore, the energy for a min-max operation is simply modeled as an addition:

$$T_{MINMAX}(p) = T_{ADD}(p) \tag{8}$$

G. Register File

The energy consumed in reading and writing a register operand is linearly related to the operand's width, as Alalusi and Victor point out, since control overhead is negligible compared to the energy spent in the actual data path [3]. The analytical models used for register file reads and writes are given below, in Equations 9 and 10. The constant term comes from the selection of the correct register within the file and will vary with size of the register file, but not the precision. The addition of 8 to the precision is to take into account the sign bit and exponent in addition to the mantissa precision, *p*.

$$T_{READ}(p) = 2 * (p + 8) + 16 \tag{9}$$

$$T_{WRITE}(p) = 3 * (p + 8) + 16 \tag{10}$$

H. Summary

Most arithmetic operations are built upon the addition and multiplication units. Transitions incurred in addition are purely linear with respect to precision, while multiplication's equation has a quadratic term. Thus, we expect our experimental results to have a second-degree polynomial curve, somewhere between purely linear and purely quadratic, depending on the relative frequencies of use of these operations.

V. Experimental Results

In order to utilize our energy model and visualize the actual errors in rendering, we incorporate the model into ATTILA, a cycle-accurate GPU simulator [21]. We added energy logging functionality: when an operation is executed, it calculates its own energy usage based on the current precision (kept constant during a given simulation) and logs this information for further analysis. In addition, the simulator was modified to provide full support for variable-precision rendering. The GPU's arithmetic functions were modified to operate on a custom data type that performs truncation of floating-point results to any specified precision. For each simulated frame, the transformed vertices and resulting color buffer are also saved, which allows for examination of error caused by reducing transformation precision. Thus, the data gathered from the simulator is the energy usage of each operation per cycle, the final frame buffer for the simulated frames, and the positions of transformed vertices.

We quantified the errors in the *x* and *y* dimensions in individual transformed vertices, since such errors are readily visible on screen as horizontal and/or vertical displacement of the rendered vertex.

However, errors along the *z* dimension (i.e., depth) manifest quite differently. Since the resulting rendered image is in 2D, errors in depth for an individual vertex do not translate to any *x* or *y* errors. However, depth errors can cause a triangle to become incorrectly occluded by another triangle whose depth happens to be incorrectly computed to be nearer to the eye than the former. This phenomenon is called "z-fighting," and it can sometimes cause errors that are more apparent than *xy* errors. A detailed discussion of this type of error is presented in Section VI-A.

The applications that we simulated and analyzed were Doom 3, Prey, Quake 4, and a simple torus viewer, all traces released by the ATTILA group specifically for use with the simulator. Several hundred frames (to create useful videos) of the first two applications were simulated, and several sample frames, used for energy and error analysis, were logged for all four applications.

A. *Rendering Error*

As a frame of an application was simulated with ATTILA, information regarding its transformed vertices was logged, and only those vertices that were within the view frustum were analyzed. For those vertices that were on screen at full precision, the error was found by taking the root of the squares of the *x* and *y* distances to the full precision position. As in [2], clipped vertices were not taken into account. The errors seen at precisions less than 8 bits were, for the most part, far too high to make the applications useable, so only precisions greater than or equal to 8 bits were simulated and analyzed for these examples.

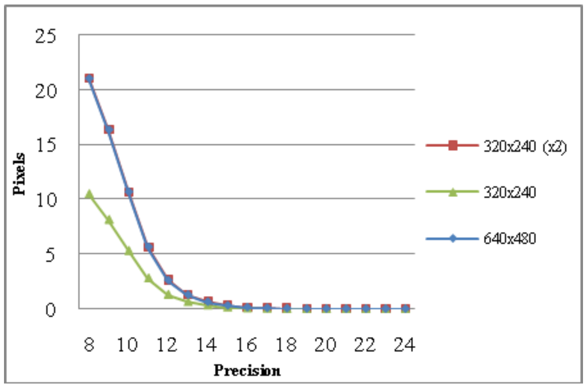


Fig. 4: Average screen space error in the frame of Doom 3 (1). The *xy* error is negligible until the precision drops below 15 bits. A lower-resolution version of the same frame shows a similar pattern, and nearly identical error values when the lesser precision image is scaled by a factor of two.

Comparing error versus precision across multiple applications reveals that the pattern is consistent and is not limited to just one application. Table 1 lists data gathered from each of the applications. Figure 4 shows the results from the frame of Doom 3 seen in Figure 1, with little to no error at high precisions, and increasing error as precision drops. Figure 5 shows that the other applications follow a similar trend, which is consistent with our analytical model. (The frames used for error and energy analysis in each of the remaining applications are shown in Figure 6, for reference.) Doom 3 has the highest overall error, while the torus shows very little error, even at quite low precisions.

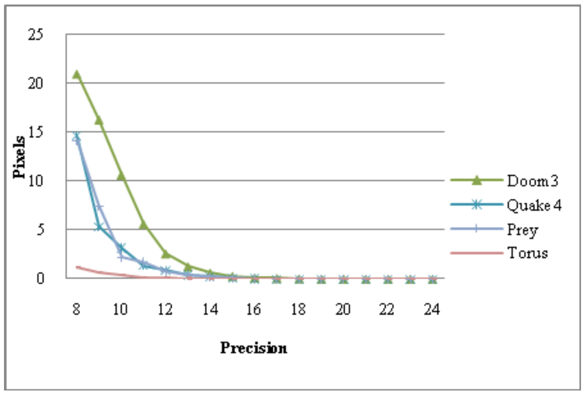


Fig. 5: Average screen space error for a single frame of several applications rendered at 640x480 pixels. The trend seen in each application is similar to the pattern predicted by our analytical model (3) on a different scale due to the various positions of untransformed vertices.

The low error of the torus model may be due to its relative simplicity. Its geometry is compact and regular, similar to that of other mobile 3D applications, such as a GPS display or graphical user interface (GUI), while the other applications have vertices both very close as well as very far away in all regions of the screen that are subject to disparate transform matrices.

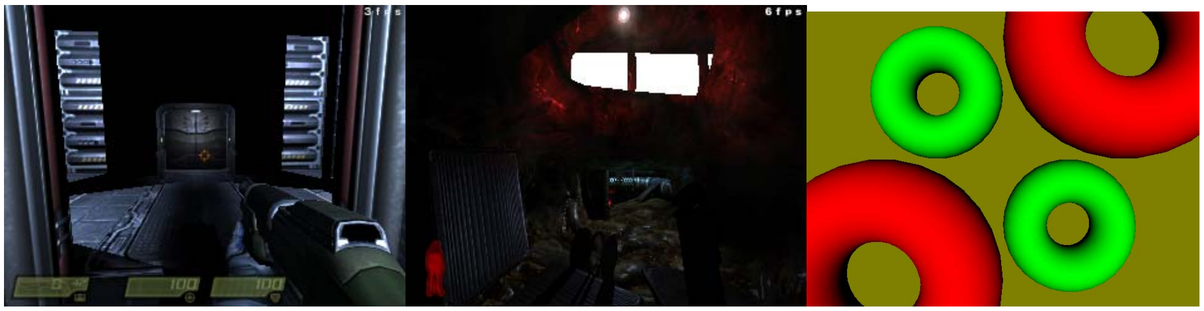
The *xy* errors are quite small and even less of a factor in application usability than when first considered. A user may not even see a screen-space error of several pixels as an artifact, since this error would be shared by all triangles that share that vertex. So, all related geometry would be moved, not just a vertex or triangle here and there, making the error less visible. Partly for this reason, the *xy* error is not the limiting factor when choosing a precision. See Section VI for further discussion of this phenomenon.

B. *Energy Savings*

The energy characteristics of the applications were as generally expected, given our energy model: the energy usage was higher at higher precisions, and decayed nearly linearly towards lower precisions. There is a slight upward curve in the plot, but it is not very pronounced since the multiplication unit's quadratic coefficient is relatively small. Work involved in transforming vertices is not dependent on screen size, so the results were identical for the same frame of a given application at different sizes. Fig. 7 shows the graph of simulated energy vs. precision, and Table 2 gives the data for alternate precisions for reference, along with the total number of vertices transformed for the sample frames.

C. *Energy-Precision Tradeoff*

We present a method of characterizing the tradeoffs in energy and image quality in reduced precision rendering. We allow the designer and user of an application to choose a



(a) **(b)** **(c)**

Fig. 6: Single frames simulated for error/energy purposes of (a) Quake 4, (b) Prey, and (c) a torus viewer, which has much simpler and more compact geometry.

Application	Doom 3		Quake 4		Prey		Torus
Resolution	320x240	640x480	320x240	640x480	320x240	640x480	640x480
Precision							
8	10.54	21.02	7.33	14.62	7.72	14.17	1.169
10	5.34	10.68	1.62	3.20	1.26	2.25	0.355
12	1.32	2.60	0.49	0.93	0.46	0.80	0.089
14	0.33	0.64	0.11	0.21	0.14	0.23	0.020
16	0.10	0.14	0.02	0.04	0.03	0.05	0.006
18	0.02	0.04	0.01	0.01	0.01	0.01	0.002
20	0.01	0.01	~0.00	0.01	~0.00	~0.00	0.001
22	~0.00	~0.00	~0.00	~0.00	~0.00	~0.00	0
24	0	0	0	0	0	0	0

Table 1: Summary of average error per vertex (in pixels) in a single frame of applications simulated at various precisions. It is seen that each application's average error increases with a decrease in precision, as expected. Also, resolution plays only a minimal role on relative screen space error. That is, doubling the resolution effectively doubles the error of a transformed vertex. In Prey's case, though, the relative error actually lessens to a minor degree with an increase in resolution. Thus, increases in display resolutions will not pose any problem to the efficacy of reduced precision transformations.

balance between energy savings and image quality appropriate to their needs. Figure 8 shows the energy-precision tradeoff curves for all simulated applications. Energy usage is normalized for each application so savings are readily apparent as a percentage of the total energy consumed. We found that xy errors did not cause any perceptible errors when these errors were less than a tenth of a pixel on average. Furthermore, applications did not become unusable until the errors in *x* and *y* exceeded, on average, a pixel. At these errors, energy saved was roughly 40% and 50%, respectively. However, actual savings were not quite this pronounced, since *z*-fighting limited the utility of the applications before *xy* errors grew to an unacceptable level.

VI. Discussion

We briefly discuss two important findings of our work. Errors caused by *z*-fighting, not errors in *xy* screen position, dictated lower bounds for precisions, and the applications tested did not share a common lowest acceptable precision.

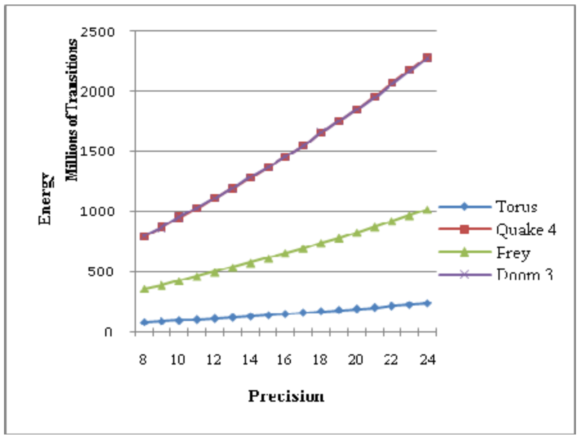


Fig. 7: Energy usage as a function of precision. Quake 4 and Doom 3 had nearly the same energy usage, which was a little more than twice that of Prey. The torus viewing application, though it used relatively little energy, still showed the same relative savings at reduced precisions.

A. *Z Errors*

Significant *xy* screen space error did manifest itself at very low precisions, but the limiting factor in playability was due to *z* errors. Videos made from frames rendered at different precisions reveal that *z* errors are visible far before *xy* errors in the transformed geometry.

As mentioned above, *xy* errors will be shared among triangles sharing a particular vertex, so they will not be seen as actual errors until progressive viewpoints cause the error to be manifested differently. Much more apparent are errors in the depth of a vertex. As precision drops, the depths of transformed vertices begin to converge to a more and more limited set of values. This results in *z*-fighting, seen in the vending machines in Fig. 1(c), which, since it flickers from frame to frame, is much more immediately apparent to a user than slight vertex displacement.

	Doom 3	Quake 4	Prey	Torus
Vertices	193080	163875	126297	17154
Precision				
8	805	798	364	82
10	962	954	432	98
12	1117	1111	503	115
14	1292	1287	580	132
16	1464	1461	660	151
18	1668	1664	745	171
20	1858	1856	832	192
22	2076	2075	927	215
24	2283	2284	1022	238

Table 2: Transition count for a single frame of several applications in millions of transitions. The top line of data in the table gives the total number of vertices transformed in rendering the frame.

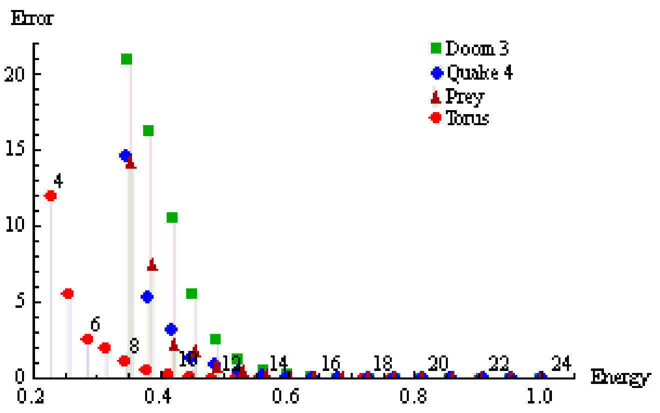


Fig. 8: Energy-Precision tradeoff curves for all simulated applications. The clustering of application energy usage at a given precision (shown by vertical lines) at specific normalized energy values indicates that energy savings at a certain precision (inset numbers) is not heavily dependent on the content. It is also seen that each application's curve is similar in shape; the difference is in the magnitude of errors. In general, the games tested needed higher precisions to remain usable than did the torus viewer.

This trend has an important implication: the precision at which *xy* errors become unreasonable is much lower than the precision at which depth errors become unreasonable. There are several ways that developers can address this issue. These methods are well known to developers and artists, as *z*-fighting is a problem even at full precisions. Reduced precisions require more aggressive use of these techniques. If hardware depth testing is disabled in areas prone to *z*-fighting and the correct draw order is observed, there will be no ambiguity as to which of two coplanar polygons should be drawn. This will eliminate the *z*-fighting artifact, but could add an extra step to the graphics programmer's pipeline. Also, when an artist designs a model for an application, such as the vending machines, designing them so that there are not two near-coplanar faces can greatly delay the onset of *z*-fighting artifacts as precision is reduced.

B. *Content-Dependence of Errors*

There was no specific error or precision at which all the simulated applications became unusable. Discrepancies between acceptable precisions in applications are to be expected. Deering [22] noticed that in at least one case, a 3D model actually looked *better* when it had fewer bits dedicated to its representation, yet other models became unrecognizable after only a few bits were lost. Chow [23] suggests that multiple precisions may be necessary to represent an object well when compressed, which can have much the same effect as reduced precision.

VII. Conclusions and Future Work

We have shown that there is a tradeoff inherent in variable-precision vertex transformations between the error and energy savings for an application. After developing an energy model for the transformation stage of a GPU, we simulated several applications at different precisions. Our experimental results for vertex position error matched those of past research. Furthermore, we have explored the relationship between rendering error and energy savings, both analytically and experimentally, in order to help developers and users of mobile applications choose an operating range to find an acceptable mix of error and energy savings to prolong battery life. Our error model can be applied to any arbitrary vertex shader, allowing a developer to easily see the errors incurred by a shader in use in their applications. They can then take appropriate action to ensure these errors do not cause undesired effects for the end user.

Our results show that significant energy savings can be achieved by using reduced-precision vertex computation, thereby indicating that this idea holds promise. We plan to extend our work to save energy in all parts of the mobile graphics pipeline. The fragment shader uses similar operations, and texture units, rasterization, and memories may all gain energy savings by reducing precision. We will also design circuits to implement variable-precision operations, an important part of proving viability and understanding the static and dynamic overheads involved in varying precisions.

ACKNOWLEDGEMENT

Financial support was provided by the National Science Foundation under grant CCF-0702712. Equipment was provided by NSF Research Infrastructure grant number 0303590.

REFERENCES

[1] X. Hao and A. Varshney, “Variable precision rendering,” *Interactive 3D Graphics*, 2001, pp 149-158. [Online] http:// doi.acm.org/ 10.1145/364338.364384

[2] K. Akeley and J. Su, “Minimum triangle separation for correct Z-buffer occlusion,” *Graphics Hardware*2006, pp.27-30.

[3] J. D. Foley, A. vanDam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice in C.* Addison-Wesley, 2nd edition, 1995.

[4] S. Alalusi and B. Victor, “Variable word width computation for low power,” CS 252 Computer Architecture (unpublished). Berkeley (2000).

[5] J. Tong, D. Nagle, R. Rutenbar, “Reducing power by optimizing the necessary precision/range of floating-point arithmetic.” *IEEE Trans. on VLSI Systems*, June 2000, pp. 273-286.

[6] S. Jain, “Low-power single-precision IEEE floating-point unit,” MIT (Master’s Thesis), May 2003.

[7] G. Lafruit, L. Nuchtegaele, K Denolj, J. Bormuns, “3D computational graceful degredation.” *IEEE International Symposium on Circuits and Systems*, May 2000.

[8] H. Yasuura, T. Ishihara, M. Muroyama, “Energy management techniques for SOC design.” *Essential Issues in SOC Design*, 2006. Ed. Youn-Long Steve Lin.

[9] T. Yeh, G. Reinman, S. Patel, P. Faloutsos. “Fool Me Twice: Exploring and Exploiting Error Tolerance in Physics-Based Animation,” *ACM Transactions on Graphics*, 2007.

[10] Y. Liu, S. Furber, “The design of a low power asynchronous multiplier,” *International Symposium on Low Power Electronics and Design*, August 2004, pp. 365-371.

[11] D. Phatak, S. Kahle, H. Kim, J. Lue, “Hybrid signed-digit representation for low power arithmetic circuits,” *Proceedings of the Lower Power Workshop in Conjunction with ISCA*, June 1998. [Online] http://www.cs.umbc.edu/~phatak/publications/hsd-lowpower.pdf

[12] J. Tseng, “Energy-efficient register file design,” MIT (Mater’s Thesis), Dec. 1999. [Online] http://www.cag.lcs.mit.edu/scale/papers/jhtsengsm.pdf

[13] X. Zhao and Y. Ye, “Structure configuration of low power register file using energy model,” *IEEE Asia-Pacific Conference on ASIC*, 2002, pp.41-44. [Online] http://ieeexplore.ieee.org/iel5/8021/22152/01031527.pdf

[14] V. Zyuban and P. Kogge, “The energy complexity of register files,” Intl. *Symposium on Low Power Electronics and Design*, Aug. 1998, pp.305-310. [Online] http:// doi.acm.org/10.1145/280756.280943

[15] S. Kim, “Reducing ALU and register file energy by dynamic zero detection,” *Performance, Computing, and Communication Conference*, April 2007, pp.365-371. [Online] http://ieeexplore.ieee.org/iel5/4197898/4197899/04197951.pdf

[16] J. Mao; Q. Zhao; C., C.G., "Optimal dynamic voltage scaling in power-limited systems with real-time constraints," *43rd IEEE Conference on Decision and Control*, Dec. 2004, pp. 1472-1477.

[17] M. H. Chowdhury, J. Gjanci, P. Khaled, “Innovative Power Gating for Leakage Reduction,” *IEEE International Symposium on Circuits and Systems 2008*, May 2008, pp. 1568-1571.

[18] D. Chen, B. Zhou, Z. Guo, P. Nilsson, “Design and implementation of reciprocal unit,” *48th Midwest Symposium on Circuits and Systems*, Aug. 2005, pp.1318-1321, Vol. 2. [Online] http://ieeexplore.ieee.org/iel5/10622/33557/01594352.pdf

[19] N. Foskett, R. Prevett, S. Treichler, “Method and system for performing pipelined reciprocal and reciprocal square root operations,” *U.S. Patent 7117238, NVIDIA Corporation*, Oct. 2006.

[20] M. Ercegovac, T. Lang, J-M. Muller, and A. Tisserand, “Reciprocation, square root, inverse square root, and some elementary functions using small multipliers,” *Research Report #97-47, Laboratoire de l’Informatique due Parallelisme*, Nov. 1997.

[21] V. Moya, C. Gonzalez, J. Roca, A. Fernandez, R. Espasa, “ATTILA: A cycle-level execution-driven simulator for modern GPU architectures,” *International Symposium on Performance Analysis of Systems and Software,* March 2006. [Online] http://personals.ac.upc.edu/vmoya/docs/ISPASS%20-%20ATTILASim.pdf

[22] M. Deering, “Geometry compression,” *Proceedings of the 22nd Annual Conference on Computer graphics and Interactive Techniques*, Sept.1995, pp.13-20. [Online] http://doi.acm.org/10.1145/336154.336227

[23] M.M. Chow, “Optimized geometry compression for real-time rendering,” *Visualization ’97, Proceedings*, Oct.1997, pp.347-354, 559. [Online] http://ieeexplore.ieee.org/iel4/5302/14360/00663902.pdf