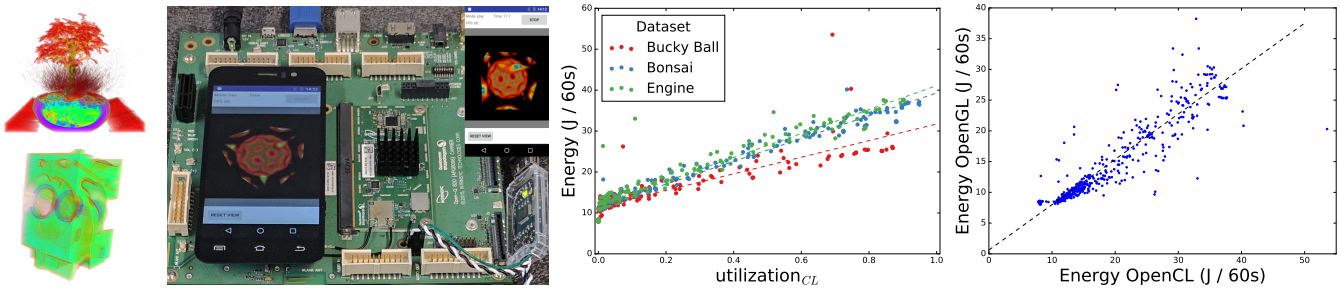
Eurographics Conference on Visualization (EuroVis), Posters Track (2017)
A. Puig Puig and T. Isenberg (Editors)

Power Efficiency of Volume Raycasting on Mobile Devices

M. Heinemann, V. Bruder, S. Frey, and T. Ertl

University of Stuttgart, Germany



(a) *Datasets* **(b)** *mobile hardware test setup and app* **(c)** *energy consumption w.r.t. dataset* **(d)** *OpenCL vs. OpenGL*

Figure 1: *Raycasting volume datasets (a) on a mobile system (b) shows a direct correlation between the GPU utilization and the energy consumption for different parameter configurations (c). However, depending on the dataset, there is a different trend noticable (c). Furthermore, an increased energy consumption can be measured when using the OpenCL API compared to OpenGL (d).*

Abstract

Power efficiency is one of the most important factors for the development of compute-intensive applications in the mobile domain. In this work, we evaluate and discuss the power consumption of a direct volume rendering app based on raycasting on a mobile system. For this, we investigate the influence of a broad set of algorithmic parameters, which are relevant for performance and rendering quality, on the energy usage of the system. Additionally, we compare an OpenCL implementation to a variant using OpenGL. By means of a variety of examples, we demonstrate that numerous factors can have a significant impact on power consumption. In particular, we also discuss the underlying reasons for the respective effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms

1. Introduction and related work

Mobile devices such as smartphones have gained much popularity over the last decade, contributing to a significant advancement of their hardware over the years. Latest systems-on-a-chip (SoC) enable complex rendering applications on such mobile devices, by providing decent computing power through integrated GPUs that support modern APIs such as OpenCL or OpenGL. However, running compute intensive tasks, such as rendering 3D objects over a longer period of time, is often limited by battery capacity. Therefore, the understanding of the factors that influence energy consumption plays an important role for these applications, because optimizations in this regard can reduce power consumption and ultimately lead to a longer runtime of the respective mobile device. In this work, we investigate the power consumption of a mobile direct volume rendering application. In doing this, we focus on analyzing the influence of different rendering parameters, such as frames per second (FPS), resolution, or dataset, on the energy consumption of the whole system. Furthermore, we examine the difference in using the OpenCL computing API compared to the OpenGL rendering pipeline with respect to power efficiency. In previous work, Johnsson et al. [JGDAM12] analyze the energy consumption of multiple rendering algorithms on different GPUs, including desktop hardware and mobile platforms. They found out that power consumption varies significantly over different architectures and is not directly proportional to computation speed. While we focus on volume racyasting in our work, Johnsson et al. analyze algorithms for the classic rendering pipeline, such as shadow algorithms. Similarly, Mochocki et al. [MLC06] discuss and optimize the power consumption of the mobile rendering pipeline by looking at the influence of various parameters such as frame rate, resolution, level-of-detail, lightning, and texture model. For non-rendering applications, Aragon et al. [AJM*14] compare the power consumption of algorithms running on CPU against GPU implementations. Collange et al. [CDT09] and Ma et al. [MDZD09] discuss the GPU power consumption for desktop GPUs.

DOI: 10.2312/eurp.20171166

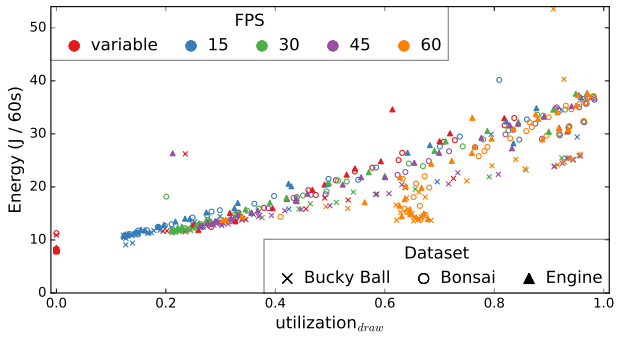


Figure 2: *Power consumption w.r.t. utilization, measuring draw-to-screen (OpenGL) in addition to bare frame computation (OpenCL), i.e. same data as in Fig. 1c but including draw-to-screen overhead.*

2. Methods and implementation

As a basis for our measurements, we use the Intrinsyc Open-Q 820 Development Kit [Int] running Android 6.0 in combination with an ARM Energy Probe [ARMa] (Fig. 1). This setup allows us to measure the overall power consumption of the system's core components: a Qualcomm Snapdragon 820 SoC (including an Adreno 530 GPU), memory, and communication hardware. We implemented a standard volume raycasting algorithm in an Android app using front-to-back compositing and early ray termination. For the sake of comparison, we implemented two backends for our raycasting loop, (1) using an OpenCL 1.2 compute framework kernel and (2) using an OpenGL fragment shader and the rendering pipeline. The app can receive parameters via the Android Debug Interface and start the render view with a pre-selected parameter set. The energy usage is recorded with ARM Streamline [ARMb]. The ratio of the time that the GPU is actively working has a strong impact on power consumption, and we denote it as *utilization* in the following. It is defined as sum of frame execution times of all measured frames in the sequence, divided by the total measurement time. We use the system's sleep function to achieve target frame rates and to circumvent the v-sync of Android. We employ two different methods to measure frame times (which accordingly has an impact on utilization): utilization$_{CL}$ measures the kernel execution time, and is limited to the OpenCL implementation (e.g., Fig. 1c); utilization$_{draw}$ measures one pass through the rendering loop, i.e. incorporating the OpenGL part and all overhead (e.g., Fig. 2).

3. Results and discussion

We ran multiple tests with different sets of parameters, including various viewport resolutions, frame rates, raycasting step sizes and datasets. Additionally, we ran each parameter combination with an interaction script (sequence of recorded user interactions, including rotation and zoom) as well as a constant view. All combinations resulted in a total of 450 test cases. In the case of variable frame rates, we only render a new frame, when the recorded user interaction changes the camera view. Renderings of the datasets are shown in Fig. 1a (Bonsai, Engine) and b (Bucky Ball). We evaluate all cases with our OpenCL as well as the OpenGL implementation.

OpenCL implementation. Fig. 2 shows the total energy consumption with respect to utilization (including drawing to screen overhead) for all configurations colored by FPS target. All data points are based on a 60*s* measurement period. The plot shows a roughly linear correlation between activity and power consumption. Measurements with almost zero activity are from cases with variable frame rates and constant view, meaning that only one frame is rendered during the whole time. There is an observable drifting-apart of the data points with increasing activity and a noticeable cluster of 60 FPS measurements at an activity of around 60%. There are a few outliers that do not seem to have a patter in terms of a specific parameter configuration and are probably caused by background tasks. To further investigate the observed cluster, we use the OpenCL kernel runtime based utilization$_{CL}$, which is shown in Fig. 1c. As can be seen, the cluster is non-existent using this measurement method. Accordingly, the source of this is likely to be found in displaying the rendering result with OpenGL (in detail, we assume that this is due to the context sharing of the two APIs and/or the system managed v-sync). Fig. 1c and the form-encoding in Fig. 2 show the reason for the drifting apart of the measurements for higher activity. Here, we can observe a clear linear trend for each dataset, which we attribute to cache coherency that is typically influenced by volume resolution and early ray termination. All other investigated parameters do not show special patterns in our utilization to power consumption diagram, leading us to the assumption that they only influence frame execution times and not the power consumption directly.

OpenCL vs. OpenGL. We compared our OpenCL implementation with an OpenGL version using fragment shaders. For the OpenGL variant, we could not determine exact frame execution times comparable to OpenCL kernel times. Therefore, we used utliziation$_{draw}$ as the basis for comparison (Fig. 1d). The plot shows the relation of OpenCL to OpenGL power consumption, with each data-point representing a parameter configuration. Linear regression yields the result that the OpenGL version uses around 25% less energy than the OpenCL version. The only correlation we found w.r.t. different parameter combinations is the case with variable frame times and a constant view that has almost no difference between OpenCL and OpenGL (only one frame is drawn). A qualitative analysis of the data showed no sign of the OpenGL version running particularly faster than OpenCL. However, we could not determine the exact frame execution times for the OpenGL variant and therefore can only assume that the increased energy usage when using OpenCL is due to the introduced overhead of using both APIs and/or inferior optimization for our mobile platform (because of the lack of OpenCL integration into the Android SDK).

Conclusion. Various parameter setups influence the energy consumption during volume rendering mainly through influencing computation time. However, different datasets show a distinctive linear trend in power consumption. Reducing computational time and limiting the FPS seems to reduce energy consumption consistently for the examined application, as is using OpenGL instead of OpenCL.

Acknowledgments

The authors would like to thank the German Research Foundation (DFG) for supporting the project within project A02 of SFB/Transregio 161 and the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[AJM*14] ARAGON E., JIMÉNEZ J. M., MAGHAZEH A., RASMUSSON J., BORDOLOI U. D.: Pattern matching in opencl: Gpu vs cpu energy consumption on two mobile chipsets. In *Proceedings of the International Workshop on OpenCL 2013 & 2014* (New York, NY, USA, 2014), IWOCL '14, ACM, pp. 5:1–5:7. URL: http://doi.acm.org/10.1145/2664666.2664671, doi:10.1145/2664666.2664671. 1

[ARMa] ARM: ARM Energy Probe. URL: https://developer.arm.com/products/software-development-tools/ds-5-development-studio/streamline/arm-energy-probe. 2

[ARMb] ARM: ARM Streamline. URL: https://developer.arm.com/products/software-development-tools/ds-5-development-studio/streamline. 2

[CDT09] COLLANGE S., DEFOUR D., TISSERAND A.: *Power Consumption of GPUs from a Software Perspective*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 914–923. URL: http://dx.doi.org/10.1007/978-3-642-01970-8_92, doi:10.1007/978-3-642-01970-8_92. 1

[Int] INTRINSYC: Open-Q 820 Development Kit. URL: https://www.intrinsyc.com/snapdragon-embedded-development-kits/snapdragon-820-development-kit/. 2

[JGDAM12] JOHNSSON B., GANESTAM P., DOGGETT M., AKENINE-MÖLLER T.: Power efficiency for software algorithms running on graphics processors. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics* (Aire-la-Ville, Switzerland, Switzerland, 2012), EGGH-HPG'12, Eurographics Association, pp. 67–75. URL: http://dx.doi.org/10.2312/EGGH/HPG12/067-075, doi:10.2312/EGGH/HPG12/067-075. 1

[MDZD09] MA X., DONG M., ZHONG L., DENG Z.: Statistical power consumption analysis and modeling for gpu-based computing. In *Proceeding of ACM SOSP Workshop on Power Aware Computing and Systems (HotPower)* (2009). 1

[MLC06] MOCHOCKI B., LAHIRI K., CADAMBI S.: Power analysis of mobile 3d graphics. In *Proceedings of the Conference on Design, Automation and Test in Europe: Proceedings* (3001 Leuven, Belgium, Belgium, 2006), DATE '06, European Design and Automation Association, pp. 502–507. URL: http://dl.acm.org/citation.cfm?id=1131481.1131617. 1