An Effective Viewport Resolution Scaling Technique to Reduce the Power Consumption in Mobile GPUs

Imjae Hwang[1], Hyuck-Joo Kwon[2], Ji-Hye Chang[1], Yeongkyu Lim[3], Cheong Ghil Kim[4] and Woo-Chan Park[1]

[1] Department of Computer Engineering, Sejong Univeristy
Seoul, Korea
[e-mail: {ijhwang, jhjang}@rayman.sejong.ac.kr, pwchan@sejong.ac.kr]
[2] Department of Electrical and Electronic Engineering, Yonsei Univeristy
Seoul, Korea
[e-mail: hyuckjookwon@yonsei.ac.kr]
[3]LG Electronics
Seoul, Korea
[e-mail: postrain70@gmail.com]
[4]Department Of Computer Science, Namseoul University
Cheonan, Choongnam, Korea
[e-mail:cgkim@nsu.ac.kr]
*Corresponding author: Woo-Chan Park

Received September 1, 2016; revised February 28, 2017; accepted April 29, 2017; published August 31, 2017

Abstract

This paper presents a viewport resolution scaling technique to reduce power consumption in mobile graphic processing units (GPUs). This technique controls the rendering resolution of applications in proportion to the resolution factor. In the mobile environment, it is essential to find an effective resolution factor to achieve low power consumption because both the resolution and power consumption of a GPU are in mutual trade-off. This paper presents a resolution factor that can minimize image quality degradation and gain power reduction. For this purpose, software and hardware viewport resolution scaling techniques are applied in the Android environment. Then, the correlation between image quality and power consumption is analyzed according to the resolution factor by conducting a benchmark analysis in the real commercial environment. Experimental results show that the power consumption decreased by 36.96% on average by the hardware viewport resolution scaling technique.

Keywords: Mobile GPU, GPU Power Consumption, FBO, Hardware Scaler, Viewport, Resolution

https://doi.org/10.3837/tiis.2017.08.009 **ISSN : 1976-7277**

1. Introduction

With the rapid deployment of mobile device technology, mobile devices have become a necessity in modern society. Mobile devices using a small embedded battery for their power sources have been also prevalent due to the technology development of semiconductor integrated circuits (ICs). The development of semiconductor ICs doubles every 18 months according to Moore's Law, but the energy density of a lithium secondary battery increases by only 5% every 12 months. If the battery's energy density maintains this development speed, it will double only by the year 2030 after 14 years from now [1]. Because the development speed of battery is very slow compared to that of a semiconductor IC, effective methods to maximize the battery efficiency with limited power have become important.

The key component for mobile devices is application processors (APs), which consist of CPUs, GPUs, and other units for applications running on the Aps, such as games, browser, UI, etc. As the area occupied by GPUs in APs has increased compared to CPUs, GPUs have become one of the major power-consuming processing units.

Studies on mobile low power consumption have become important due to limitation of development of battery energy density and high-resolution graphic processing [2]. The resolution is continuously increasing to HD, FHD, QHD, and even 4K and higher, which makes power hungry for mobile devices. The high resolution also increases graphic rendering workload and the framebuffer memory bandwidth. Furthermore, the complexity of various applications used by GPUs also increases because of standard APIs and game engines. For this reason, mobile GPUs' power efficiency has become critical and is being studied [3][4][5], and studies on prediction of power consumption through an analytic model of mobile GPUs' power consumption are being progressed [6][7][8].

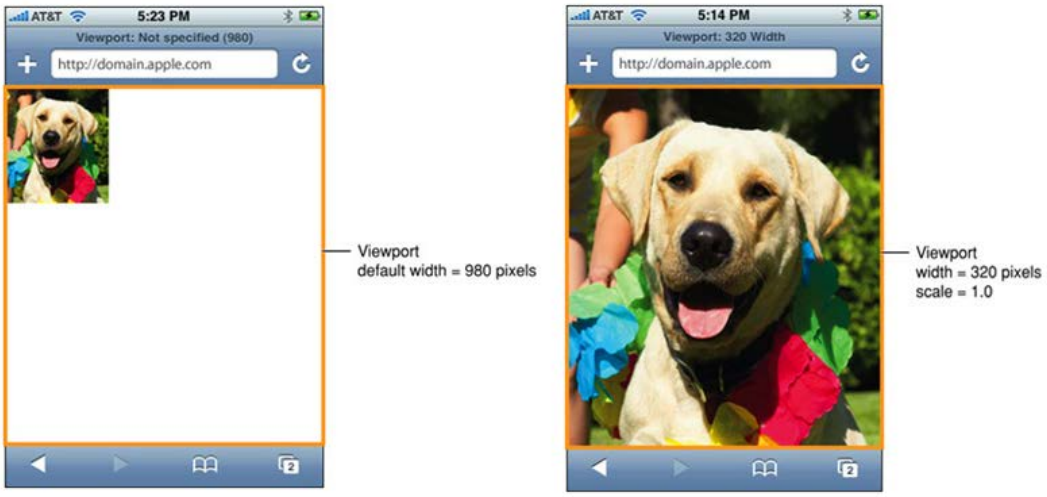

Fig. 1. Screens before and after using Viewport [9]

K. Nixon et al. introduced a resolution and frame-rate scaling method to reduce the power consumption of mobile GPUs in 2014 [10]. The suggested technique downscales the resolution which is higher than human visual acuity to a lower resolution with hardly recognizable changes. Other related studies include the Game Tuner developed by Samsung

Electronics [11] and the dynamic resolution rendering developed by Intel [12]. Samsung Electronics' Game Tuner is only supported by some mobile devices and game applications. Intel's dynamic resolution rendering conducted resolution scaling with a framebuffer object (FBO), which made low power rendering possible.

In this paper, at first, image quality degradation according to the various resolutions is analyzed by applying a software viewport resolution scaling (VRS) technique. Then, through the result of the analysis, a resolution factor that can minimize both image quality degradation and power consumption is proposed. Finally, we apply the proposed resolution factor into hardware VRS. For this purpose, an environment for accessing an Android is constructed, and the software and hardware VRS techniques are presented. To find the optimal resolution factor to minimize image quality degradation, the peak signal-to-noise ratio (PSNR) is calculated by applying software VRS technique and Qualcomm's Trepn Profiler is used to measure reducing the power with hardware VRS. Subsequently, the correlation between the image quality and power consumption is analyzed according to the resolution factor. As the result, the optimal resolution factor is found to be 50%, and the adaptation of this resolution factor reduced the power consumption to the 36.96% on average by the hardware VRS technique. For the experimental environment, Nexus 5 with the Adreno 330 GPU is chosen as the test mobile device. To apply the VRS technique, the Momo and Rolling Ball benchmarks are used, which are provided by the GFx game engine.

The remainder of this paper is organized as follows. Section 2 describes the concept and previous studies on VRS. Section 3 describes VRS with the proposed software/hardware scaler. Section 4 describes the experimental environment and results of the proposed technique. Finally, Section 5 outlines the conclusion of this paper and future studies.

2. Related Work

Previous studies on viewport resolution include K. Nixon et al. [10], which introduced resolution and frame rate scaling methods for reducing the power consumption of mobile GPUs. Other previous studies include the Game Tuner application [11] developed by Samsung Electronics, and dynamic resolution rendering [12], which runs in the OpenGL|ES 2.0 environment, developed by Intel. In this section, VRS as well as Samsung Electronics' Game Tuner application and Intel's dynamic resolution rendering will be examined.

2.1 VRS

Visual acuity refers to the ability to distinguish very small differences or the minimum angular distance that can differentiate two points [13]. In particular, human visual acuity refers to the viewing angle that can distinguish target objects. Minimizing the viewport resolution in consideration of human visual acuity is very effective for lower power because it greatly decreases the frame rendering computation load. However, quality degradation coming from scaling must be analyzed by considering human visual acuity. **Fig. 2** shows an example of VRS.

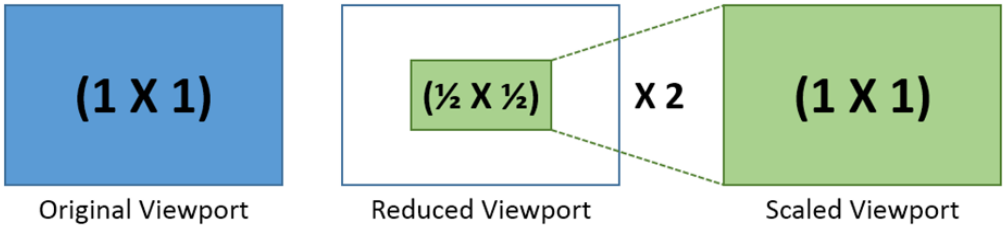

Fig. 2. Example of VRS

VRS methods are divided into software and hardware methods. The software scaling generally creates an off-screen texture through the framebuffer object and performs viewport scaling, whereas the hardware method performs scaling in the viewport through the hardware scaler.

2.2 Samsung Electronics' Game Tuner

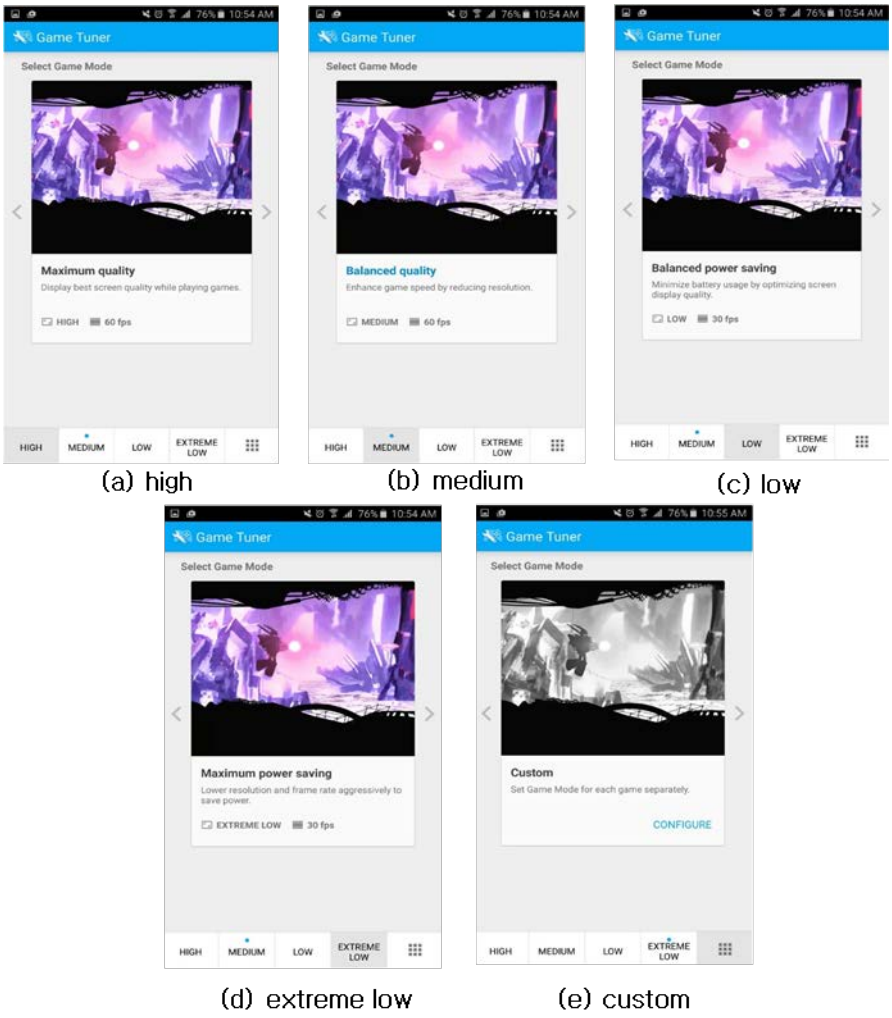

Fig. 3. Screens when Game Tuner is running in five modes

Samsung Electronics provides the Game Tuner application, which allows users to directly control heating and power consumption by adjusting the resolution and frames per second (FPS) of mobile games. Game Tuner runs a resolution that is lower than that of the mobile device to control performance, battery consumption, and temperatures. **Fig. 3** shows the screens of Game Tuner in five modes.

The five modes of Game Tuner are as follows: the high mode (a) with 2560×1440 QHD resolution and 60 fps; the medium mode (b) with 1920×1080 FHD resolution and 60 fps; the low mode (c) with 1280×720 HD resolution and 30; the extreme low mode (d) with 720×480 SD resolution and 30 fps; the custom mode (e).

Fig. 4 shows screens for comparing the modes of Game Tuner on a mobile game. When the Game Tuner were activated, the outlines and texts of the graphics looked blurred. Furthermore, the battery temperatures can be controlled by lowering the resolution.

Game Tuner is only supported by Galaxy S6, Galaxy S6 Edge, Galaxy S6 Edge+, and Galaxy Note 5 for supported games in the custom mode list, including Clash of Kings, Hay Day, Minion Rush, Subway Surf, Temple Run2, Hearthstone, and Clash of Clans.

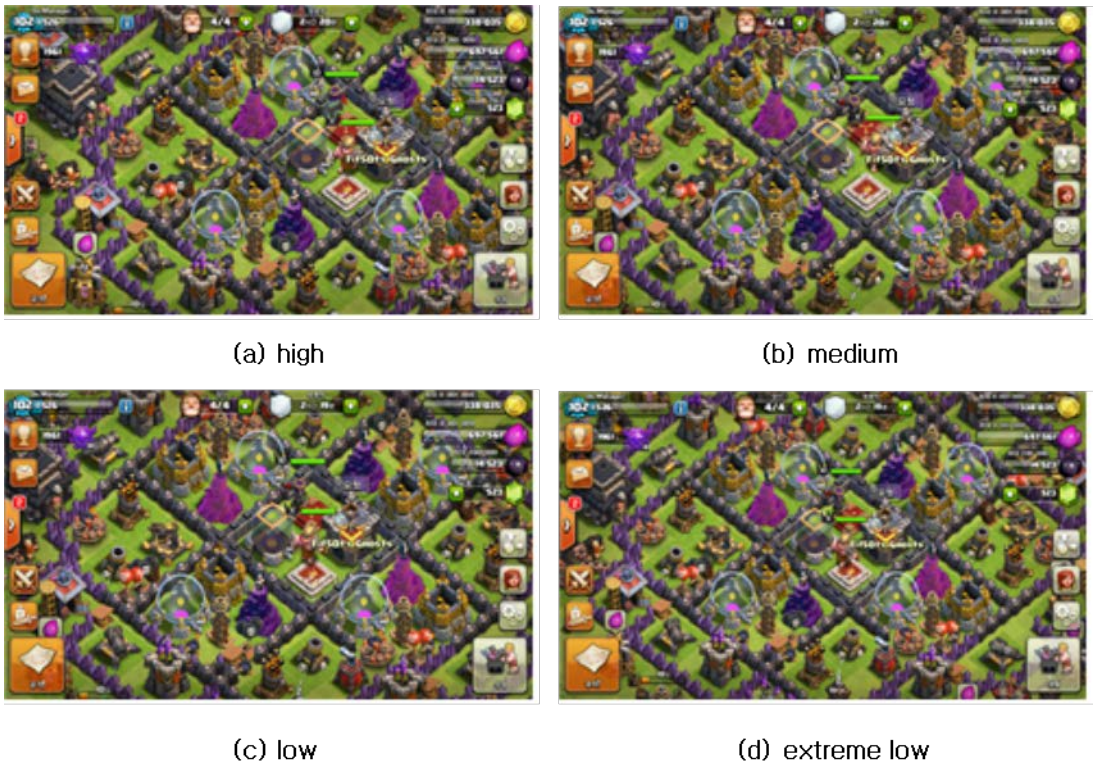

Fig. 4. Comparison of mobile game screens when Game Tuner is running

2.3 Intel's Dynamic Resolution Rendering

A major bottleneck point in game and graphic workloads is fragment or pixel shader processing. The fragment shader calculates the effects of lighting, texture sampling, and post-processing. Thus, much processing time is spent in the final color calculation step for each pixel of the screen, which increases the overall power consumption. To reduce this overhead, Intel developed dynamic resolution rendering [12] to which the VRS technique using FBO was applied.

Intel's dynamic resolution rendering can dynamically render the resolution using FBO. This technique is appropriate for measuring the GPU power consumption according to the viewport resolution setting because it can dynamically control resolution. **Fig. 5** shows the resolution scaling process.

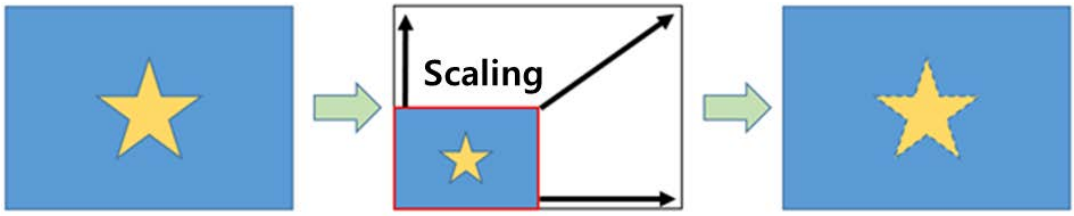

Fig. 5. Resolution scaling process

3. VRS through the Proposed Software/Hardware Scaler

This section presents software VRS using FBO and hardware VRS using a hardware scalar.

3.1 Software VRS using FBO

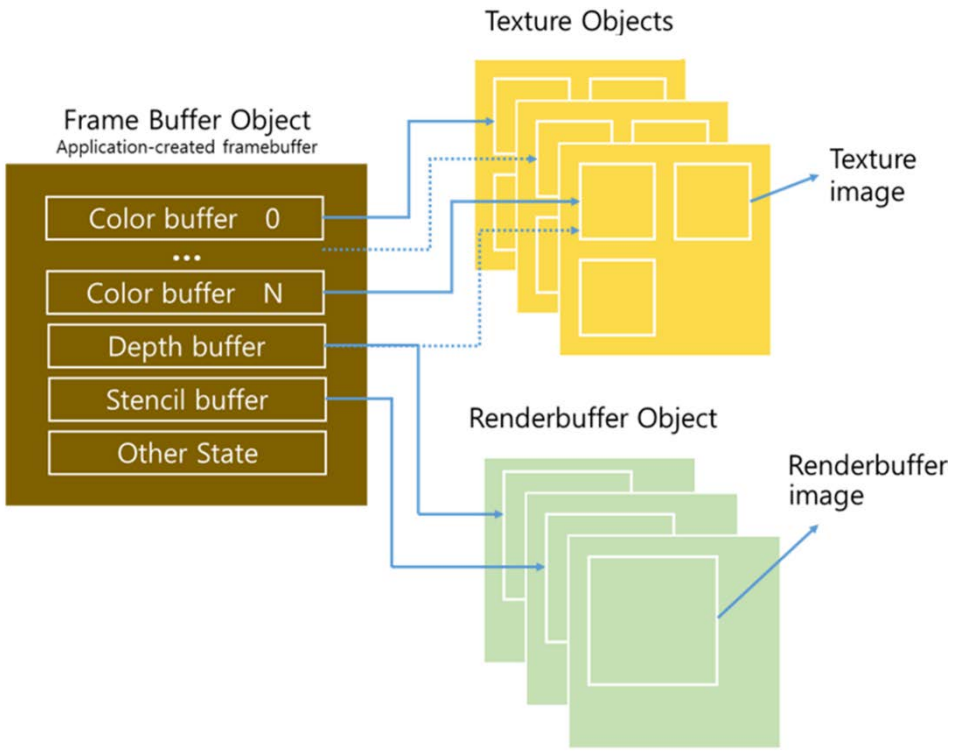

Fig. 6. FBO and its relationship with texture objects and renderbuffer objects [14]

OpenGL renders images in the buffer and brings the result images to the context and displays them on the screen. The framebuffer is the set of buffers in which the final output of the OpenGL rendering is stored. In computing, a buffer is a high-speed memory in which data are temporarily stored before the data are transmitted between the main memory and

peripherals to resolve the difference in the data transfer speed between them. Among these buffers, the framebuffer is a memory for temporarily storing image data to be shown on the display in the raster scan method. In the OpenGL rendering pipeline, geometry and texture data are transformed and mapped respectively before being rendered on the screen as 2D pixels.

The framebuffer object (FBO) is an OpenGL extension for flexible off-screen rendering, including texture rendering. FBO, which is similar to the renderer target model in DirectX, is used in OpenGL for efficiency and ease of use. **Fig. 6** shows FBO and its relationship with texture objects and renderbuffer objects.

The software implementation of VRS using FBO is as follows. First, we add a dynamic resolution render FBO and a shader program to the OpenGL | ES 2.0 environment. Second, we create an off-screen render target and attach the target to the shader program; then, we perform rendering on the off-screen render target according to the resolution factor. Finally, we perform texture mapping of the off-screen rendering output in accordance with the screen to express the resolution according to the resolution factor.

3.2. Hardware VRS Using a Hardware Scaler

As images can be rendered to various sizes regardless of display size, the proportion of pixels on images and display may not match. A hardware scaler is appeared to address this problem. The role of the hardware scaler is to solve the problems caused by increasing or reducing images by force and to change the image scale to a desired scale. Using a hardware scaler can minimize both power consumption and artifacts.

The hardware scaler can be applied to the application using it at the JAVA code level. The method involves scaling the viewport resolution using setFixedSize() during the init() of the GLSurfaceView, which inherits the View and SurfaceView. **Fig. 7** shows the operation process of the VRS through the hardware scaler. When the setFixedSize() is called during the initialization process of the GLSurfaceView, VRS is carried out by the setFixedSize() and the output is stored in the framebuffer. At this time, VRS is performed using the hardware scaler. **Fig. 8** shows the scaling output by reducing the 1776×1080 resolution of the target device Nexus 5 to a quarter using the method in **Fig. 7**.

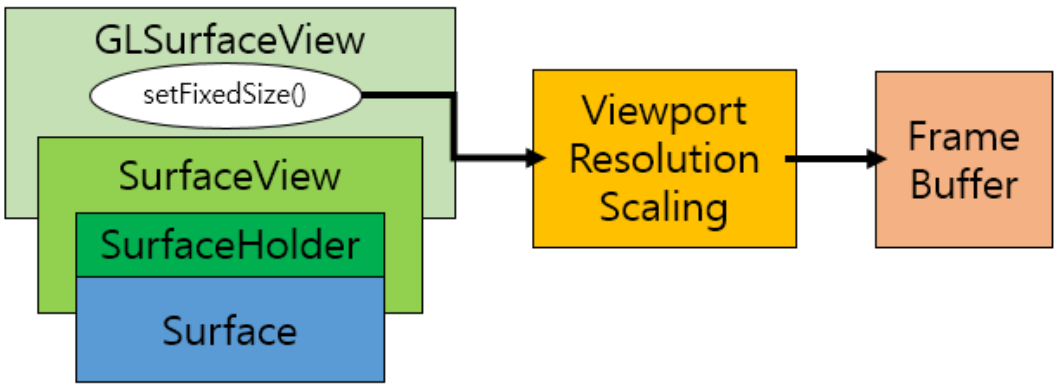

Fig. 7. Overview of VRS using the hardware scaler

(a) Momo benchmark to which the hardware scaler was applied

(b) Rolling ball benchmark to which the hardware scaler was applied

Fig. 8. Results of the hardware scaler application

4. Experimental Results and Analysis

This section describes the experimental environment and suggests the optimal resolution factor through this environment.

4.1 Experimental Environment

The host environment for this study is i7-4770 CPU 3.40GHz, NVIDIA GeForce GTX 750 Ti, and the Windows 7 64-bit operating system. The target device is Nexus 5, which is one of the Android reference devices. Nexus 5 includes the Snapdragon 800 MSM8974 SoC chip, which has the Krait 400 Quad-core 2.26 GHz CPU and the Adreno 330 450 MHz GPU, and 2048 MB of memory. The operating system of this target device is Android 4.4.2 (Kitkat).

The target device of Game Tuner is Samsung Galaxy S6, which includes Exynos 7420 SoC chip and 3072 MB of memory. Samsung Galaxy S6 also has 2560×1440 screen resolution. The operating system of this device is Android 6.0.1 (Marshmallow). In the image quality measurement, the images are compared for verifying pixel loss or difference. The PSNR is calculated for assessing image loss information as like in video loss compression.

In the power consumption measurement experiment, Trepn Profiler [15], which is the freeware profiler application of Qualcomm, is used [**Fig. 9**]. Trepn Profiler is designed for the power and performance profiling of mobile devices equipped with Qualcomm Snapdragon processors. We used version 5.1 for the experiment, and the DATA POINTS environment of Trepn Profiler is switched off, except for battery power.

Fig. 9. Qualcomm's Trepn Profiler

4.2 VRS Application Benchmark Using FBO

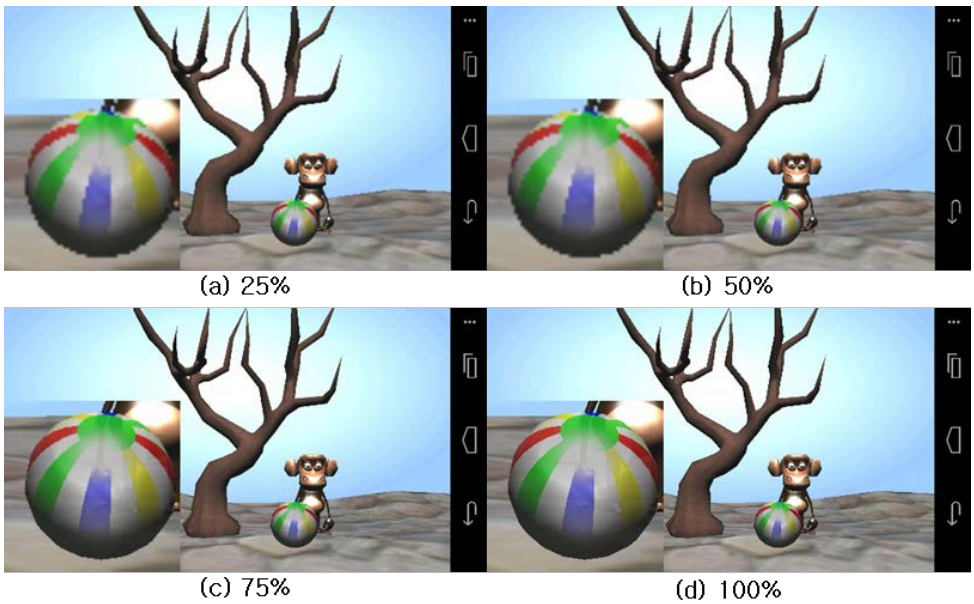

Fig. 10. Comparison of Momo benchmark images according to resolution

This section describes benchmarks implemented by applying VRS to the GFx game engine [16]. The procedure is to add the FBO and shader source code in the OpenGL|ES 2.0 environment, create an off-screen render target, and add a shader.

The resolution can be expressed by performing rendering to the off-screen render target according to the resolution factor and by scaling with shader. **Fig. 10** shows screens generated by applying VRS to the Momo benchmark of a static scene, which is an example of the GFx game engine. The images in **Fig. 10** have different qualities according to the resolution factor.

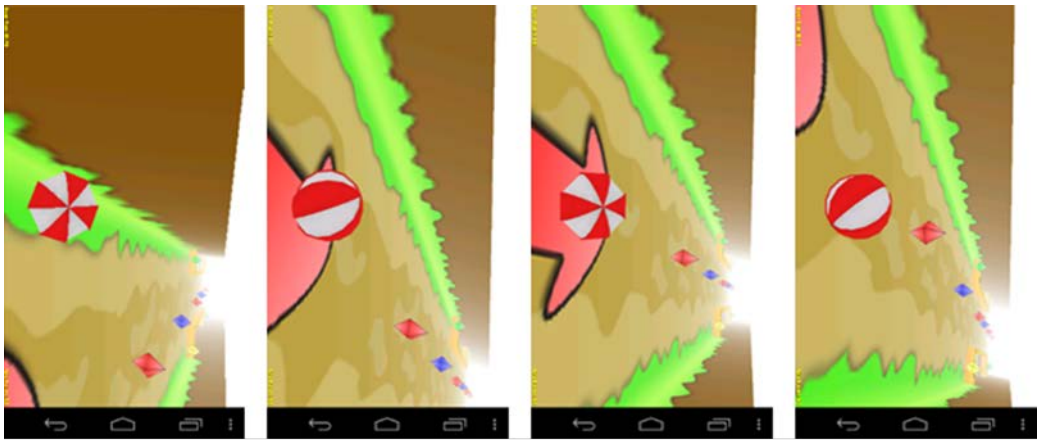

Fig. 11. Comparison of the Rolling ball benchmark images according to resolution

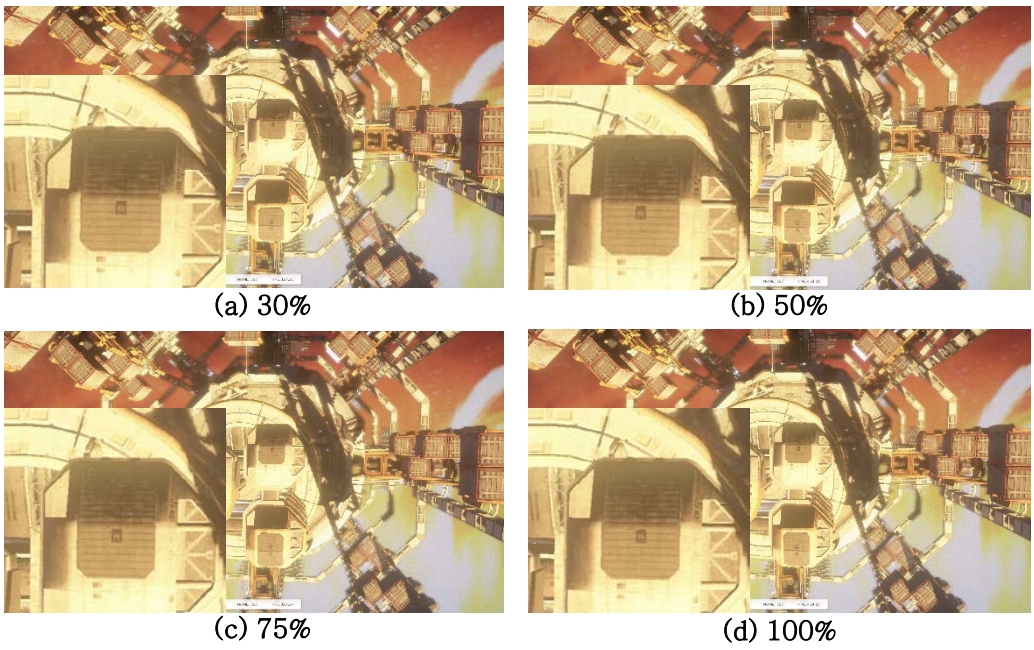

Fig. 12. Comparison of the 3DMark images which Game Tuner was applied

Fig. 11 shows screens generated by applying VRS to the Rolling ball benchmark of a dynamic scene unlike the Momo benchmark of a static scene. The application method is to

modify the framebuffer object in the source code of the Rolling ball benchmark of the GFx game engine, which is editable. After adding a render target, an off-screen render target is created and rendered according to the resolution factor. The scaling is performed using a full-screen quad shader. **Fig. 11** shows the dynamic scenes of the Rolling ball benchmark with resolutions changed according to four resolution factors.

Fig. 12 and **Fig. 13** shows screens generated by applying Game Tuner to 3DMark [17] and Angry birds [18] for comparison. Resolution factors used in Game Tuner are 100%, 75%, 50% and 30% because 30% resolution factor is minimum resolution factor that Game Tuner provides.

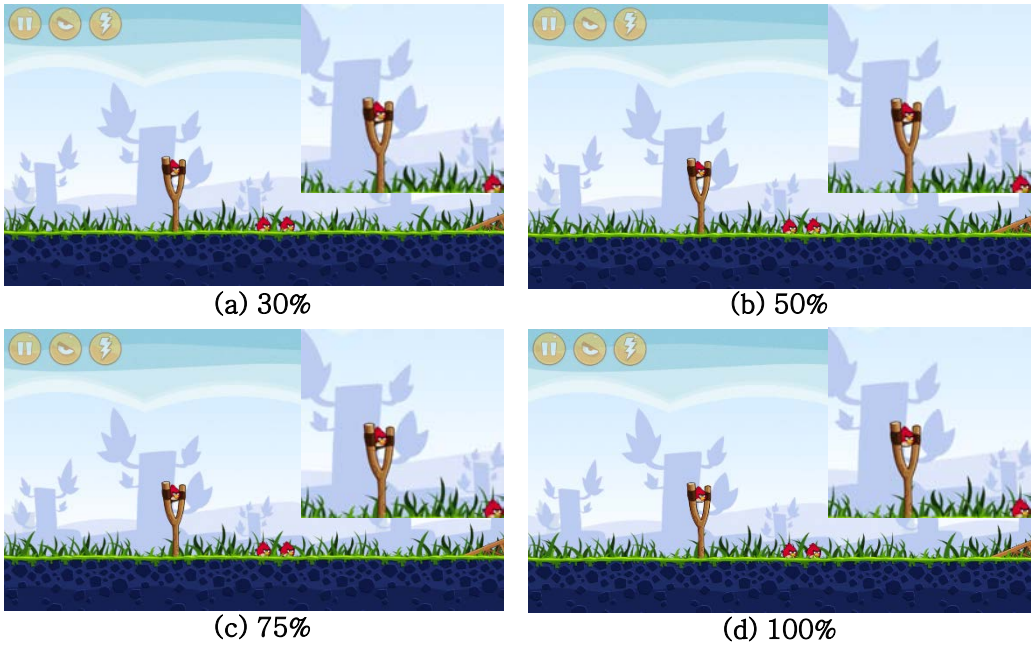

Fig. 13. Comparison of the Angry birds images which Game Tuner was applied

4.3 Results and Analysis

4.3.1 PSNR According to the Resolution Factor

Fig. 14 shows the calculated PSNR values with 75~20% resolution factors of two benchmarks which VRS was applied. For a loss image with a bit depth of 8 bit, 30~50 dB PSNR is an ordinary value. A higher PSNR value indicates a smaller difference from the original [19]. When the resolution factor is 20~45% in **Fig. 14**, the PSNR of the Momo benchmark is 29.64~31.65dB, and the PSNR of the Rolling ball benchmark is 28.23~29.93dB. When the resolution factor is 50~75%, the PSNR of the Momo benchmark is 32.05~35.27dB, and the PSNR of the Rolling ball benchmark is 30.76~35.60dB. Therefore, images with 50% or higher resolution factors may be considered as ordinary quality images.

Fig. 15 shows the comparison of images with 75%, 50%, and 25% resolution factors. The green dots indicate different pixel values at the same position of two images. The number of green dots increases as the resolution factor decreases. This indicates an increasing difference between two images.

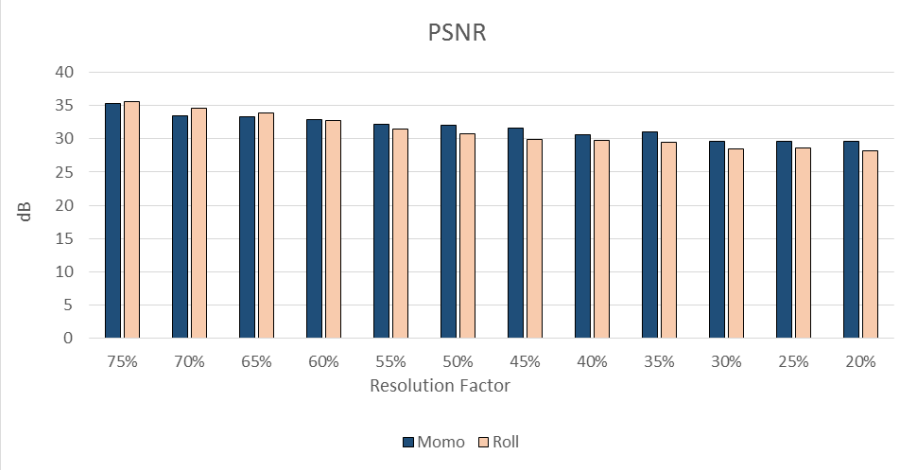

Fig. 14. Comparison of the PSNR between the Momo and Rolling Ball benchmark images

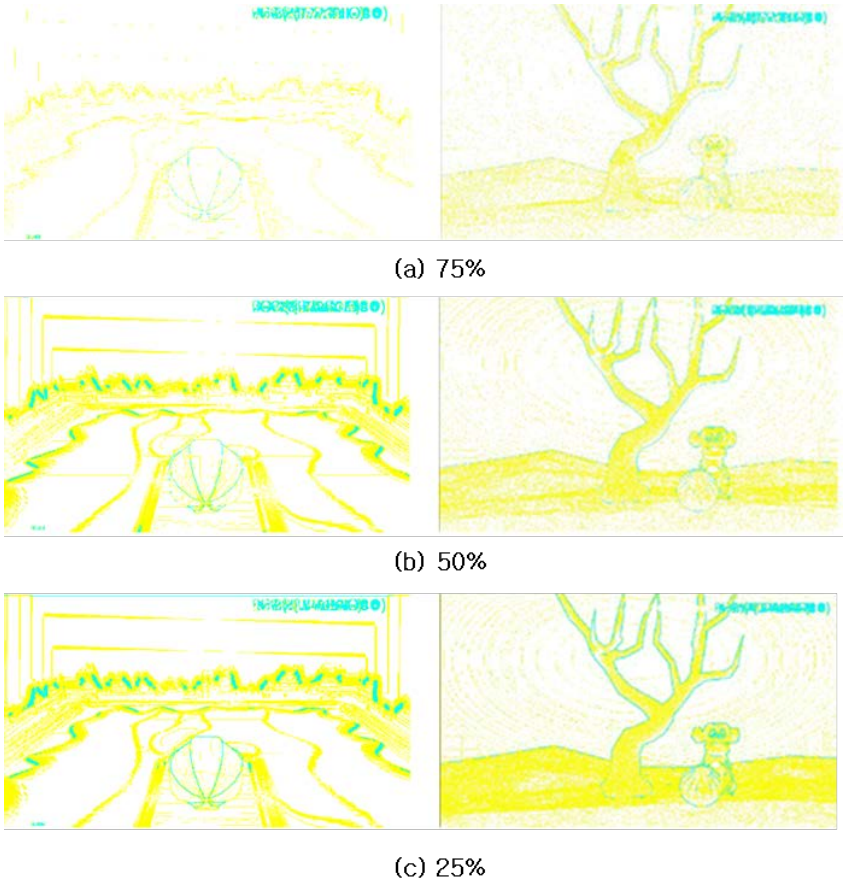

Fig. 15. Comparison of the Momo and Rolling Ball benchmark images

Fig. 16 shows the calculated PSNR values with 75%, 50%, 30% resolution factors of four benchmarks(Momo, Rolling ball, 3DMark and Angry birds) which Game Tuner was applied.

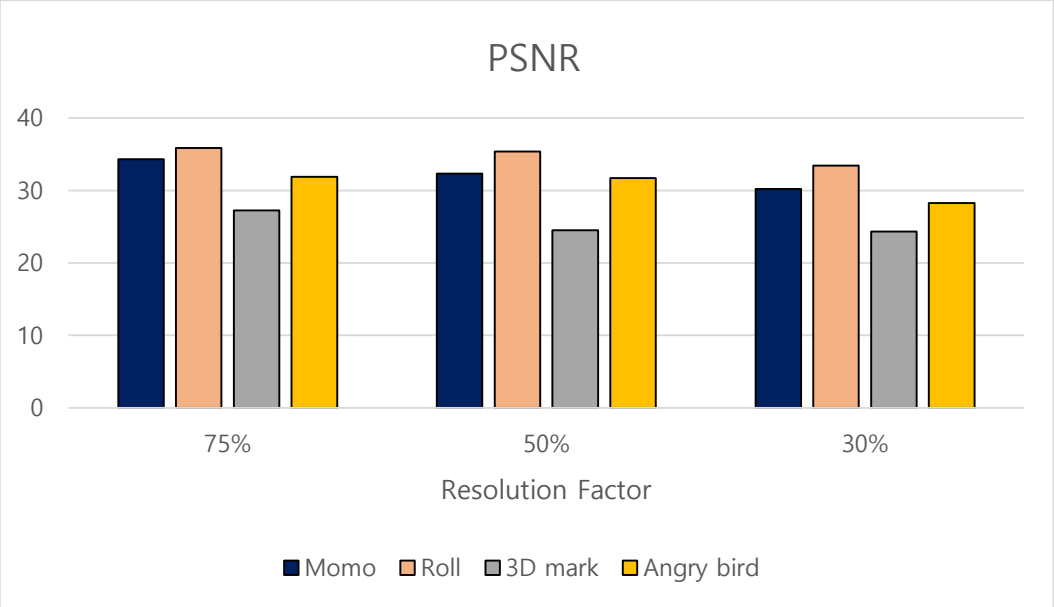

Fig. 16. Comparison of the PSNR for images which Game Tuner was applied

4.3.2 Power Consumption by VRS using FBO and Hardware Scaler

Table 1 illustrates the analysis of the power of the benchmarks to which VRS is applied using a hardware scaler and **Fig. 17** shows the power consumption reduction rates for each resolution factor. The power consumption reduction rates of 100%, 75%, 50%, and 25% resolution factors are calculated using the Trepn Profiler. The VRS technique using a hardware scaler shows greater reduction rates of power consumption than the VRS technique using the framebuffer object.

Table 1. Power measurement results for VRS using the hardware scaler

Resolution Factor (%)	100		75		50		25	
Benchmark	Momo	Roll	Momo	Roll	Momo	Roll	Momo	Roll
Average Power Consumption (mWh)	134.17	119.96	95.39	85.74	79.45	80.20	72.47	77.51
Power Consumption Reduction Rate (%)	0	0	28.90	28.52	40.78	33.14	45.98	35.38

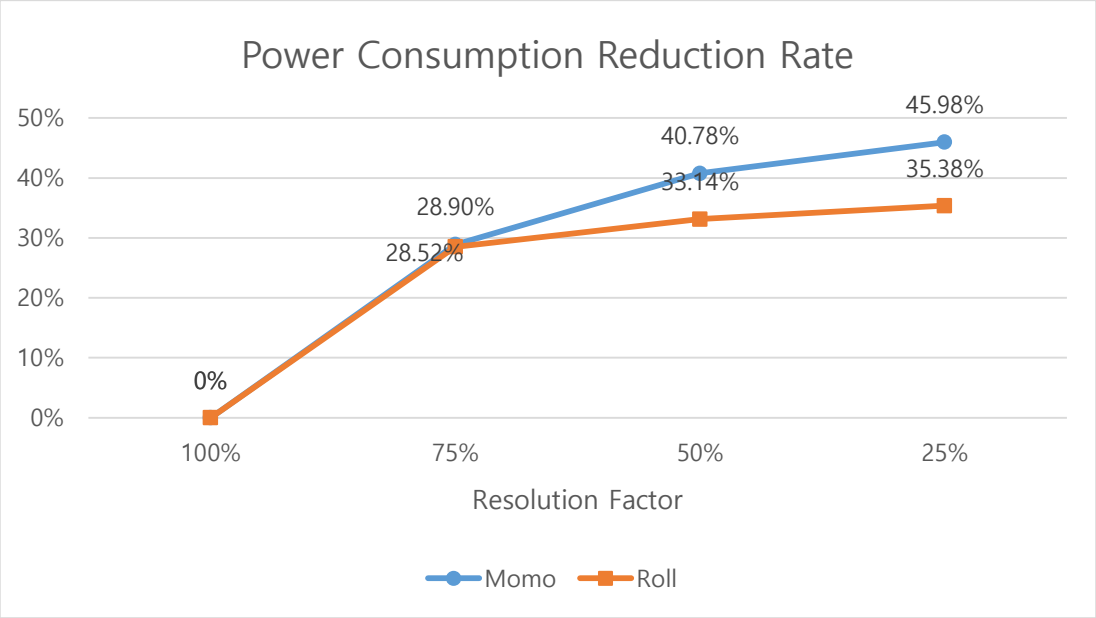

Fig. 17. Power Consumption Reduction Rate

Table 2 illustrates the analysis of the power of four benchmarks which Game Tuner is applied and **Fig. 18** shows the power consumption reduction rates for each resolution factor. The power consumption reduction rates of 100%, 75%, 50%, and 30% resolution factors are also calculated using the Trepn Profiler. Experiment results show that each power consumption reduction rates are 2.00% ~ 9.67% for Momo benchmark, 1.83% ~ 13.12% for Rolling ball benchmark, 1.21% ~ 30.33% for 3DMark and 1.37% ~ 5.58% for Angry birds, respectively.

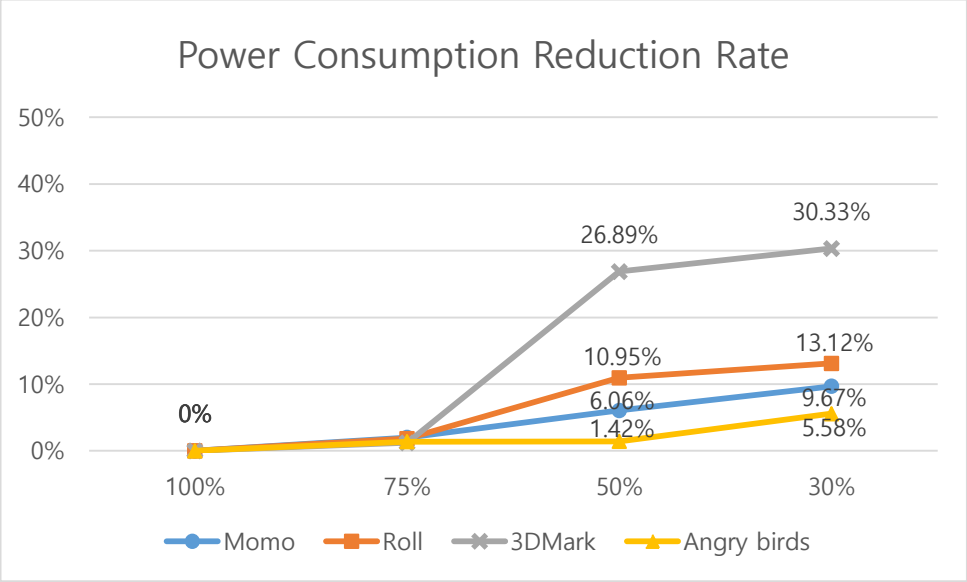

Fig. 18. Power Consumption Reduction Rate for Game Tuner

Table 2. Power measurement results for Game Tuner

Resolution Factor (%)	Benchmark	Average Power Consumption (mWh)	Power Consumption Reduction Rate (%)
100	Momo	441.59	0
	Rolling ball	581.53	0
	3DMark	1085.14	0
	Angry birds	716.62	0
75	Momo	432.77	2.00
	Rolling ball	570.91	1.83
	3DMark	1072.05	1.21
	Angry birds	706.80	1.37
50	Momo	414.83	6.06
	Rolling ball	517.86	10.95
	3DMark	793.39	26.89
	Angry birds	706.42	1.42
30	Momo	398.91	9.67
	Rolling ball	505.24	13.12
	3DMark	755.97	30.33
	Angry birds	676.63	5.58

5. Conclusion and Future Work

In this paper, software and hardware VRS techniques to reduce the power consumption of mobile GPUs are presented. It is important to find an efficient resolution factor because the resolution and GPU power consumption of a mobile device are in a trade-off relationship. For this purpose, an Android environment for applying the software and hardware VRS techniques is constructed and benchmarks are performed by applying the software and hardware VRS techniques in this environment. The images are compared, calculated PSNR, and measured the power consumption to analyze the correlation between the image quality and power consumption according to the resolution factor. As a result, the resolution factor that minimizes power consumption while maintaining the maximum image quality is found to be 50%. The average power consumption reduction rate of the hardware VRS technique applying this resolution factor is 36.96%.

In this paper, the results of the VRS techniques are determined through feedback. Therefore, the PSNR must be compared to the static rendering results, and the optimal resolution factor must be statically applied to the target application. In future work, it is necessary to compare the PSNR dynamically to find the resolution factor that minimizes the image quality degradation at the Android API level, and to maximize the hardware scaler efficiency by dynamically applying the resolution factor to the hardware scaler of the target application.

Acknowledgement

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2017-2016-0-00312) supervised by the IITP(Institute for Information & communications Technology Promotion)

References

[1] J. Paradiso and T. Starner, “Energy scavenging for mobile and wireless electronics," *IEEE Pervasive Computing*, Vol. 4, No. 1, pp. 18-27, Jan. 2005. Article (CrossRef Link)
[2] “Qualcomm Presentation - How to minimize the power consumption of your app.” Article (CrossRef Link)
[3] T. Akenine-Moller and B.Johnsson, "Performance per What?," *Journal of Computer Grahpics Techniques*, Vol.1, No.1, Oct. 2012.
[4] B. Johnsson and T. Akenine-Moller, "Measuring Per-Frame Energy Consumption of Real-Time Grahpics Applications," *Journal of Computer Graphics Techniques*, Vol.3, No.1, Mar. 2014.
[5] B. Johnsson, P. Ganestam, M. Doggett, and T. Akenine-Moller, "Power Efficiency for Software Algorithms running on Graphics Processors," in *Proc. of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics. Eurographics Association*, pp.67-75, June. 2012.
[6] Y. G. Kim, M. Kim, J. M. Kim, M. Sung, and S. W. Chung, "A Novel GPU Power Model for Accurate Smartphone Power Breakdown," *ETRI Journal*, Vol. 37, No. 1, pp. 157-164, Feb. 2015. Article (CrossRef Link)
[7] B. Mochocki, K. Lahiri, and S. Cadambi, “Power analysis of mobile 3D graphics,” in *Proc. of the conference on Design, automation and test in Europe: Proceedings. European Design and Automation Association*, pp. 502-507, Mar. 2006. Article (CrossRef Link)
[8] J. M. Vatjus-Anttila, T. Koskela, and S. Hickey, "Power consumption model of a mobile GPU based on rendering complexity," in *Proc. of Next Generation Mobile Apps, Services and Technologies (NGMAST), 2013 Seventh International Conference on*. IEEE, pp. 210-215, Sep. 2013. Article (CrossRef Link)
[9] “Viewport Resolution.” Article (CrossRef Link)
[10] K. Nixon, X. Chen, H. Zhou, Y. Liu, and Y. Chen, “Mobile GPU power consumption reduction via dynamic resolution and frame rate scaling," in *Proc. of the 6th USENIX conference on Power-Aware Computing and Systems. USENIX Association*, pp. 5-5, Oct. 2014.
[11] “Game Tuner.” Article (CrossRef Link)
[12] “Dynamic Resolution Rendering on OpenGL|ES 2.0.” Article (CrossRef Link)
[13] S. He, Y. Liu, and H. Zhou “Optimizing Smartphone Power Consumption through Dynamic Resolution Scaling,” in *Proc. of the 21st Annual International Conference on Mobile Computing and Networking, ACM*, pp. 27-39, Sep. 2015. Article (CrossRef Link)
[14] Simon Green. “The OpenGL Framebuffer Object Extension,” *Games Developers Conference*, http://download.nvidia.com/developer/presentations/2005/GDC/OpenGL_Day/OpenGL_FrameBuffer_Object.pdf, (2005).
[15] “Trepn Power Profiler.” Article (CrossRef Link)
[16] “GFx:: Game and Graphics Engine,” 3D. Article (CrossRef Link)
[17] “3DMark.” Article (CrossRef Link)
[18] “Angry birds.” Article (CrossRef Link)
F. Gmira, S. Hraoui, A. Saaidi, A. Oulidi, and K. Satori, "Securing the architecture of the JPEG compression by an dynamic encryption," *Intelligent Systems and Computer Vision (ISCV), IEEE*, pp. 1-6, Mar. 2015. Article (CrossRef Link)

Imjae Hwang received the B.S. degree in Internet engineering from Sejong University, Seoul, Korea in 2012. He is currently doctoral student in Computer engineering, Sejong University, Seoul, Korea. His current research interests include 3-D rendering processor, high performance computing, real-time ray tracing and mobile GPU.

Hyuck-Joo Kwon received the B.S., M.S., PhD. degrees in Computer engineering from Sejong University, Seoul, Korea in 2009, 2011, and 2016 respectively. He is a currently postdoctoral researcher in Electronical and Electronic engineering, Yonsei University, Seoul, Korea. His current research interests include 3-D rendering processor architecture, hardware acceleration, real-time ray tracing and mobile GPU.

Ji-Hye Chang received the B.S., M.S. degrees in Computer engineering from Sejong University, Seoul, Korea in 2012 and 2016 respectively. She is currently an engineer in Essys Corperation, Seoul, Korea. Her current research interest includes autonomous collaborative system.

Yeongkyu Lim received his B.S. degree from Kyungpook National University in 1997 and received M.S degree form Dept. of Computer Science at Korea University in 1999. In 2013, he received Ph. D from Dept. of Computer Science at Yonsei University, Seoul, Korea. He has been working on LG Electronics over 17 years. His research areas are Embedded Systems, HCI, Mobile GPU Architecture and GPU Computing.

Cheong Ghil Kim received the B.S. in Computer Science from University of Redlands, CA, U.S.A. in 1987. He received the M.S. and Ph.D. degree in Computer Science from Yonsei University, Korea, in 2003 and 2006, respectively. Currently, he is a professor at the Department of Computer Science, Namseoul University, Korea. His research areas include Multimedia Embedded Systems, Mobile AR, and 3D Contents.

Woo-Chan Park received the B.S., M.S., and Ph.D. degrees in Computer science from Yonsei University, Seoul, Korea, in 1993, 1995, and 2000, respectively. From 2001 to 2003, he was a Research Professor with Yonsei University. He is currently an Associate Professor of Computer engineering, Sejong University, Seoul. His current research interests include ray tracing processor architecture, 3-D rendering processor architecture, real-time rendering, advanced computer architecture, computer arithmetic, lossless image compression hardware, and application-specific integrated circuit design.