Effect of 3D Content Simplification on Mobile Device Energy Consumption

Jarkko M. Vatjus-Anttila, Timo Koskela, Seamus Hickey
Center for Internet Excellence
P.O. Box 1001
FI-90014 University of Oulu
email: <firstname.lastname>@cie.fi

ABSTRACT

Extensible 3D virtual spaces and their services are often too heavy for a mobile device to handle. The burden caused by such services is divided between extensive amounts of content, which need to be downloaded prior to using the service, and the complexity of the graphical reproduction process (rendering). In this paper, it is shown how texture optimization, texture compression and geometry optimization of such a content will affect the overall energy consumption of a mobile device during rendering. Results show that careful optimization of the 3D content for the mobile device gain energy savings of up to 40%.

Categories and Subject Descriptors

H.3.4 [**System and Software**]: Information networks

Keywords

Rendering, mobile device, energy consumption, detail reduction, open source

1. INTRODUCTION

There is an increasingly large number of Internet based services which are based on 3D graphics. Examples of such networked services include games, virtual spaces and other similar services. Users are increasingly using these services with mobile devices as well, as those devices improve in 3D processing power, which has been previously a privilege of laptop and workstation computers. However, due to highly heterogeneous types of the mobile devices, it is very difficult to create content for the 3D services in such a way that it will work optimally with all possible device types. This leads to a requirement to simplify the 3D content for each device type on demand-time [10]. However, in order to accomplish a good result on demand-time, it is important to understand the impact of different simplification methods on the mobile device.

In this research, the focus was entirely on local energy measurements of the mobile device. We wanted to see how controlled alteration of the 3D content affected the mobile device total energy consumption, once the device was put to do a certain rendering task. In general, we feel that in order to gain from content optimization, one needs first to understand the effects of the various methods for it. This research focuses on simplification of textures and geometry, of which the 3D based services are built from. The understanding of how the parameters affect the energy consumption would allow eventually building systems, which eventually optimize content directly for the mobile device. In this research, 1) textures are scaled down according to display size of the mobile device, 2) textures are forced to be in preselected compressed format and 3) geometry is reduced using polygonal simplification methods. Those methods are measured individually and also a combination of them is created to see their joint effect on the overall consumption. The 3D scene used in the testing was a 3D representation of a city. The city model consisted of nine street blocks, including roughly 500,000 vertices, 300,000 triangles and 35MB of uncompressed texture maps. In theory, it is expected that when increasing the display size and having a large amount of texture maps, it will result in a growing number of graphics processing unit (GPU) per-fragment operations and hence those parameters are expected to dominate the energy consumption.

The primary motivation for this research has been a lack of results in the field of overall energy consumption and mobile 3D graphics rendering. There exist a number of generic algorithm studies and some very detailed studies about narrow topics in the mentioned field, but overall, there seems to be a rather limited knowledge available on what kind of an effect the above mentioned methods have on the overall mobile device energy consumption. This paper will contribute there showing the real effects of the content optimization from the energy consumption perspective. The rest of the paper is organized as follows: First, there is a view on the previous research done in the field of mobile 3D graphics, as well as 3D content adaptation and algorithms. Second, the background for the measurement setup and the measurement setup itself are described in detail. Finally, measurements results are presented and conclusions are drawn.

2. PREVIOUS RESEARCH

Brown et al. [3] have studied the possibility to reduce detail of the given geometrics based on the object's importance on the device display. They show a method for forecasting

where the detail can be reduced in a way the human eye would not notice the difference. This is a good insight into what could be optimized away from the 3D scene. Pool et al. [7] have been researching methods on how shader algorithms implemented in the GPU can be optimized by reducing calculation precision to gain savings. By doing this, they have been able to gain savings of 10-20% of overall system power. However, lowering precision in the shading process is a very detailed change, which will result in lower rendering quality in some cases, and hence, it is not usable universally in all applications. Anand et al. [1] indicate how the utilization of the mobile device display should be done more efficiently by applying tone mapping on the rendered data, and hence, allow lowering of the display backlight intensity.

Cantlay [4] has done research on how the GPU is handling the different MipMap-levels during a rendering process. Their motivation was to find out whether it is possible to save video memory by making run-time optimization of the textures. The optimization was based on an analysis whether certain parts of the given textures were going to show at all in the final rendering result. If there were such parts in the textures, those could easily be left out during the optimization process. The paper summarizes how the presented methods were able to achieve up to 80% savings in the GPU texture memory utilization in certain applications without a noticeable decrease in the overall image quality

It is essential to understand how the mobile GPU processes the data, in order to be able to do correct simplifications for it. Mochocki et al. [2] have done research on how GPU power consumption varies with different kinds of rendering loads. They have measured how the power consumption is split between geometry processing stage, triangle setup stage and rendering stage. The study indicates that by correctly altering the GPU parameters of the rendering pipeline, savings in terms of rendering complexity are achievable, which eventually will lead to lower overall energy consumption of the GPU. The results can be applied to the content optimization process by acknowledging which kind of optimization methods should be used to favor the rendering process of the local GPU.

Using compressed textures during the rendering process is one of the optimizations, which could result in more optimal graphics reproduction. Having a texture asset in compressed format in the GPU memory during the rasterization process drives to a lower internal GPU bus bandwidth requirements. Among compressed texture formats, ETC1 is having a fixed compression ratio of 6:1, meaning a single pixel will reserve 4 bits of the GPU memory instead of 24 bits compared to an uncompressed 24-bit RGB image. Hence, during the rasterization process, sampling of a single texel is using 1/6 of bandwidth compared to the 24-bit RGB texture. Also, Ström et al. have shown how utilizing compressed texture formats, which are natively supported by the mobile hardware, are very suitable for rendering. Reasoning is that the compressed textures can be kept in their compressed form during rendering, hence allocating less video memory and causing less sampling bandwidth internally to the GPU. [8] [9]

For geometry, there are polygonal simplification methods

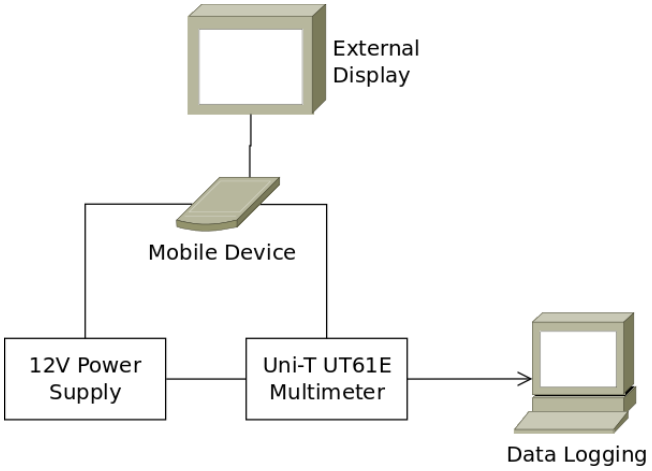

Figure 1: Measurement setup

available, as described by Lindstrom et al. [5]. In the method, a list of edge connections are calculated from the original mesh. Then, based on chosen strategy of classification, least significant edges are collapsed into a single vertex, and neighbor triangles (faces) are adjusted accordingly. With the method, it is possible to collapse a chosen number of edges, and hence reduce the amount of geometry from the original model.

3. METHODS, TOOLS, AND SETUP

Following methods, tools and setup were chosen for the basis of the measurements in this research.

3.1 Method Selection

Based on the previous research, a set of methods and algorithms were chosen for this paper. First, Cantlay [4] indicated how too large texture mipmaps can end up not being used in during rendering. Hence, it was chosen that in addition to the original 3D scene, a scaled down version of the scene was built with maximum of 128x128 pixel textures. The texture size was chosen to be an approximate for a good mipmap level for the 3D city model scene, with the given display sizes. As the density of geometry was known, and hence the density of texturing per single 3D model was known, the 128x128 texture size was considered to be a realistic approximation of a good maximum mipmap level for a mobile device.

Second, the research conducted by Ström et al. [8] [9] have clear indication about the possible benefits of ETC1 compressed texture format. Because the format was also supported natively by the mobile hardware used in the testing, it was a good choice for the measurements. DXT1 and DXT5 compressed formats [6] were also chosen, because those are widely used formats in the industry and likewise supported by the hardware we selected for the testing.

For geometry simplification, a polygonal simplification method was implemented as presented by Lindstrom et al. [5]. This algorithm was chosen, since it has been successfully implemented by a number of computer graphics applications. In this research, the implementation strategy for the simplifi-

Table 1: Attributes of the selected 3D scenes

Scene	Textures [B]	Vertices	Faces
Original RGB	35795328	501257	306540
Original ETC1	5965888	501257	306540
Original DDS	6099072	501257	306540
Scaled RGB	2273664	501257	306540
Scaled ETC1	378944	501257	306540
Scaled DDS	389248	501257	306540
Edges 75%	35795328	363327	205582
Edges 50%	35795328	286762	141632
Edges 25%	35795328	151158	90334

cation was to calculate all edge groups of the given models and then start optimizing them, starting from the shortest edges in order to produce geometry with certain pre-defined detail levels. For this research, edge reduction levels of 75%, 50% and 25% were used.

3.2 Measurement Setup and Selected Tools

The measurement setup was built around a mobile device based on NVidia Tegra2 chipset, developed by Trimslice. The device was chosen because of the hardware chipset. The same hardware is used in a number of mobile devices in the market today. The device was running Ubuntu Linux 11.04 with hardware accelerated OpenGL ES 2 drivers from NVidia. The display for the device was an external one, allowing us to execute the tests in different resolutions. The device was powered using an external 12V laboratory power supply, the energy consumption was measured with UNI-T UT61E digital multimeter and the measurement results were logged with a separate PC. All wireless radio interfaces were powered down from the device, since those were not needed in the testing. Hence, the measured values include energy consumption of the mobile device, excluding radio interfaces and the external display. The measurement setup is illustrated in Figure 1.

All the test cases were run using three pre-defined display resolutions. 320x240 resolution was chosen because it represents the lowest-end resolution of mobile phone displays today. 800x480 was chosen as another target resolution due to majority of the smartphones having such a display currently in the market. Finally, 1280x720 (so called HD-ready display) was chosen as well, since there are already an emerging number of high-end mobile devices utilizing such a high resolution display. These combined, it should cover a good selection of mobile devices in the market from the display size perspective.

In Table I, the original 3D virtual space and its variants are described. The first row, labeled Original RGB, of the table describes the attributes of the original 3D city scene. Textures column indicates a number of bytes consumed by the raw texturemaps in the specific scene variant. Vertices and Faces describe the amount of geometry in each scene variant respectively. The scene variants in Table I are defined as follows:

- Original RGB: The original 3D city scene to provide baseline of energy consumption for other scenarios.

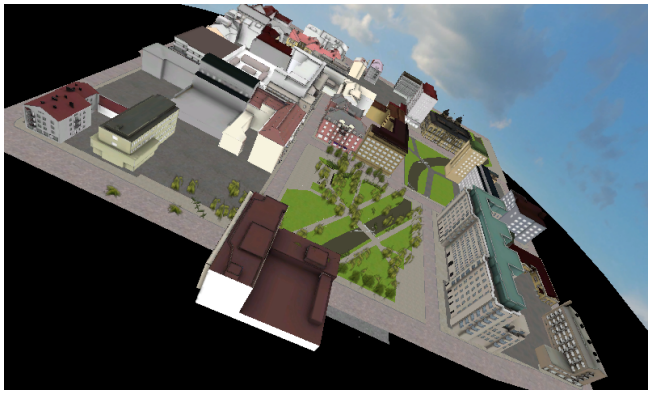

Figure 2: A sample screenshot of the 3D city model

- ETC v1: A variant of the original 3D city model with all textures converted to ETC1 format
- DDS/DXT: A variant of the original 3D city model with all textures convertex to DDS (Direct Draw Surface) DXT1/5 [6] format. DXT5 was used for texture maps with alpha channel (2 maps), DXT1 format for the rest of textures.
- Scaled RGB: The same as the original 3D city model, but all textures scaled to maximum of 128x128 resolution. Original textures smaller than 128x128 were left untouched.
- Scaled ETC v1: The scaling of 128x128 applied, and all texture data types changed to ETC1.
- Scaled DDS/DXT: The scaling of 128x128 applied, and all texture data types changed to DDS/DXTx
- Edges 75%: All textures used in RGB format from the original 3D city model. Retained 75% edges left from the original geometry.
- Edges 50%: All textures used in RGB format from the original 3D city model. Retained 50% edges left from the original geometry.
- Edges 25%: All textures used in RGB format from the original 3D city model. Retained 25% edges left from the original geometry.

Finally, after the individual measurements, a joint effect of the individual test cases was measured. For this purpose, the largest display size, both original and downscaled textures in DDS format and all the optimized geometry levels. This test case was seen a good comparison and a proposal for a real-life use case.

The rendering was performed using a custom built, and open sourced, 3D rendering software, written in C++ [11]. Its purpose was to render the given 3D scene as fast as possible, utilizing OpenGL ES 2 API and its most efficient methods. The most important ones were code paths for compressed textures and usage of Vertex Buffer Objects (VBO) for fast geometry instancing. The camera in the scene was positioned in a way that the whole scene was visible all the time

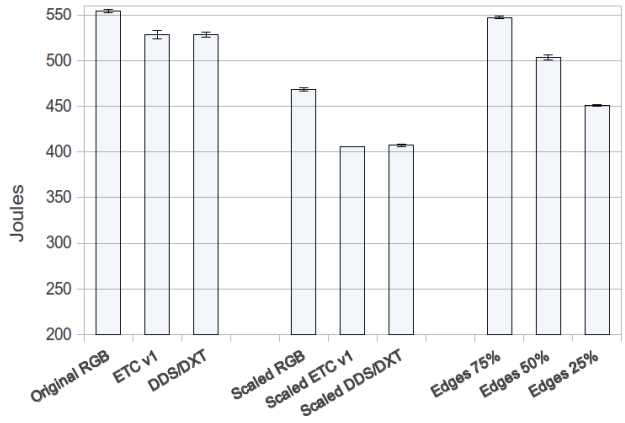

Figure 3: Total energy consumed, resolution 1280x720

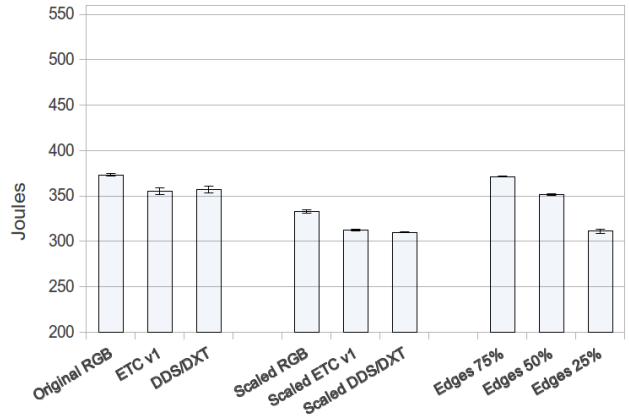

Figure 4: Total energy consumed, resolution 800x480

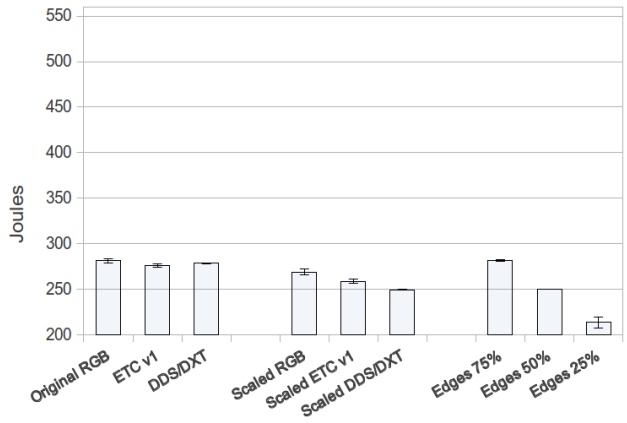

Figure 5: Total energy consumed, resolution 320x240

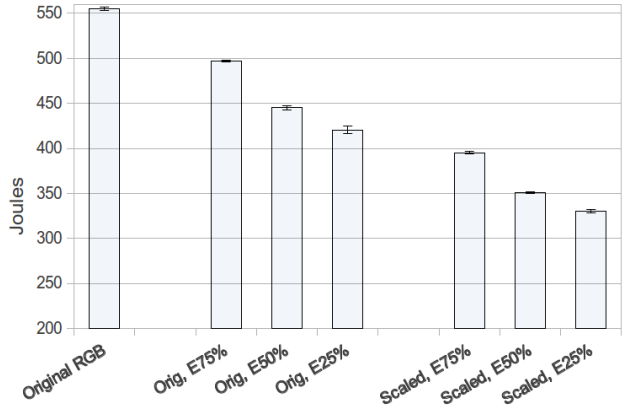

Figure 6: Total energy consumed in selected scenarios, resolution 1280x720

to ensure that all vertices will be rendered for the whole duration of the test. An example screenshot of the 3D city scene is shown in Figure 2.

A free tool Etcpack v1.06 from Ericsson was used to create all ETC1 based textures for the test scenes. Likewise, an open source library Crunch was used to create all DXTx variants of the textures. The edge collapse operation for the geometry was done using a custom made open sourced tool [11], which implemented an algorithm described by Lindstrom et al. [5]. All 3D scene variants were created off-line. All of the scene variants were locally available in the mobile device under test, i.e. nothing was downloaded from a remote service. Each scene was rendered for a duration of 2,000 frames and the tota l energy consumption was measured for the rendering phase. Each scene was rendered and measured ten times and the energy consumption measurements of consecutive runs were averaged to minimize the error.

The aim of the test was to reveal which test parameter affected the rendering process energy consumption and in which way. It was expected that more efficient methods and data formats will produce faster rendering result, hence lowering the burden of the mobile device, also, eventually enabling it to consume less energy during the process.

4. TEST RESULTS AND DISCUSSION

Figures 3, 4 and 5 present the overall energy consumption of each of the defined 3D scene variant for display sizes 320x240, 800x480 and 1280x720, respectively. In the figures, the original 3D city model is always the left most bar, and represents the baseline of energy consumption when the original scene is completely unmodified. Other scene variants in the graph can be directly compared to this column to see the effect of each method. Each bar in the figures has also the standard deviation visible, which was calculated over the ten consecutive measurements.

Overall, the most interesting phenomena which is visible in all of the figures 3 to 5, is that reducing the geometry by 25%, does not seem to affect the overall energy consumption at all. This indicates that the major time of the rendering process is spent in the fragment shaders performing per fragment calculations for the 3D scene. Hence, it can be stated that the 3D city model is either âĂİover texturedâĂİ, or the other way around: it would have more room for additional geometry to keep texturing and vertex transformation tasks

in balance. Going further, it seems that geometry reduction beyond 75% has a great impact on the overall energy consumption, and it is emphasized further with larger display resolutions. For example, using resolution 1280x720, the reduction of 75% of the total edges drops the overall energy consumption approximately 18%. By carefully selecting the edge collapse algorithm for the given input content, the geometry can be reduced while still keeping the original visual quality at an acceptable level. In this case, the selected edge priority based collapsing worked very well for the building models. Also, generally it can be stated that using large display resolutions (Figure 3), optimizing the texture maps give more benefit compared to small display resolutions (Figure 5) where geometry optimization gains most benefits.

The selected texture format for rendering also has an impact on the overall energy consumption. Having all texture maps in RGB format in GPU video memory is not a recommended setup. RGB textures will occupy 4-6 times more video memory and their rendering consumes 5% more energy when using original 3D city texture sizes and 15% more when using scaled down texture maps. Hence, it seems that energy efficiency of the compressed textures is emphasized when using scaled down (more optimal) texture sizes for the mobile display. When comparing the results between ETC1 format and DXTx formats, the consumption figures are slightly smaller in favor for ETC1. However, all of the figures fall inside the standard deviation of the measurements. It can be stated that both formats outperform RGB textures energy consumption wise, but a side-by-side comparison between ETC1 and DXTx shows them to perform roughly equally.

Having all the textures in a reasonable size for a mobile device display (in this case 128x128 texture maps), it is also benefiting the overall energy consumption. For the mobile GPU, using smaller textures reserves less video memory, and their size drives to a more efficient usage of the GPU texture cache. This is because sampling of smaller textures requires having less texels in the GPU cache, and hence, it makes texture cache hits more likely during the rasterization process. As a design guideline, the overall scene should be carefully analyzed how large texture maps are really needed. An over textured scene will perform worse, and when mipmapping is in use, it is possible that the large texture maps are not even used at all, as Cantlay [4] showed with his experiments.

Finally, Figure 6 shows the results from the measurement where both textures and geometry were optimized at the same time. The display resolution in use was 1280x720. The tests were run with both the original sized, as well as the optimized (128x128) texture sizes. In the figure, it can be seen that energy consumption drop the maximum of 40% when optimized textures, as well as 75% of the geometry are removed. The value is 11% lower than rendering the unoptimized scene in lower 800x480 resolution. Based on this, it can be defined that by optimizing the content for the hardware there is an opportunity to run the rendering in higher resolution within the same energy consumption range.

5. CONCLUSION

In this research, a number of methods were tested on how they affect the overall energy consumption of a mobile device during rendering of a 3D scene. The scene used in the testing was a 3D representation of a city. The 3D city consisted of nine street blocks, including roughly 500,000 vertices, 300,000 triangles and 35MB of uncompressed texture maps. Eight variants of the original 3D city model were created using compressed textures, texture size scaling for mobile device display and geometry reduction via polygonal simplification (edge collapse) algorithm. Three different display resolutions were tested, which were 320x240, 800x480 and 1280x720. All tests were run using NVidia Tegra2 based hardware, using a custom build open sourced C++ OpenGL ES 2 renderer and without network connectivity. The renderer is an open source component, hence it allows further experimentation and modifications around the research subject [11]. The energy consumption was recorded during the rendering process using a separate digital multimeter.

In the measurements, it was shown that using compressed textures instead of uncompressed RGB data is highly beneficial. This applies to both video memory usage, and energy required to render the given 3D scene. Compressed textures allow 4-6 times less video memory usage (depending on the chosen compression format), as well as faster and more energy efficient rendering. When using 1280x720 display resolution, using the RGB textures consumed 5% more energy (original size) and 15% more energy (downscaled size) compared to compressed textures. Benefits of the compressed textures are emphasized when using larger resolutions and when the texture sizes are optimized for the given display size. Primary reasoning is in the GPU texture caching, where more optimized textures are more likely to be in the texture cache, hence causing more cache hits and hence quicker and more efficient rendering.

Likewise, geometry compression gave similar energy consumption benefits. Using the largest display resolution, a geometry reduction of 50% of the original edges dropped energy consumption by 8%, and reduction of 75% of the original edges dropped the total consumption by 18%. Edge collapse reduces vertices and faces, and hence also loses detail from the original geometry. It is very important to choose a proper approach to make the reduction to keep visual artifacts minimized. For the 3D city scene, which consisted largely only of buildings, the chosen method (based on edge length priorities) worked well. Interestingly, during the testing it was noted that 25% reduction of the edges did not yet drop the overall energy consumption at all. This is likely due to an authoring error in the original 3D city scene. The original scene was overtextured, and reduction of the geometry did not gain anything energy consumption wise. In other words, the test scene performance was not originally bound by the geometry, but instead by the original texture maps.

When the texture optimization and geometry optimization was combined, and the measurements were done using the highest 1280x720 resolution with DDS textures, the overall energy consumption went down as much as 40% in total. It should also be noted that the measurement setup actually consumed less energy than rendering the unoptimized scene in lower 800x480 resolution. It indicates that there is clearly

room for gaining energy savings, by carefully optimizing the content for the mobile hardware.

The results in this paper gives an insight into understanding how the various algorithms (as presented in the literature section) applied to both images and geometry affect the energy consumption of a mobile device. By using this information, it is possible to start building fully automated content optimization services to the network, which utilize these methods based on the mobile device type, on demand-time. This way, all the content regardless of the original data format and author could be optimized automatically for the mobile devices in a way that the energy consumption is minimized. This research used one relatively complex 3D scene for the measurements. In the future these results should be generalized with various kinds of 3D models to allow creation of a generic energy consumption model based on 3D graphics complexity.

Also, content optimization will have an effect on the wireless transmissions. Decreasing the amount of content, and optionally compressing it, will lower the required bandwidth to download all the content to the device before rendering. One of the important next steps would be to find out how these methods will affect the overall energy consumption of the mobile device when the network interface is taken into account and the content is optimized and downloaded from a remote service.

6. ACKNOWLEDGEMENT

The authors would like to thank the staff at the Intel and Nokia Joint Innovation Center for their help and assistance. This work has been carried out in the Tekes Chiru Project.

7. REFERENCES

[1] B. Anand, J. Sebastian, Soh Yu Ming, A. L. Ananda, Mun Choon Chan, and R. K. Balan. Pgtp: Power aware game transport protocol for multi-player mobile games. In *Communications and Signal Processing (ICCSP), 2011 International Conference on*, page 399, feb. 2011.

[2] K. Lahiri B. Mochocki and S. Cadambi. Power analysis of mobile 3d graphics. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, DATE '06, pages 502–507, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[3] R. Brown, L. Cooper, and B. Pham. Visual attention-based polygon level of detail management. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 55–ff, New York, NY, USA, 2003. ACM.

[4] I. Cantlay. *Mipmap Level Measurement*, pages 437–457. GPU Gems 2. Addison Wesley, 2005.

[5] A. Dainotti, A. Pescape, and G. Ventre. A packet-level traffic model of starcraft. In *Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems*, HOT-P2P '05, pages 33–42, Washington, DC, USA, 2005. IEEE Computer Society.

[6] J. Munkberg, P. Clarberg, J. Hasselgren, and T. Akenine-Möller. High dynamic range texture compression for graphics hardware. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 698–706, New York, NY, USA, 2006. ACM.

[7] J. Pool, A. Lastra, and M. Singh. Precision selection for energy-efficient pixel shaders. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 159–168, New York, NY, USA, 2011. ACM.

[8] J. Ström and T. Akenine-Möller. ipackman: high-quality, low-complexity texture compression for mobile phones. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '05, pages 63–70, New York, NY, USA, 2005. ACM.

[9] J. Ström and P. Wennersten. Lossless compression of already compressed textures. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 177–182, New York, NY, USA, 2011. ACM.

[10] J. Vatjus-Anttila, S. Hickey, and E. Kuusela. Methods and network architecture for modifying extensible virtual environment to support mobility. In *MindTrek'11*, September 28 2011.

[11] J. M. Vatjus-Anttila. Opengles2 render engine for energy measurements. http://chiru.fi/?udpview=12357&sid=5&src=db23044, July, 2013.