Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 887–898

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

An Investigation into Power Aware Aspects of Rendering 3D Models on Multi-Core Processors

Guruprasad Konnurmath[a*], Satyadhyan Chickerur[b]

[a]*School of Computer Science and Engineering. KLE Technological University, Hubballi, 580031, India*
[b]*Centre for High Performance Computing, K L E Technological University, Hubballi, 580031, India*

Abstract

Graphics Processing Units (GPUs) are designed to process numerically intensive applications and specially customized with additional computational power to achieve efficient rendering of 3D applications. Shaders are basically the computer programs that run on graphics rendering pipeline and inform about how to process and render each pixel, to perform efficient rendering of 3D applications. Shaders included in our Geometric complexity, texture resolution, per-pixel lighting. These shaders expose GPUs to suffer more power utilization during rendering process. In this paper, initially GPU power usage is investigated at run-time for rendering configuration to define shader parameters on a 3D scene and then power prediction model is proposed at different shader calls on GPU to generate power-aware dynamically reconfigurable rendering model. In the implementation process we integrate the TensorFlow management library routines and dynamic voltage and frequency scheduling (DVFS) mechanism to improve efficiency in rendering process and reach optimal power savings. To analyze the power efficiency of GPUs, shader parameters such as geometric complexity, texture resolution, per-pixel lighting, filtering that expose more towards power consumption are experimented with common GPU workloads during rendering process, then frequency and voltage is altered based on the framerate monitoring. Also in addition to this the frame rate is monitored and maintained in such a way that GPU frequency is adjusted in parallel if the framerate is within the acceptable range so that unnecessary power usage is reduced. During the experimental tests a lower limit of 30 frames per second and upper limit of 60 frames per second is configured. The GPU frequency is reduced if the existing framerate is greater than upper configured limit and frequency range is increased if framerate is below the configured lower limit. Compared with the previous state of art, results show power savings around 40% and improved speedup in computation time with maximumquality of a 3D scene.

*Keywords:*GPU; 3D graphics rendering; Shader parameters; TensorFlow; DVFS; Workloads;

* Corresponding author. Tel.: +91-9538012981
E-mail address: Guruprasad.konnurmath@kletech.ac.in

1877-0509
10.1016/j.procs.2023.01.069

1. Introduction

3D model rendering is a substantial ultimatum in computer graphics processing which is majorly divided into categories. Dynamic rendering of 3D model in real time refers to thorough analytical synthesis on rendered scenes over a raw graphics rendering hardware providing flexibility in processing and aid interactions for clear visual appearance and 3D computer games. The physical hardware and software integrated based domain rendering significantly project on the algorithms which precisely coordinate the properties of light called physics of light necessary for 3D movie productions, architectural visualizations, etc. As compared with rendering in real time, these algorithms take larger amount of time to produce satisfactory results [1]. From background perspective, decreasing power consumption for applications related to rendering 3D models is major need and a significant challenge in computer graphics [2][5]. Researchers in this domain believe in stressing upon decreasing the computation overhead which has lesser impact on experiencing visual effects leading to result oriented solution. To render 3D computer graphics applications in real time, the requirement to render finest scene quality and natural dynamic response on GPUs is the most challenging issue for the system designers. Improvement in rendering performance leads to enormous power consumption. Real-time rendering applications consume huge amount of power correlatedly reduce battery life affecting the related parts of the system. Several metrics such as rendering configuration, power efficiency, application characterization and throughput contribute towards the evaluation and analysis of power consumption on GPUs [2]. Considering all these factors the main motive of this paper is to render high quality photorealistic 3D model in the shortest possible time within given power budget. During rendering process, the GPU accelerator immediately recognize change in the color for each pixel to be displayed on the resulting screen, after each subsequent 3D scene is loaded into various stages of rendering oriented pipeline. In the following process of the work, the geometrical complexity, texture resolution, viewport, illumination and many other parameters are essential to analyze the status of 3D objects for each scene [3][7][13]. In addition to this complexity of extracting such parameters, modern processors remodulate themselves during dynamic runtime in order to improvise throughput performance under tightly drawn power related constraints. In parallel, these constrains shall immensely affect the GPU core frequency and voltage [4][8] with change in available bandwidth [10].

In alignment with these considerable solutions achieved through software perspectives, in the proposed work a demonstration of power aware optimal settings for processing elements that draws benefit of graphics explored rendering parameters associated in relevance to most of the 3D graphics applications to attain sustainable power savings in GPUs. With respect to this real-time 3D model rendering, our research work mainly focus on rendering the best image quality under optimized power consumption. Along with the design of algorithms we also focus on the software tools that integrate and contribute towards the rendering and power aware workloads. In this paper, we primarily focus on power saving methodology for 3D model applications that dynamically tune with the parameters associated to optimize power dissipation without disturbing other computing resources inline. A configured setting of power saving methodology is framed through the results obtained from various experiments.

At first, the features of 3D model is analyzed that are being rendered through various stages of rendering pipeline and measure the power related factors of different parameters that contribute towards 3D model processing in graphics rendering pipeline. The geometric complexity for each 3D scene, texture resolution, level of detail and per-pixel lighting for each 3D scene are the parameters analyzed with respect to power aware aspects and their inherent features. Frame rate that applies to all kind of scene oriented applications and GPU frequency whose main function is to render graphics are the two key features that perform prominent decisive role in the implementation phase. Framerate of a scene is set between 30 fps (frame rate per second) and 60 fps and GPU frequency is monitored and adjusted, so that inessential power usage of computational resources can be reduced. In order to investigate and analyze the power features of the parameters that contribute towards more power consumption during rendering, tuned settings from C1 to C8, as mentioned in Table 1., is configured individually. The TensorFlow framework mainly informs GPU associated information namely type and count of GPUs, application type processing on GPU and data occupied by the memory. Pipeline mechanism feature is possessed by TensorFlow library that generates massive data parallelism meanwhile leading to faster processing speed. The TensorFlow libraries commits in identifying the GPUs and faster processing [4]. Ensembling altogether, the parametric analysis of the proposed work is observed and performed in integration with TensorFlow library, GPU dynamic frequency and voltage scheduling technique [4] configure by varying the frame rates for any of the dynamic real-time 3D model application.

Proposed major objectives addressed to achieve power aware optimal rendering framework include:

- Objective 1: Investigate GPU power consumption at run-time for rendering configuration to define shader parameters on a 3D scene.
- Objective 2: Propose a power prediction model based on scene complexity and bandwidth occupancy at different shader calls on GPU.
- Objective 3: Implement an algorithm on a GPU accelerator to generate optimal rendering configuration.

2. Related Work

Real time 3D graphics rendering plays a major role in several multimedia applications used in today's modern world. The requirement for improved quality of 3D scene being rendered and immediate dynamic response motivates the system developers to boost the performance of rendering 3D models. Work in the paper [1] [3] describe the role of integrated and discrete GPUs with workload distribution. Improper load distribution on GPU cores lead to huge power consumption of computing resources related to rendering [21]. In our proposed work this issue is addressed by using TensorFlow libraries that identify GPU cores individually based on the allocated workload and able to process with speedy execution. Researchers argue that the lower power consumption design of processors as well as accelerators is the major implementation challenge and also the power optimization on processing components shall be the greater limitation aspect in the future upcoming technologies [6]. In the analysis of design process, optimized power dissipation, effective throughput and chip area are the major design metrics [4][11] that need to be evaluated.

The GPUs optimized power consumption is an increasing concern in various domains both in the design of algorithms as well as hardware integrated architectures [15][16]. In the process of rendering higher quality scene require more computing resources and thus take more time to calculate. The primarily focus is to identify the shader parameters, supposed to have more memory access with GPU that led to much bandwidth utilization. Due to this more power consumption in GPU is observed. The power model in paper [2][7] is designed on the basis of classic rendering stage-based strategy with basic functionalities like batch processing, texture, vertex or fragment shaders. The algorithm framed in the work is built on two principal components: first the peculiar prediction model of power consumption and the second quality error estimation strategy during run time. These two mechanisms acquiesce in finding the optimal efficient configuration for rendering, maintaining transparency to user and observer. Results and methodology describe the importance of shader parameters in the rendering process as GPU power savings. It is also observed that to calculate scene complexity, precomputation takes more time and requires many memory accesses hampering performance and cannot handle dynamic scenes [9][23]. Significantly in this work, investigation on shader parameters is performed mainly geometric complexity, texture resolution, per-pixel lighting, filtering are identified that expose to more power consumption and aim at resolving the issue. If there exist highest complexity in the scene to fit into the GPU's memory, then the GPU rendering engine shall either access RAM or swaps on the disk, this slows down the rendering process [10]. To overcome this problem, the TensorFlow library used in the proposed work outperform massive parallelism, cleanup the disk space and improves the execution speed for rendering process. To analyze the importance of 3D scene the research work in the paper [5] create a ongoing power budget rendering methodology in real time that leads to balancing of power consumption and rendered quality of image and gives the generalized implementation mechanism across various platforms and types of 3D scenes. The results are explored using six scenes with power saving configuration integrated to run on GPUs [1][14]. Fig. 1. show the programmable graphics pipeline, the vertex and pixel shaders perform computationally intensive jobs, mainly textures with larger dimensions use more memory and require more data, this load balancing directly affects speed of GPU. For vertex shader, each vertex is fed as input data in sequential format. Manipulation algorithm of vertices is analysed and executed in Vertex shader. The output of this vertex shader is integrated and loaded into succeeding stages of rendering pipeline meant for graphics processing. Rasterizer stage is next following stage in the pipeline. Fragments are output for this stage and fed into pixel shader. And finally, this pixel shader performs required pixel manipulation and then the results are stored in the frame buffer [6]. The experiments conducted in [17], show that many factors such as vertex sharing, vertices count, texture mapping, LOD affect processing performance and the results prove to be the basis for analysis of performance and optimizing power consumption on GPUs in 3D graphics rendering.

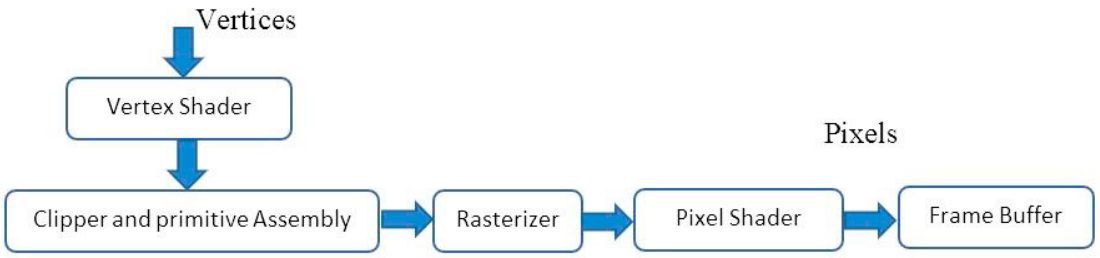

Fig. 1. Programmable graphics pipeline

When BW occupancy is very high, the GPU may execute its workload poorly because threads are competing for space in the GPU's memory caches (*cache thrashing*) [7][8]. In few scenarios, pixels appear shaded without being showing appearance on the screen, and lead to enormous bandwidth utilization and power dissipation [11]. Overdraw is a feature of pixel rendering that creates increased fillrate (pixels count the rendered by associated hardware per sec) and rise in bandwidth (data transferred in and out to GPU per second). These fillrate and bandwidth can scarce resources and consume power in idling state [12][13]. In our proposed methodology optimal power savings is achieved after analysing the power consumption for shader parameters by altering the limit range of GPU core frequency through frame monitor.

3. Proposed Methodology

The design architecture of methodology proposed is presented in the Fig. 2. The architecture is primarily comprised of three components: (a) the TensorFlow library, which mainly performs device placement for 3D rendering in the initial stage and achieves data parallelism, (b) the Graphics Rendering Parameters Control (GRPC), that portrays as an intermediary and performs parameters manipulation that shows an impact on power consumed during run time and, (c) the Dynamic Voltage and Frequency Scheduling technique, which performs adjusting the frequencies of the GPUdepending upon the framerate control.

First, the power consumption of GPU and related computational resources for 3D scene rendering is investigated. Then the TensorFlow library is used for dedicated device placement and to achieve data parallelism and setup frame rates that the 3D applications run on a GPU within a configurable range defined by the user. During the process of rendering a 3D scene, the geometric complexity, texture resolution, level of detail and filtering parameters shall be modified under graphics rendering parameter control to minimize the usage of computing resources needed for accomplishing processing in real-time. Also, amidst rendering process it is not necessary to maintain higher GPU frequency if the current frame rate isn't within an admissible range. By analyzing the prevailing status of GPU load through TensorFlow, which notifies about GPU's idle timeline and portion of workload carried, power consumption can be minimized by varying voltage and frequency on those specified GPUs. At the time of experimentation DVFS technique is applied by changing the frequency levels that is monitored under frame monitor. We consider the rendering process for 3D scene to generate a best quality image, let the rendering settings that includes minimum rate 30 frames per second (fps) so that it is acceptable to user experience. Based on this best rendering setting we devised the model to have frame rate greater than the real frame rate. This configuration is accomplished through manipulating the parameters of the input scene and the dynamic voltage and frequency of the GPU.

In our experimental tests, the lower limit of 30 frames per second and higher or upper limit of 60 frames per second is observed. This range limit gives an acceptable dynamic real-time efficiency for most 3D rendered applications, but vary depending on the system's hardware configuration as shown in Fig. 3., that gives an overview of dynamic voltage and frequency scheduling algorithm used in the proposed methodology for power-aware rendering of 3D scene over a GPU by setting frame monitor. In the comparison of the samples suppose the

current ongoing framerate is greater than the configured upper limit, the frequency of the GPU core is reduced to the next level. And if the framerate is below the lower limit set, GPU core frequency is increased to reach a greater framerate. It needs to be observed that graphics processing unit frequencies is performs switching every second per once to avoid the GPU governor of raising the extra burden of workload.

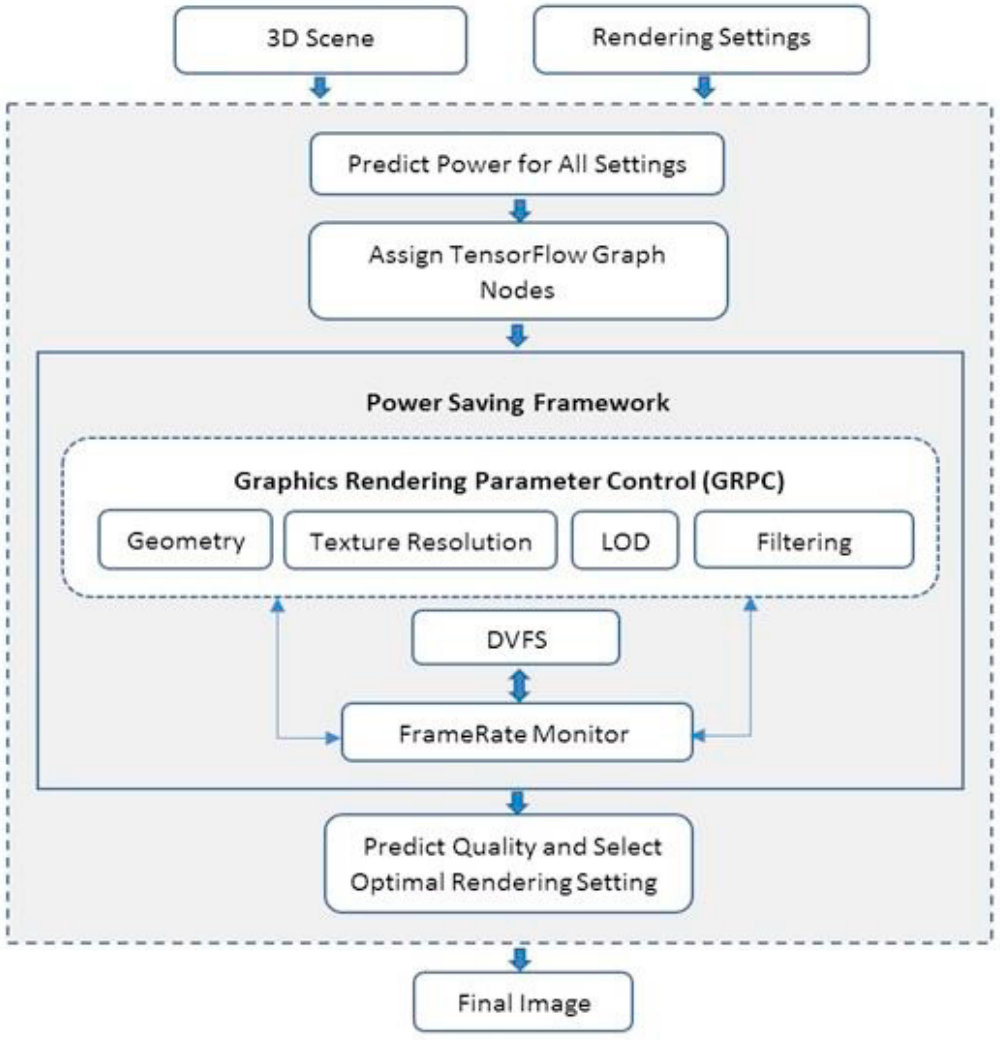

Fig. 2. Architecture of the proposed methodology

In order to reduce imposition of additional overhead, switching of GPU to varying frequencies is done at most every second. During real time rendering process, the frame rate of the 3D scene is sampled and stored under user defined window. By default, buffer of 5 samples is used. The average of these five samples is taken and compared with the current frame rate. Based upon workload over GPU identified through TensorFlow, the DVFS technique is initiated by varying the voltage and frequency levels of GPU cores [4]. At the last stage a power optimal rendering configuration is set to render the final image with reduced power savings upon application of the proposed methodology. The main motive of this research work is, during the process of rendering a 3D scene, the geometric complexity, texture resolution, level of detail and filtering shall be modified under graphics rendering parameter control to minimize the usage of computational resources required for accomplishing real-time rendering. Also, amidst rendering process it is not necessary to maintain higher GPU frequency if the current framerate is under the normalized sufficient range. The algorithm proposed comprises of speedy, computationally inexpensive and

controlled frame rendering design with optimal power budget achieved by reducing complexities in the scene mainly geometry and textures.

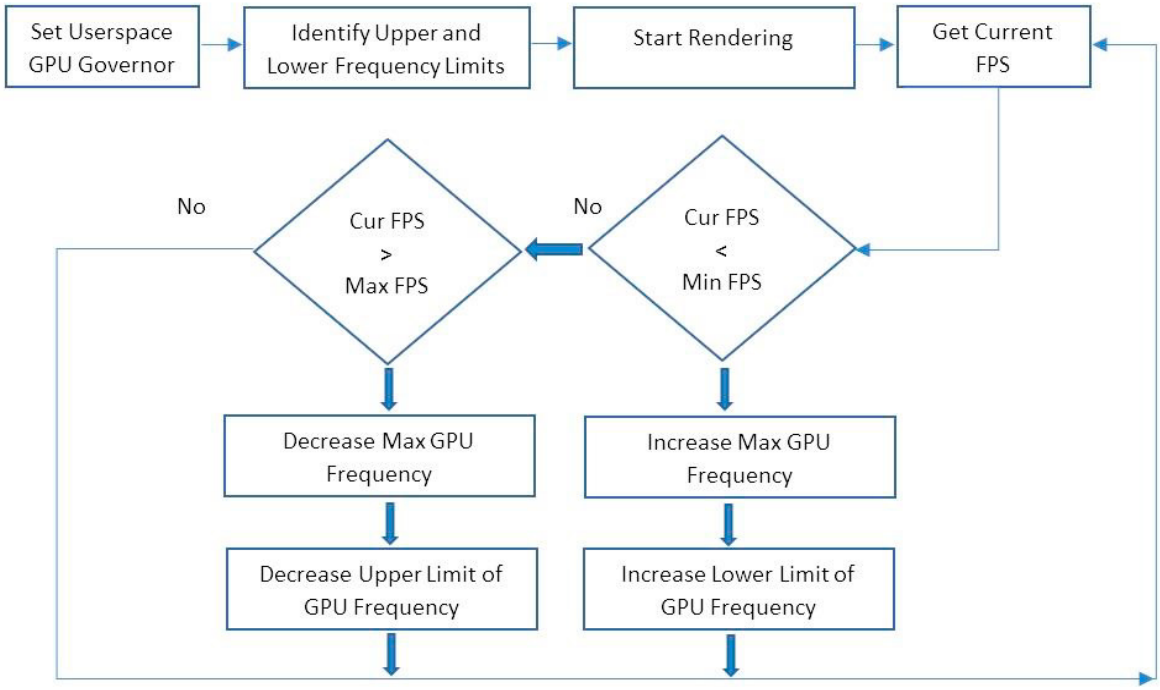

Fig. 3. An overview of Dynamic Voltage and Frequency Scheduling Algorithm

3.1 TensorFlow library

In this research work the role of TensorFlow library to dynamically managing the resources, i.e dynamic resource management system (DRMS) to generate the graph nodes. Here the mathematical operations are performed by the nodes in the graph. Edges of the graph that communicate among nodes are named as tensors and these are represented by the multidimensional arrays. This flexible and portable architecture of TensorFlow makes it to distribute computations on single or many GPUs in accordance to the requirement. Since TensorFlow acts as graph which is considered to be the powerful data structure through its parallel execution mechanism it allows massive increase in the execution speed. As an additional advantage it identifies duplicate operations or the sub-optimal graphs and replaces those operations with the substantial alternatives in order to optimize the time constraints. These features of TensorFlow leads to optimized computation resulting into boosted efficiency in the execution time and optimized power savings.

3.2 Graphics Rendering Parameters Control (GRPC)

This GRPC projects as an intermediary amidst power optimized framerate-configured framework and rendering parameters which gives the best power budget savings. The GRPC can be improvised by including some more parameters. Since it not confined to only the current work, it has the flexibility to integrate with several other factors. Practically the Graphics rendering parameters control is software implemented and performs power analysis of shader parameters geometric complexity, level of detail management, texture size and filtering selection independently. The proposed methodology can be tested with few more parameters to predict the accuracy of the framework.

3.3 Dynamic Voltage and Frequency Scheduling (DVFS)

The major need to integrate the DVFS mechanism in this paper is to perform analysis of the power consumed by the GPU through altering of frequency and voltage levels during by understanding the workload status of GPU core. When the frequency level gets decreased, alternately the consumption in GPU power shall be decreased. When the GPU core clock is at a highest frequency level GPU's power utilization increase. The general way to reach power savings on GPU is to clearly recognize application processing requirements and parallel adjust GPU voltage and frequency level to supply sufficient processing power while rendering 3D scenes without working at its maximum capacity. The proposed GPU dynamic voltage and frequency scheduling frequency scaling integrated with GRPC, substantially minimizes the overall power consumption on GPUs undergoing 3D model rendering. The following equation describes the GPU's power consumption:

$$\text{Power} = \text{Capacitance} \times \text{Voltage}^2 \times \text{Frequency}$$

- P: the actual power consumed by the GPU
- C: capacitance
- V: voltage level supplied to GPU
- F: clock frequency of the GPU

4. Implementation

The GPU core and memory frequencies, bandwidth, clock speeds, shaders count and textures are the major parameters that contribute in rendering 3D applications along with performance and power consumption of GPUs. The proposed work can be implemented on different platforms of NVIDIA GeForce GTX, RTX graphics support with basic processor starting from Intel Core i5. Experimentation of the proposed work is implemented on desktop PC, i7-7700 Intel core and NVIDIA GeForce RTX 2080Ti with NVIDIA Quadro P4000 graphics card. The configuration includes around 4000 CUDA cores, GPU core clock speed ranges from 1200MHz to 1350 MHz, memory bandwidth from 192 GB/s to 616 GB/s, texture fill rate upto 420.2. The selected configured frequency triggers every process to pick a new optimal rendering configuration of selected number of frames after each of the previous configurations is set. This frequency configuration setting results in quick identification of any change in the rendered scene with reduction in the impact of related computations.

4.1. Power Measurement

NVML (NVIDIA Management Library) API is used in order to measure the power utilized by the graphics card used in laptop or desktop personal computers. This NVML is configured to observe and manage the status of NVIDIA GPU devices. This gives direct assessment of the power usage on a particular GPU with its related circuitry support. The average power measured in the experimental setup will be over 30 frames. The average Thermal Design Power (TDP) for the configuration is around 270w with support of 5,304 gflops floating-point performance. Using DVFS mechanism the GPU frequency is altered dynamically every 1 second and framerate is managed from 30 to 60 fps. From this NVML configured setting, optimal power consumption can be observed.

5. Results and Discussion

To form the rendering workload, a complex characteristic 3D synthetic scene is used which represents a particular scenario of 3D game shown in the Fig. 4. A predetermined path of camera parameters are analysed in order to flow through 3D scene in a redundant procedure among any kind of experimental tests. Actual geometry in 3D scene includes above 300k polygons. In addition, 70 model types were included, which represents statues and with each of them have 3 levels of geometric information as shown in table 1. All the models are rotated programmatically to avoid the rendering engine with respect to the calculations on caching. The included models are systematically configured in terms of position and size for managing level of detail so that it is activated effectively along the run time.

Fig. 4. Power profiled 3D scene

To configure and validate the proposed methodology, a systematic sequence of experimental tests were conducted consisting a count around 8 different tuned settings as shown in Table 1. These tuned settings are produced by analysing 4 distinctive parameters mainly the geometric complexity, texture resolution and LOD, filtering features and dynamic voltage and frequency scheduling (DVFS) which affect the power consumption on GPUs. 2048^2, 1024^2 and 512^2 pixels are the three texture resolutions been considered as shown in the table and each texture for every configuration setting and the geometric complexity is changed using Level of detail manager. All these configurations are programmatically loaded during the start of the rendering process. Standard filtering is used in this experimental process that improves the image quality with added computations over fragment shader with respect to GPU.

Table 1. Experimental configuration power consumption of each parameter independently

Tuned Setting	Texture Resolution	Filtering	LOD management	DVFS
C1	2048^2	Yes	No	No
C2	1024^2	Yes	No	No
C3	512^2	Yes	No	No
C4	2048^2	No	No	No
C5	2048^2	Yes	Yes	No
C6	2048^2	Yes	No	Yes
C7	2048^2	No	Yes	Yes
C8	512^2	No	Yes	Yes

In order to analyse and evaluate the power consumed by each measurement, a standard tuned setting called as reference condition (C1) is used. This C1 is assumed to be the maximum expected power consumed among all the tuned setting conditions mentioned in Table 1. Under this scenario the maximum best possible quality within all available factors is considered. The tested framerate for every tuned setting was set from 30 fps utpto 60 frames fps. In every case the framerate is balanced closed to user mentioned limit of frame rates. Lowest values for C7 and C8 is observed, but the tuned settings have the highest power savings. In contrast, C4 in contrast achieves a huge framerate, under the settings filtering is kept off and there is no overhead of LOD nor the DVFS. From the Table 1. it is clearly understood through observation that C1 which acts as referenced tuned setting, found to be more expensive, meanwhile for the tuned setting C7 and C8 are DVFS enabled settings provide best power savings.

5.1. Textures

Tuned settings C1 through C3, examine consumed power for extraneous workload introduced by changing the test scene's resolution of the texture. Upon varying texture resolution in the tested scene, the power consumption in the workload for the tuned settings C1 through C3 is investigated. No desirable benefit is found for tuned settings C2 and C3. With reduction of texture resolution to 1024^2 pixels leads to improvement of marginal but negligible power consumption of GPU by 2%.

5.2. Filtering

For analyzing the power consumption due to filtering on GPU the tuned setting C4 is defined. With this setting filtering is disabled and slightly compromised with quality of the image. All other parameters remain same as observed for the tuned setting C1 the reference setting. By disabling the filtering process leads to energy saving around 35% for the GPU whenever compared with the reference condition setting.

5.3. Level-of-Detail (LOD)

The consequence for geometric complexity is observed under the tuned setting C5, Level of detail management is enabled. The energy savings under LOD management in comparison with reference tuned setting C1 is about 33% forGPU with total gain of 15% is observed.

5.4. Dynamic Voltage and Frequency Scheduling

The DVFS implementation procedure that has been executed in the methodology is analyzed through in C6. By enabling the DVFS mechanism the GPU frequency is managed dynamically during runtime for each 1 second and configured to manage the framerate within 30 to 60 fps. It can be monitored clearly that an improvement around 74% in GPU power consumption is accomplished. Under tuned setting C6 the consequence of power consumption for DVFS is observed. DVFS core frequency is managed dynamically at runtime for every 1 second and frame rate of range between 30 to 60 fps is maintained. More than 50% of the improvement is GPU power consumption is observed underthis DVFS state.

5.5. Overall Observations

Fig 5. show the framerates achieved at all the experimental tuned settings. From the Table 1., it is clear that settings C7 and C8 exhibit the observations with all optimization configurations enabled and varying texture resolution of 512^2 and 2048^2 pixels. Total comprehensive saving for the condition C7 is more with power saving of 50% for GPU. Majority of the power savings for GPU is mainly because of DVFS technique and minimal percentage ratio is due to Level of detail management and filtering. It can be shown that GPU power savings is achieved by enabling DVFS and disabling LOD and filtering. In case of texture resolution even though it was reduced up to 512^2 under tuned setting C8 much power savings was not observed.

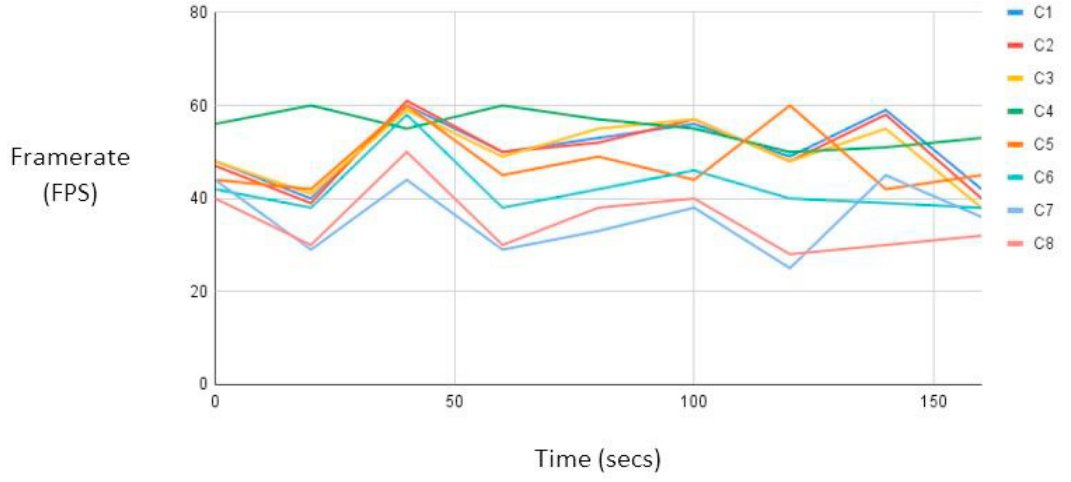

Fig. 5. Framerates achieved at all the experimental tuned settings

With these observations of power savings on GPU for the proposed methodology, we investigated the potential impact of per-pixel lighting and shading calculations mainly referred as vertex-based calculations while rendering 3D scene over a GPU. The per-pixel lighting scenario is found to be more intensive computationally and demands more power during the process. Hence both fillrate and bandwidth can scarce resources and consume power. To address these issues, the proposed methodology is implemented to analyze and understand the implications of per-pixel lighting in terms of power dissipation on GPUs. The minimized compute time limit and duplicated pixels that are eliminated reduce the switching activity to incorporate given triangles.

Table 2., show the speed of the rendering performance is improved from existing 1.235× upto 1.548×. Hence overall power is saved, and average power consumed is reduced as compared with the existing works in [5] and [7] that requires much precomputation and implementation mechanisms for rendering the scenes. There is an improvisation in power efficiency from1.812× to 4.826× with respect to given size of the triangle. Less precision is needed for smaller sized triangles and therefore more power saving is observed from the proposed methodology. It is also observed that even though the size of the triangle is larger as up to 255 in both cases of width and height, still improvement of 1.876× in power efficiency can be seen. When the GPU is operating at 250MHz, the minimal average GPU power being consumed is 142.0 mW. To render a pixel, it will take an average of 600.8 pJ. The total average efficiency of the rendering pipeline is around 2.143MPixels/mJ. Dynamic voltage and frequency scheduling scaling technique can be introduced at the time when the performance configuration is not at a greater value.

Table 2. Power comparison with and without proposed methodology

Triangle Size	Without proposed methodology				Without proposed methodology				Efficiency improvement
	Bus Bandwidth Occupancy	Avg Power	Pixel Energy	Efficiency	Bus Bandwidth Utilization	Average Power	Pixel Energy	Efficiency	
(Width in pixels)	(%)	(pJ)	(MPixels/ mJ)	(mW)	(%)	(pJ)	(MPixels/ mJ)	(mW)	
255(Large)	93	248.343	518.989	1.045	94	137.534	282.454	3.888	1.812x
171	90	256.867	570.542	1.812	92	137.623	290.214	3.675	1.856x
128	87	268.443	614.324	1.753	89	137.745	300.099	3.402	1.948x
102	83	274.676	617.136	1.612	85	138.098	310.785	3.217	1.892
85	80	302.554	700.087	1.587	82	138.435	320.498	3.056	2.300
51(Middle)	63	317.825	864.231	1.291	66	139.786	390.213	2.975	2.203
25	48	340.774	1303.645	0.843	52	140.897	480.899	2.587	2.8
10	30	377.432	2260.065	0.554	35	142.567	760.012	2.067	2.998
5	20	473.988	3622.978	0.398	24	145.544	1080.034	1.560	3.401
2(small)	14	324.433	6417.825	0.256	15	150.091	1350.067	1.989	4.826
Average	-	318.533	1748.982	1.115	-	140.832	556.527	2.841	

6. Conclusion and Future Work

In this paper, experimental tests were conducted with respect to power optimization and presented the results from various power measurements under proposed methodology that is comprised of the TensorFlow library, Graphics Rendering Parameter Control and Dynamic Voltage and Frequency Scheduling technique. The major 3D scene parameters, i.e., geometry resolution, filtering, LOD, resolution and size of texture provide a significant contribution to exploring a 3D scene and are considered for experimental tests. For each case, with fine-tuned configured setting of the proposed methodology, a predefined 30 frames per second can be maintained that gives a deep insight into the frames and free exploration of the rendered 3D scene. Also the scene resolution is lowered rarely as it may lead quality errors. The outcome of the results shows power savings on GPU using each of the parameters independently and the combination of all parameters demonstrates an overall 40% of the power savings.

The framework has been implemented successfully comprising of many key advantages. It does not require any huge precomputation; dynamic scenes are handled smoothly and do not lead to any degradation in frame rates that may affect the image quality. Also, in the case of per-pixel lighting analysis, the proposed methodology with the integrated framework saves considerable power consumption without affecting the scene quality and provides improvised performance. The predefined power setting is configured individually during each demo that can be used to monitor the proposed optimal rendering configuration process. Through TensorFlow library management better parallelism have been achieved that led to faster rendering process. Maximum power savings is achieved with the implementation of DVFS technology in the proposed methodology. In general, from the implemented integrated framework resulted in optimal power consumption with improved rendering power efficiency. The predefined camera paths, scene exploration, and measurements with various qualities under different configuration settings exhibit the potential work implemented and can be extended for the long run to include some more parameters of complex 3D models. In extension with the existing work, a few more parameters such as bandwidth occupancy and fill rate among pixels that are computationally intensive and require much power utilization during the rendering process can be considered for investigation in integration with the proposed work. Also for more elaborative and comparative work, resource balancing techniques using different 3D models to manage CPU and GPU workloads can be considered as a part of future work.

References

[1] Xiao Lei and Vetria L. Byrd. (2020) “Real-Time Rendering With Heterogeneous Gpus.” *International Conferences Computer Graphics, Visualization, Computer Vision and Image Processing*. ISBN: 978-989-8704-21-4 © 2020.

[2] Yunjin Zhang, Marta Ortin, Victor Arellano, Rui Wang, Diego Gutierrez, Hujun Bao. (2018) “On-the-Fly Power-Aware Rendering.” *Eurographics Symposium on Rendering.* Volume 37, Number 4.

[3] Songlan Wang, Ji Zhang. (2021) “Research and Implementation of Real-time Render Optimization Algorithm Based on GPU.” *Journal of Physics: Conference Series.* 2136 (2021) 012059. DOI:10.1088/1742-6596/2136/1/012059.

[4] Guruprasad Konnurmath and Satyadhyan Chickerur. (2021) “Power-Aware Characteristics of Matrix Operations on Multicores.” *Applied Artificial Intelligence (Taylor & Francis Group).* Volume 35, Number 15, 2102–2123. https://doi.org/10.1080/08839514.2021.1999013.

[5] Yunjin Zhang, Rui Wang, Yuchi Huo, Wei Hua, and Hujun Bao. (2021) “PowerNet: Learning-based Real-time Power-budget Rendering.” *IEEE Transactions on Visualization and Computer Graphics*. DOI 10.1109/TVCG.2021.3064367.

[6] Ádám István. (2019) “Szűcs Improving Graphics Programming Withshader Tests.” *An International Journal for Engineering and Information Sciences.* DOI: 10.1556/606.2019.14.1.4. Vol. 14, No. 1, pp. 35–46.

[7] Rui Wang, Bowen Yu, Julio Marco, Tianlei Hu, Diego Gutierrez, HujunBao. (2016) “Real-time Rendering on a Power Budget.” *ACM.SIGGRAPH 16 Technical Paper*. Anaheim, CA. ISBN: 978-1-4503-4279-7/16/07. DOI: http://dx.doi.org/10.1145/2897824.2925889.

[8] Songtao He1, Yunxin Liu1, Hucheng Zhou. (2015) “Optimizing Smartphone Power Consumption through Dynamic Resolution Scaling.” *ACM*. ISBN 978-1-4503-3619-2/15/09. DOI: http://dx.doi.org/10.1145/2789168.2790117.

[9] Chao Peng and Yong Cao. (2012) “A GPU-based Approach for Massive Model Rendering with Frame-to-Frame Coherence.” *Computer Graphics Forum.* DOI: 10.1111/j.1467-8659.2012. 03018.x. Volume 31, Number 2.

[10] Chenhao Xie, Shuaiwen Leon Song , Jing Wang, Weigong Zhang , Xin Fu. (2017) “Processing-in-Memory Enabled Graphics Processors for 3D Rendering.” *IEEE International Symposium on High Performance Computer Architecture*. 2378-203X/17 $31.00 © *IEEE.* DOI:10.1109/HPCA.2017.37.

[11] Sparsh Mittal And Jeffrey S. Vetter. (2012) “A Survey of Methods for Analyzing and Improving GPU Energy Efficiency.*” ACM Computing Surveys*. Volume 47, Number 2, Article 19. DOI: http://dx.doi.org/10.1145/2636342.

[12] Michael Kenzel, Bernhard Kerbl, Dieter Schmalstieg, Markus Steinberger. (2018) “A High-Performance Software Graphics Pipeline Architecture for the GPU.” *ACM Transactions on Graphics*. Volume 37, Number 4, Article 140. https://doi.org/10.1145/3197517.3201374.

[13] C. Kanagasabapathi, Siddamma, Siva S Yellampalli. (2019) "Design of Programmable Pixel Shader Computing Unit." *International Journal of Innovative Technology and Exploring Engineering (IJITEE).* ISSN: 2278-3075, Volume, Issue-9S3.
[14] Yong He, Tim Foley, Teguh Hofstee, Haomin Long And Kayvon Fatahalian. (2017) "Shader Components: Modular and High Performance Shader Development." *ACM Transactions on Graphics*. Volume 36, Number 4, Article 100.
[15] Markus Schütz, Bernhard Kerbl, Michael Wimmer and . TU Wien. (2022) "Software Rasterization of 2 Billion Points in Real Time." *arXiv*:2204.01287v1 [cs.GR] 4.
[16] Jeff Pool, Anselmo Lastra, Montek Singh. (2011) "Precision Selection for Energy-Efficient Pixel Shaders." *HPG '11: Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. Pages 159–168. https://doi.org/10.1145/2018323.2018349.
[17] Lidong Xing, Taoli, Hucai and jungang. (2019) "Power Consumption Optimization for 3D Graphics Rendering." *Journal of Harbin Institute of technology*. Volume 26, Number. 1. DOI: 10.11916/j.issn. 1005-9113. 17036.
[18] Gene Wu; Joseph L. Greathouse; Alexander Lyashevsky; Nuwan Jayasena; Derek Chiou. (2015) "GPGPU Performance and Power Estimation Using Machine Learning." *IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. DOI**:** 10.1109/HPCA.2015.7056063.
[19] Joao Guerreiro; Aleksandar llic; Nuno Roma; Pedro Tomas. (2019) "DVFS-aware application classification to improve GPGPUs energy efficiency." *ACM Digital Library*: *Parallel Computing,* Volume 83, Issue C, pp 93–117.
[20] Songlan Wang, Ji Zhang. (2021) "Research and Implementation of Real-time Render Optimization Algorithm Based on GPU." *Journal of Physics: Conference Series:* 2136 (2021) 012059. DOI: 10.1088/1742-6596/2136/1/012059.
[21] Jurn-Gyu Park, Nikil Dutt, Sung-Soo Lim. (2021) " An Interpretable Machine Learning Model Enhanced Integrated CPU-GPU DVFS Governor." *ACM Transactions on Embedded Computing Systems*. Volume 20, Number 6, Article 108.
[22] Markus Schutz, Bernhard Kerbl, Michael Wimmer, TU Wien. (2021) "Rendering Point Clouds with Compute Shaders and Vertex Order Optimization." *Computer Graphics Forum.* Volume 40, Number 4.
[23] Pavel Peresunko , Denis Mamatin , Oleslav Antamoshkin, Evgeniya Peresunko, Alexander Nikitin. (2021) "Models of Experts for Shaders Estimation of Rendering Complex 3D Scenes in Real Time." *3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA). ©2021 IEEE*. DOI: 10.1109/SUMMA53307.2021.9632071.