TECHNICAL TRANSACTIONS | **CZASOPISMO TECHNICZNE**
MECHANICS | MECHANIKA
1-M/2013

JACEK PIETRASZEK*, MACIEJ KOŁOMYCKI*, ELŻBIETA KOCYŁOWSKA**

THE IMPACT OF CUDA TECHNOLOGY ON THE EFFICIENCY OF BLENDER RENDERER PROGRAM

OCENA WPŁYWU TECHNOLOGII CUDA NA WYDAJNOŚĆ PROGRAMU RENDERUJĄCEGO BLENDER

A b s t r a c t

The paper summarizes the quantity assessment of GPU processing and supporting CUDA technology on the efficiency of BLENDER rendering program.

Keywords: *CUDA*, *GPU*, *orthogonal design*, *design of experiment*, *BLENDER*

S t r e s z c z e n i e

W artykule przedstawiono ilościową ocenę wpływu zastosowania przetwarzania GPU i wspierającej to technologii CUDA na wydajność programu renderującego BLENDER.

Słowa kluczowe: *CUDA*, *GPU*, *plany ortogonalne*, *planowanie doświadczeń*, *BLENDER*

* PhD. Jacek Pietraszek, MSc. Maciej Kołomycki, Institute of Applied Informatics, Faculty of Mechanical Engineering, Cracow University of Technology.
** BA. Elżbieta Kocyłowska, Faculty of Interior Design, Academy of Fine Arts in Kraków.

1. Introduction

1.1. Rendering

In the last years the intense development of computer technologies had a strong influence on widening the possibilities offered by programs that are used to create photorealistic scenes, and that increased the hardware requirements even more.

The last step in creating computer graphics is the process of rendering. It is based on the analysis of the three-dimensional model or scene, which later on creates a two-dimensional (static or animated) outcome image. The module used in the process is called a rendering engine.

The development of 3D graphics reaches back to 1960, when W. Fetter, a designer working for Boeing Corp., generated very first orthographic images of a human silhouette using a computer and created the "computer graphics" term [1]. In 1963 I.E. Sutherland has proposed a system for interactive image generation, a computer screen called sketchpad in his doctoral thesis work [2, 3]. First three-dimensional wireframes were built from simple geometric shapes, which made it possible to view the model from all angles. HSR (hidden surface removal) algorithm, which allowed creation images of full solids, was created by Sutherlands' co-workers: D.C. Evans, C. Wylie, G.W. Romney and A. Erdahl [4].

With the development of technology, the next obstacle that computer graphics had to face, was the realism of obtained images. Improving the complexity of a scene without undue increase in geometry complexity – and at the same time, memory usage – became possible thanks to the introduction of increased number of polygons. It allowed for the formal complication of the mesh, but because of the simplified shadow system, the smoothness created with the increased polygon count was lost with a sufficiently close camera.

New shading model was created by H. Gouraud later on [5]. The algorithm was based on calculating the normal vector of every vertex in the model (in order to determine the angle of incidence of the light beam), calculating the colour of pixels of the vertices and linearly interpolating the values between them. Thanks to that, the obtained effect is smoother. The only flaws are still visible at the edges and with the way the light source is reflected on the model surface. The concept introduced by Gouraud was developed by B.T. Phong, whose algorithm [6] interpolates vectors normal to surface elements of the face and then determines the colours of each pixel. It allows us to obtain a very smooth surface, but the main problem is the increased time of image processing. In 1978 J.F. Blinn proposed the disturbed surface geometry [7]. Addition of a special black and white texture allowed to define which pixels had a perturbed normal, and that in turn allowed for to simulate the look of realistic objects in a much more precise way. Bump mapping [8] is still used in many media such as games or animations. Its development is displacement mapping [9], where both the pixel and the normal are transformed. With the development of the method, a technique called normal mapping was created. The problem of fading of surface details by the edges returned, as well as the usage of the processor and the memory increased. The solution to the problem was implementing a data structure, which created an illusion of a very complicated model, without further complication of its geometry. Innovations in the domain of computer technologies, starting from 1960, changed the way of thinking about computer modelling, which lead to the creation of new techniques and trends in its usage.

The level of rendering realism is described using lots of parameters, the most important which are:

- shading – the correlation of the colour and brightness of the object with the light,
- shadows and soft shadows – the way the light is blocked by the object's surface,
- reflection and refraction – the reflection and refraction of light by the object,
- graphic, optic transparency and opacity – the way the light is transmitted by a object,
- translucency – light scattering by objects,
- indirect illumination and caustics – the light reflected from other objects and focused as defined by material properties,
- texture mapping and bump mapping – giving the model material texture and unevenness.

The level of render realism is based on preforming very complex and labour-intensive calculations. For many years it was the main impulse in developing the graphics processor cards. It led to implementation, by NVIDIA Company, of multicore compute unified device architecture (CUDA).

1.2. CUDA computing technology

CUDA (compute unified device architecture) is a specific architecture of graphics processing units proposed and implemented in 2006 by NVIDIA Company [9]. The main advantage is the two-way communication with the program that is used within the memory of the main host and the ability to send a GP GPU (General-Purpose computing on Graphics Processing Units) program to the card's memory, where it will be used by its multi-core processor organized as a hierarchical symmetric structure. The number of cores varies from hundreds to thousands and the size of card's memory from several MiB to several GiB. GP GPU cores in CUDA architecture are equipped in a set of general computing programming instructions. It distinguishes it from existing solutions, in which only subsets of specialized commands for graphics processing were available.

A typical program using CUDA architecture is built from several pieces that are executed either on the main CPU (Control Processing Unit), or the GPU card's processor [10]. NVIDIA provides a C language compiler separating the code into two groups of instructions – for the CPU and GPU – during compilation. The code for CPU is a typical ANSI C code. The one used for GPU is extended by keywords identifying parallel data processing functions called kernels and functions associated with these data structures. Due to the high labour intensity of creating programs this way, tools that provide automatic identification of code fragments suitable for parallelization were developed. The leading role is played by the family of compilers distributed by The Portland Group for FORTRAN, C and C++ languages [11].

The modern multipurpose CPU processor contains at most a dozen cores that allow concurrent execution of instructions. The buffer memory blocks (called cache) and predictive blocks are complex. They enable concurrent execution of successive instructions within the same core on the basis of their compliance tracking. The case is different in the GPU processors since they contain from several dozen to thousands of cores working concurrently with the simultaneous sharing of resources such as memory and registers. The buffer memories in GPU processors are small, since their main purpose is to avoid collisions when

accessing shared memory, and not to store data. Processing on a single core is sequential and hyper-threading is not used. Improved performance is obtained not by the sophisticated configuration of pipeline instruction execution, but through multiplication of execution units.

The critical point for the performance of the programs that use GP GPU is the data transfer from the main computer memory to the card memory, and the other way around. In the worst case scenario it can cause the resultant performance of the entire program to be lower than just being limited to CPU processor, despite great efficiency of the card. It imposes large data structure organization and algorithm selection requirements, in order to minimize such transfers.

2. Materials and methods

2.1. Blender program

Blender is a program to model and render three-dimensional images and animations. It was created in 1995 by NeoGeo Company, and was developed further by Blender Foundation. It's available on different system platforms. The performance tests were made using 2.66a version for MS Windows 7 system.

2.2. Equipment and system environment

The rendering process was made on 3 CPU variants: 1 core × 3600 MHz, 2 cores × 2800 MHz and 8 cores × 3600 MHz. 4 NVIDIA GPU cards were used: GT440, GTX560Ti, 635MGT, GTX680. Due to availability of hardware, the measurements were carried out in 6 combinations CPU vs. GPU (Tab. 1)

Table 1

Combinations of hardware CPU vs. GPU

ID	CPU	GPU
1	1 core × 3600 MHz	–
2	1 core × 3600 MHz	680GTX
3	8 cores × 3600 MHz	–
4	8 cores × 3600 MHz	680GTX
5	8 cores × 3600 MHz	440GT
6	2 cores × 2800 MHz	635MGT

2.3. Methods of analysis

Result analysis was carried out according to the methodology of the design of experiment (DoE) [12]. Calculations were made using STATISTICA [13] and Minitab [14] software.

The object of study was treated as a two-input two-level full factorial. The directly measured value was the time of rendering. Due to the fact that this is a volume measured at the ratio scale (according to the Stevens classification of measurement scales [15]) directly

applying regression would lead to nonphysical results: negative times. For this reason, the initial data pre-processing conversion was performed by using mapping that transferred positive time values into unlimited range of natural logarithms of the rendering time.

The final analysis model was the following formula:

$$\ln(t_{ij}) = \mu + CPU_i + GPU_j + CPU \times GPU_{ij} \quad (1)$$

where:

μ – the average logarithm of rendering time,
CPU_i – the effect associated with *i*-th CPU variant,
GPU_j – the effect associated with *j*-th GPU variant,
$CPU \times GPU_{ij}$ – the effect associated with interaction between CPU and GPU.

This approach prevents the obtaining of non-physical values, yet allows the assessment of mean effects of the introduction of multi-core CPU and GPU cards.

2.4. Rendered scene

The object of rendering was the 3D scene consisting of 3 cups and a teapot (Fig. 1). The metallic, reflective surface of all four elements was a defining characteristic, while the background was a simple, smooth white surface. The whole scene was illuminated with 3 point light sources. The new rendering engine called Cycles Renderer, which is developed now by Blender Foundation, was used in the experiment.

The characteristic properties of the file were:

– samples set to 1000,
– light bounces set to: min. 0 and max.4,
– performance tiles set to 256 × 256 size.

Fig. 1. Rendered scene used in the efficiency test
Rys. 1. Kreowana sceneria zastosowana w teście wydajności

3. Results

Efficiency tests that involved rendering the test scene (Fig. 1) with different hardware settings were conducted. The obtained operation times are compared in the table (Tab.2) and shown on the graph (Fig. 2).

Table 2

Obtained Times of rendering process of the test scene

ID	Time, hh:min:sec
1	01:33:08
2	00:02:26
3	00:16:49
4	00:02:22
5	00:08:47
6	00:12:09

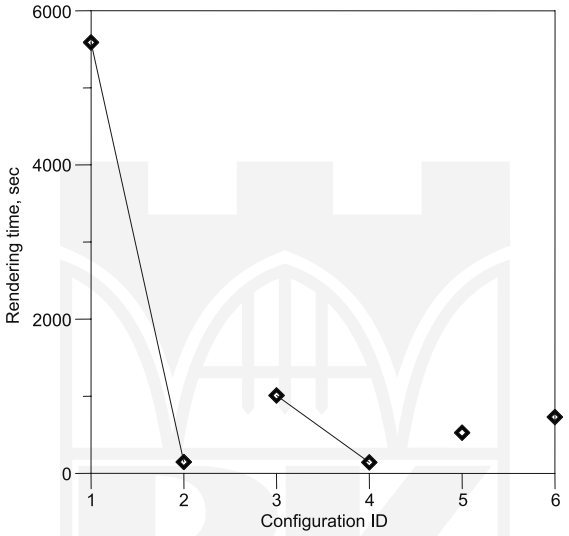

Fig. 2. Obtained test scene render time

Rys. 2. Czasy renderowania scenerii testowej

4. Result analysis

Particularly important information was obtained from the combination that used 680GTX card, because it exhausted all possible combinations of CPU and GPU. The application of DoE method to the results tied with configurations 1, 2, 3 and 4 gave a possibility to determine the main effects and the interaction effect (Tab. 3). It's important to remember that the shown effect values refer to the logarithmic scale.

The average main effects shown in Fig. 3 clearly indicate that a more significant time gain is obtained by using a GPU card than by increasing the number of cores in CPU processor.

Relatively significant interaction effect – at a level of the value of main CPU – has an antagonistic character. In practice, it means that simultaneous increase of the number of CPU cores and implementation of a GPU card has no technical and economic sense for this rendering software. Most probably it stems from the fact that the Blender software in

GP GPU mode does not enable CPU processor multithreading, and in turn its multicore capabilities have little impact. It can be clearly seen with the measured render times (ID 2 and 4, Tab. 2). The difference is only 4.5 s.

Table 3

Effect of the linear model with two-way interaction

Factor	Effect
const	6,37354
CPU	–0,87137
GPU	–2,79779
CPU × GPU	0,84062

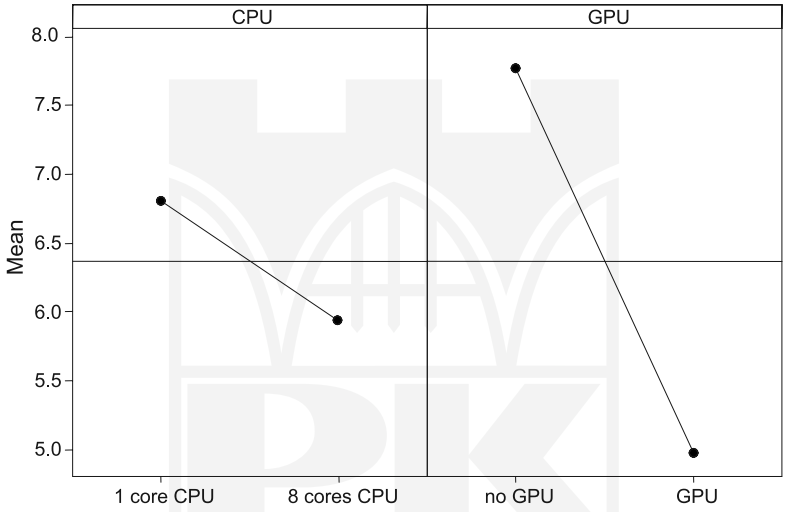

Fig. 3. Average main effects CPU and GPU (left axis – natural logarithm of rendering time)

Rys. 3. Średnie efekty główne CPU I GPU (oś lewa – logarytm naturalny czasu renderowania)

5. Conclusions

This paper presents a quantitative assessment of the impact of CUDA technology in application performance in the Blender rendering program. The selected test scene measurements were performed using a full-factorial design plan divalent. Also - for the two variants of hardware configurations - were measured for comparative purposes. The results of the measurements were analysed according to the methodology of experiment planning. Developed analysis indicates a much larger performance boost obtained from the use of GPU processor cards than from an increase in the number of CPU cores. Also it demonstrated futility, in case of Blender, of simultaneous implementation of multi-core CPU with a GPU card.

References

[1] Fetter W., *Computer Graphics*, Proceedings of the First Boston Architectural Center Conference, Boston MA, Dec. 5, 1964, 34-36.

[2] Sutherland I.E., *Sketchpad: A man-machine graphical communication system*, Ph.D. thesis, MIT Lincoln Laboratory Technical Report #296, January 1963.

[3] Sutherland I.E., Sproull R.F., Schumacker R.A., *A Characterization of Ten Hidden-Surface Algorithms*, ACM Computing Surveys 6 (1), 1974, 1-55.

[4] Wylie C., Romney G.W., Evans D.C., Erdahl A., *Halftone Perspective Drawings by Computer*, Proceedings of AFIPS FJCC, Vol. 31, 1967, 1-49.

[5] Gouraud H., *Continuous shading of curved surfaces*, IEEE Transactions on Computers C–20 (6), 1971, 623-629.

[6] Phong B.T., *Illumination for computer generated pictures*, Communications of ACM 18 (6), 1975, 311-317.

[7] Blinn J.F., *Simulation of Wrinkled Surfaces*, ACM SIGGRAPH Computer Graphics 12 (3), 1978, 286-292.

[8] Max, N.L., Becker, B.G., *Bump shading for volume textures*, IEEE Computer Graphics and Applications 14 (4), 1994, 18-20.

[9] *NVIDIA unleashes CUDA technology*, Computer Graphics World 29 (12), 2006, 4-4.

[10] Sanders J., Kandrot E., *CUDA by Example: an Introduction to General-Purpose GPU Programming*, NVIDIA Corp., Boston 2011.

[11] *PGI Fortran and C/C++ for 64-bit x64 processor-based systems Version 13.3*, The Portland Group, Inc., Two Centerpointe Drive, Suite 320, Lake Oswego, Oregon 9703.

[12] Montgomery D.C., *Design and analysis of experiments*, John Wiley & Sons, New York 1997.

[13] Statsoft, Inc., *STATISTICA (data analysis software system), version 10* (www.statsoft.com – 2011).

[14] Minitab, Inc., *Minitab, version 16* (www.minitab.com – 2012).

[15] Stevens S.S., *On the theory of scales of measurements*, Science 103, 1946, 677-680.