The Experience of The "Dust" C Code Implementation on PS3

Vladimir Savchenko
Faculty of Computer and Information Sciences, Hosei University
Tokyo, Japan
vsavchen@k.hosei.ac.jp

Abstract

This paper describes an experiment in learning computer graphics algorithms through parallelizing three C code examples: Ray Tracing, Voxel Visualization, and Image Warping on PlayStation3 (PS3). An implementation of geometry subdivision of data for PS3, which is a physically distributed memory machine, provides, for example, display of a $120\times120\times120$ voxel slab at 30 frames per second. We notice that all the code conversions were done by novices without Cell programming experience in a quite short time.
Keywords: PlayStation3, Cell, Educational approach

1. INTRODUCTION

Computer graphics (CG) education is increasingly important to supply the growing needs of the film, games, and virtual reality industries. Learning goals in CG and concepts that would be generally expected in even a beginning course could be covered at several levels, from the algorithmic and mathematical to a general conceptual treatment followed by programming realization.

During the last few decades, many educational practitioners have increasingly turned their attention to *constructivist* models of learning. Constructivist pedagogies have been successfully used in learning situations where the acquiring of a deep understanding of a subject is required [1],[2].

A common educational approach is to use books and other educational components to give students a theoretical foundation, and a low-level programming library like OpenGL or use Java – developing of animated applets – for practical exercises and projects. This paper describes our experiment of using a constructivist-oriented approach in computer graphics education. Our main premise or "learning model" is that the fundamental concepts and algorithms of CG can be acquired through the experience with parallelizing old C codes related to CG. Also, we present time performance results for running three CG applications using parallel processors of PlayStation3 (PS3) to achieve almost linear speed up of calculations.

Chips with two processors, or cores, are now standard, and four-core chips are increasingly common. But right now, even with Dual-Core machines being common, most PC programmers don't know how to use parallel processing. In spite of the obvious fact that microprocessor companies take a huge risk in adopting the multicore strategy, recently, two rather cutting-edge hardware consoles (XBox 360 and PS3) with heavy parallel processing power have been launched.

The Xbox 360 (the second video game console produced by Microsoft) provides uniform memory and three completely similar PowerPC processors. The PS3 uses the cell microprocessor developed by IBM, Sony and Toshiba as its CPU, utilizing seven of the eight "synergistic processing elements" (SPEs). In PS3, six SPEs are only available because one SPE is reserved by PS3 OS and the other is disabled for improving yield. At peak performance, the chip reaches up to 200 GFlops. It has a heterogeneous multicore architecture that consists of a Power PC processor along with SPEs that are specialized for data processing. PS3 is not only one of the most advanced gaming devices in the world, oriented on polygon processing, but it is thought that PS3 is also the perfect tool for CG and image processing applications having much quantity of data processing in such applications such as the real time rendering of high-definition images, medical image processing, encoding and/or decoding a digital data stream, and in many various applications not related to games. Moreover, an improved version of the cell chip is also the basis of Roadrunner - the fastest computer in the world [3].

Recently, many universities (for instance, the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology [4]) in order to provide a framework for research and teaching in the areas of multicore development and parallel programming have started projects on the PS3. The U.S. Air Force bought 300 PS3s for a research purpose [5]. IBM is also running a programming contest for college and university students in 25 different countries.

Our goal is three-fold, first of all, as we mentioned above our goal is to attain a deep understanding of a subject, in particular CG algorithms. Second, is to give undergraduate students the chance to get hands-on experience on PS3 hardware. Many processing algorithms such as voxel visualization, image processing are "embarrassingly parallel" or almost ideally suited to parallelization while even a simple ray tracing or ray marching for implicitly defined objects creates a problem with load balancing - distributing work dynamically. The fact that Cell-based systems are attractive for CG applications stimulated us to implement PS3 for educational and research purposes.

Finally, we are interested in the use of so called "dust code" in finding a way to make that old our programming code useful to provide a framework for research and teaching in the areas of geometry modeling and CG.

Parallelization of CG applications is mainly achieved by inserting directives in the source code.

The remaining part of this paper is arranged as following: information and related literature review on the key techniques used is given in section 2. Section 3 presents the benchmark models. PS3 implementation is discussed in Section 4. The benchmark results are given in Section 5. And Section 6 concludes the paper.

2. BACKGROUND INFORMATION AND RELATED WORK

Multiple data (SIMD) ray kernels and distributing work dynamically can produce ray-traced images (without detail-

adding of reflective maps and shadows) at interactive frame rates for complex scenes containing more than one million polygons [6], see also [7].

PS3 Cell contains one Power Processor Element (PPE—a standard PowerPC CPU) and eight Synergistic Processor Elements (SPEs). SPEs live inside *elements* on the Element Interconnect Bus, which also talks to the Memory Interface Controller and I/O controllers. This path is very fast, DMA between elements and the 256 MByte XDR system memory achieves a peak rate of 25.6 GBytes/sec. The Cell's SPEs provide very fast execution, but their local store holds only 256 Kbytes. Typically processing on the SPE involves shuttling data to and from main memory using DMA. DMA double buffer allowing the SPE's memory flow controller to coordinate the best order of operations for loading and storing is used. For more information, see [8],[9].

Cell Broadband Engine processor provides a cost-effective alternative to current ray-tracing options and helps free studios from the limitations of low-fidelity, raster-based graphics approximations in their final production frames, see [10].

3. MODELS

For some computer graphics applications, sequential implementations are too slow since computation grows as the square or cube of a modeled space. Faster approaches often rely on subdivision of 2D or 3D space where a partitioning of space is used to accelerate calculations. This partitioning - often simple geometry subdivision of 2D screen - in many applications forms the basis of scalable multi-processor approaches to visualize a scene or simulation.

3.1 EXAMPLE I. RAY TRACING

Ray tracing is one of the techniques to draw an image of a geometry scene. In our implementation of a network of Cell processors, a renderer calculates and renders a scene that consists of multiple implicitly defined spheres that includes coordinates of geometry objects, positions of a source of light and viewpoint, and others. The renderer can finely express reflectance, transparency, and refractions. This stage is the most complicated and CPU-intensive: calculation of lighting and shadowing of the objects. The ray tracing procedure is pretty simple in some way. It goes from one pixel to another; however, computational complexity naturally increases with respect to the image size. Also, when one ray communes with a lot of objects, hidden surface removal performed by projecting nearest point from a viewpoint on plane of projection, calculating reflectance maps and many others related to ray tracing techniques are computationally expensive.

3.2 EXAMPLE II. VOXELIZATION AND VOXEL VISUALIZATION

In many CG, animation, and geometry applications continuous volume is represented by a 3D grid of voxels, see, for example, [11]. This representation is convenient for a number of applications such as mixing synthetic objects into medical imagery, 3D-visualization of tomography volume data, reverse engineering, and scientific visualization. In these fields, 3D data is measured or computed at a large number of points in 3D space and then rendered to produce informative images. The main weakness of the voxel-based approach is that it is expensive in terms of computation and memory. Interaction is an effective technique for guiding the production of these images, allowing a doctor or scientist to navigate to various regions of interest and adjust the transfer function used to map the data into colors. Unfortunately, due to the size of these 3D datasets and the processing required to render them, interaction has only been possible with relatively small data sets and not of the most general organization. In particular, regular rectilinear grids of volume elements, or *voxels*, are well-suited to acceleration by modern graphics hardware. This hardware can take the form of a dedicated volume processor [12] or a more generic 3D graphics accelerator [13], see, also [14].

For data of the right form and size, this hardware makes interaction possible. Nevertheless, volume data often comes in many sizes and forms that creates difficulties of using specific volume processors. Also, interaction is an effective technique for guiding the production of new volume models, see, for instance, [15] and references therein, that is considered as a continuation of the project.

3.3 EXAMPLE III. IMAGE WARPING

One of the image processing techniques is the image warping, see, for instance [16]. Image warping is the process of digitally manipulating an image where the image coordinates are transformed from pixels (i,j) to coordinates (x,y) of destination image. Early interest in this area dates back to the mid-1960s when it was introduced for geometric applications, for example, distortion correction of lenses in remote sensing. It has found a new surge of interest from the CG field. This is largely due to increasingly powerful computers that make warping a viable tool for image synthesis and special effects. The most usual technique used in warping algorithms is based on spline interpolation. So, conventional image processing takes much time because the computer calculates a mapping over an image by calculating a spline; in our application, we apply warping technique based on the use of radial bases functions [17] that allows effectively produce forward and inverse mappings. Thus it seems reasonable and interesting for students to investigate opportunities of the use of PS3 and study numerical algorithms for calculating maps from the destination image to the source image in parallel and to check up speedup with respect to processing speed on a single processor.

4. PS3 IMPLEMENTATION

Load balancing is a central theme of multiprocessor-based systems for managing memory layout and assigning processors to jobs. In considered here applications, fortunately, there is no spatial and time dependency between processed data, therefore processors can be updated in parallel in any order. The master-slave (or host-node) paradigm is used for load balancing to speed up calculations. It means that a separate "control" program (on PPE) called the master is responsible for process spawning, initialization, collection, and display of results. The slave programs (on SPEs) perform the actual computation involved; their workloads are allocated by the master dynamically. There is no room to show relative simplicity of software development. Nevertheless, let us notice that the main master program includes two functions – startRayTracing00 and getJob – which provide , first of all, a preparation for SPE by spe_image_open, spe_context_create, spe_program_load, and pthread_create functions which are included in startRayTracing00. Secondly, ray tracing is performed by SPEs where each thread takes control of sending and getting necessary data. PPE is also used for ray tracing as well as SPEs. Processing data are distributed by getJob

function. Synchronization is taken on each thread by using the mutex method in getJob function that provides correct distribution of job assignments. SPEs use DMA transfer that allows to access system memory for writing RGB data directly to a pixels array. It makes possible performing a complete image directly by SPEs without composing a final image by PPE.

The DMA transfer controller is used to send data going to and from the main memory. 16 bytes multiple sequence alignment is used both on local store and main memory to provide the DMA transfer. Moreover, the size of data that can be sent by one time is up to 16 Kbytes. Because information transmitted in the ray tracing application is integer type (four bytes that show coordinates of the pixel and the color of the pixel) 4,000 of data or less can be sent at once. The automatic generation of accurate multiple alignments is potentially a daunting task. Nevertheless, experiments show that an improvement of the processing speed is not achieved when the number which can be sent is increasing from 1,000 to 4,000 while the processing speed is improved when this value is increasing from 1 to 1,000. Thus, size of parcel data to be sent at once was chosen to be 1,000. Let us notice that Phong [18] model was used.

The SPEs are the Cells short-vector SIMD workhorses. It is important to exploit SIMD properties of SPEs as efficiently as possible to speed up the processing speed of a SPE program. While it is not quite clear how we can harness the power of the Cell and PS3 by using SIMD operations in ray tracing application, existing voxelization/visualization C code can be easily modified to add new capabilities for model voxelization and real time visualization.

5. PERFORMANCE EVALUATION

The processing speed measurements were produced for the Cell processor and the Pentium processor for the examples shown in Figures 1, 2, 3.

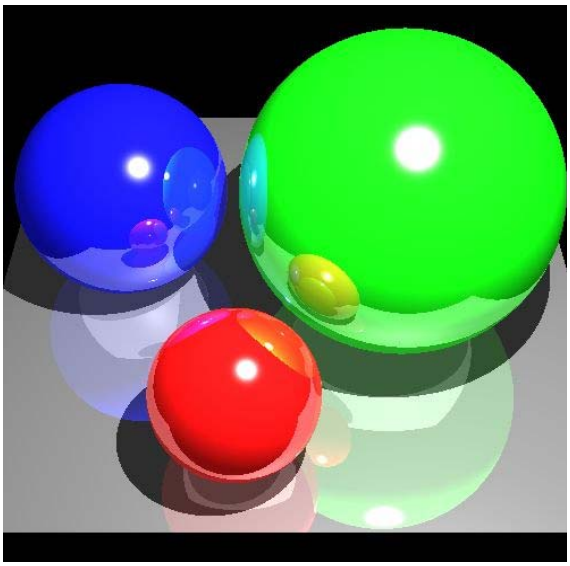

Figure 1. Ray-traced objects are rendered with detail-adding of reflective maps and shadows

The performance results are shown in Tables 1,2,3; time is given in seconds.

Table 1. Measurement results for Ray Tracing

#PUs	Processing time
Pentium 4 3.2GHz	7.81
Pentium M 1.6GHz	14.27
PPE	5.60
PPE + 1SPE	3.36
PPE + 2SPE	2.44
PPE + 3SPE	1.93
PPE + 4SPE	1.60
PPE + 5SPE	1.36
PPE + 6SPE	1.18

The image of a voxel model (defined by subtraction and union operations of three implicitly defined spheres) produced by the developed software algorithm is shown in Figure 2. The execution time of voxelization and vizualization by using 6 processors is 0.1097 sec. The execution time of processing without using Cell was 1.3075 sec. Thus, the speed of processing using the cell is about order of magnitude higher than without using the cell processor that is explained by using SIMD operations for voxelization of the model.

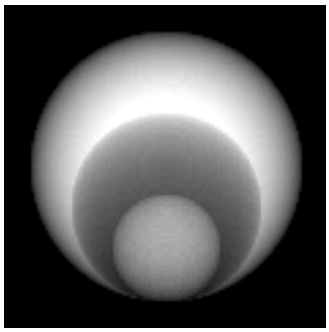

Figure 2. Result of voxel visualization. Size of the voxel slab is 120x120x120

Table 2. Processing characteristics

#PUs	Time of voxelization	Time of visualization
1SPE	0.4268	0.1103
2SPE	0.2177	0.0688
3SPE	0.1438	0.0473
4SPE	0.11	0.0459
5SPE	0.0891	0.0428
6SPE	0.07	0.0397
PPE	1.1938	0.1137

Table3. Warping test results

#PUs	Processing time
1SPE	1.96
2SPE	0.99
3SPE	0.65
4SPE	0.50
5SPE	0.41
6SPE	0.35
PPE	3.05

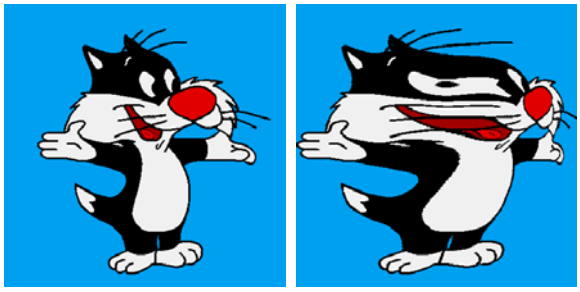

Figure 4. Warping of "Neko" image. Image size is 407×411

Let us notice that almost similar programming approach discussed above for Ray Tracing was used for voxel vizualization and image warping.

6. CONCLUSION REMARKS

The PS3 Cell processor has unique architecture; therefore a special treatment of programs developed for conventional computers is needed. Comparison of performance of Pentium 4 3.2GHz processor and using 6 Cell processors shows that the Cell processor system demonstrates about 7 times performance speed up of Pentium 4 processor. It has been said that the processing speed of Cell processor is from several to tens times faster than Pentium D processor. Nevertheless, we were not able to attain ten times acceleration of ray tracing by the use of Cell processor. Without any doubt, we can state that we succeeded in the using Cell processor with respect to Pentium 4 which performance is lower than Pentium D. Cell processor shows about 7 times performance speed up of Pentium 4 processor and about 12 times performance speed up of Pentium M processor. Moreover, because the processing speed has improved by increasing the number of used SPEs almost linearly, it seems that multiple data alignment procedure is reasonable and accurate.

However, the improvement of the processing speed by SPEs was less than our expectation. We suppose that there are two reasons of it. First one is related to uneasy question how to assign processing operations. SPE works well on simple operations, that is conditional branches must be eliminated. The second is that SIMD operations were not used in the software development of ray tracing and image warping algorithms. In future, we are planning to rewrite the source code of the SPE`s program which uses only scalar operations by using SIMD operations. It is a serious problem because of the initial code was not accustomed to the idea of the vector data. It seems that without proper compiling tools SIMDimization process can be a serious obstacle for a wide application of Cell processors.

The results presented are from a short initial evaluation. There is still much scope for optimization and these results should be considered as representative of the performance that can be obtained from the first naïve parallelizing of old C codes.

The main lesson learned is the fact that students with no background in Cell parallel programming, were able to get their projects done from scratch in just about three months. This largely goes to show that we can expect rather simple migration of "embarrassingly parallel" CG applications to the Cell processor.

ACKNOWLEGEMENTS

I would like to thank my former undergraduate students Y. Mukai, T. Suganami, and I. Tatsuma, for their encouragement to participate in this project.

REFERENCES

[1] C.T. Fosnot, Ed. Constructivism. Theory, Perspectives, and Practice. Teachers College P, 1996.

[2]E. Von Glasersfeld. Radical Constructivism and Teaching,2001,http://www.umass.edu/srri/vonGlasersfeld/onlinePapers/html/geneva/

[3] http://www.lanl.gov/roadrunner/

[4] http://www.ps3cluster.org/index.html

[5] https://www.fbo.gov/

[6] C. Benthin, I. Wald_, M. Scherbaum, H. Friedrich, Ray Tracing on the Cell Processor**, ,** IEEE Symposium on Interactive Ray Tracing**,** 2006, pp. 15-23.

[7]Real-Time Ray Tracing on the Playstation3 Cell Processor, http://eric_rollins.home.mindspring.com/ray/ray.html

[8] IBM, Cell Broadband Engine Programming Handbook, Version 1.1, April 2007.

[9] IBM, Cell broadband Engine Architecture, Version 1.01, October 2006.

[10]Interactive Ray Tracer for Cell Broadband Engine, http://www.alphaworks.ibm.com/tech/irt?open&S_TACT=105AGX59&S_CMP=GRsite-jw18&ca=dgr-jw18awirt

[11] B. Lichtenberg, R. Crane, and S. Naqvi, Introduction To Volume Rendering, Prentice Hall PTR, 1998.

[12] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro Real-Time Ray-casting System. Proceedings of SIGGRAPH 99. pp. 251-260.

[13] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographics Reconstruction Using Texture Mapping Hardware. Proceedings of 1994 ACM Symmposium on Volume Visualization. pp. 91-98.

[14] U. Neumann, Interactive Volume Rendering on a Multicomputer, Proceedings of the 1992 Symposium on Interactive 3D Graphics, SI3D '92, March 29 - April 1, 1992, Cambridge, MA, USA. ACM, 1992, pp. 87-93.

[15] T. A. Galyean and J. F. Hughes, Sculpting: An Interactive Volumetric Modeling Technique, Computer Graphics, Vol. 25, No. 4, 1991, pp 267-274.

[16] G. Wolberg, Digital Image Warping (Systems), Wiley-IEEE Computer Society, 1990.

[17] V.A. Vasilenko, Spline Functions: Theory, Algorithms,, Programs, Nauka Publisher, Novosibirsk, 1983, (in Russian)

[18] J. F. Blinn, "Models of Light Reflection for Computer Synthesized Pictures". Proc. 4th annual conference on computer graphics and interactive techniques, 1977, pp. 192-198.