PARALLEL SCENE SPLITTING AND ASSIGNING FOR FAST RAY TRACING

Liberios VOKOROKOS, Eva DANKOVÁ, Norbert ÁDÁM
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic,
e-mail: liberios.vokorokos@tuke.sk, eva.dankova@tuke.sk, norbert.adam@tuke.sk

ABSTRACT

Photorealistic rendering is very often used area in computer graphics. It is used to display human body in medicine, buildings in architecture and also in other areas of our life. For photorealistic rendering are very important powerful computer, than it is relative difficult on computing. To render a 3D object in computer graphics a high efficient computer is needed, which enable user to work with graphic information on higher level. In some cases of data visualization rendering can slow-down the whole proccess of computing. One possibility how to solve this problem is to apply photorealistic rendering on distributed systems. Designed system is outgoing from static assigning and sharing of scene based on areas, so that the time needed for rendering doesn't affect the whole time. Through this architecture rendering should be more effective according to time in distributed systems.

Keywords: *distributed systems, photorealistic rendering, raytracing, primary ray, secondary ray*

1. INTRODUCTION

Computers graphics is area, which is still in progress. Development of computer graphics is conditional especially by development of computer equipment whereby available needed power for computing by visualization in computer graphics is. Imaging of 3D objects is area of computer graphics, which is used in different scopes [11].

One of the main points, which was important for computer graphic, was introduction of shadowing algorithm, which enable to change color shade depending on location of light source. Imaging was relatively realistic, but it did not enable to imagine shadows and reflexes on figure surface. Therefore was evolved algorithm, which allows photorealistic 3D rendering of scene – ray tracing.

Ray tracing is very calculation demanding and that is why have begun searching for solution, how to speed-up this algorithm. One option is to run this algorithm on distributed systems. Right selected method for assigning several parts of scene in agreement with their difficulty to processors accelerate computing of scene. Thereby is photorealistic rendering more effective.

One of the most important parts by projecting is preparation part. System should have the ability to detect errors and to continue operating properly in the event of the failure of (or one or more faults within) some of its components. Thereafter a parts of scene followed by there difficulty will be assigned to the processors, so that to the most powerful processor will be assigned the most difficult part. Article will handle about projecting of photorealistic rendering in distributed systems with fixation on preparation part. Projected system was realized at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics, Technical University of Košice.

2. DISTRIBUTED SYSTEM

For distributed system are used different definitions in literature. However no one is adequate and no one is consistent with others. Most often used is following characteristic [4]:

„Distributed system is a system of independent computers, which seems to user as autonomous coherent system.“

This definition has to aspects. The first one is: computers are autonomous, and the second one is: users think that they are working with autonomous system. On hand this definition is possible to deduce following characteristics of distributed system (DS) [4]:

- Differences between computers, communication details between them and internal organization of DS are hidden prior to users.
- Interaction between users and system is consistent and uniform, aside from time and place, where it is realized.
- Good DS are easy to expand.
- Good DS are high accessible (not always).

Except introduced characteristics of distributed systems are needed that the distributed system will fulfill some defined standards:

- *Sharing of facilities* – Systems should enable to several applications to share system resources (hardware, data).
- *Parallelism.*
- *Explicitness* – specification of system and all his interfaces are public known.
- *Transparency* – user (and in principle neither programmer) should not know if used resources are local or remote.
- *Fault tolerance* – system should have the ability to detect errors and to continue in work after one part of distributed system will be.

In every distributed system, also if it consists of several CPU, are several methods how can hardware be organized. Especially in terms of how single parts are with one another connected and how they communicate.

In term of memory sharing, computer architectures can be divided in two categories. The first group are computers, which have shared memory, and are called multiprocessors. And the second group are computers which have not shared memory and are qualified as multicomputers. Main difference between them is, that by multiprocessors exists only one space for physical address and this space is shared to all CPUs. Unlike to multiprocessors, by multicomputers has every computer his oven memory [1].

Compared with multiprocessors building of multicomputers is relatively easy. Each CPU has directly connection to his own local memory The only one problem is the way how the CPUs communicate with one another. By multicomputers based on bus system, are processors connected through multiple shared access network as Fast Ethernet. By multicomputers based on switching, message between processors is route through crossing network. There are projected several different typologies, where to most distinguished belongs hypercube and grid (Fig. 1) [4]. Selection of system from view of hardware is important for effective behavior of system. In project is used network environment (one another connected computers through network LAN), which build distributed system. This distributed system is also called cluster [4].

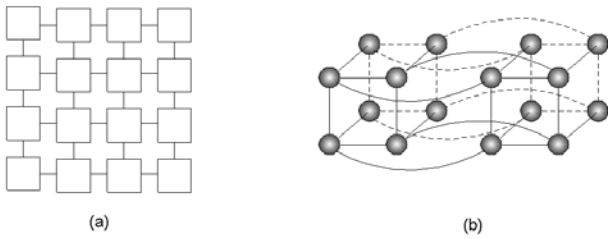

Fig. 1 (a) Grid, (b) Hypercube

Hardware is important for distributed systems, but software is that, what define how distributed systems looks like. At first it is used as resources manager for primary hardware. It allows users and applications to share sources like CPU, memories, peripheral equipment, networks and data of all art. In the second rank software facilities compose environment, in which is program executed. Also in this environment directly influence program language, in which is this program written. Projecting program is built in MPI environment (Message Passing Interface).

Operating system for distributed systems can be divided in two categories. First category consists of hard joinable systems and second category consists of free joinable systems. In the firs group operating systems in principle is trying to stay up autonomous global view on sources, which it manage. The second category of systems can be understood as a group of computers with his own operating system. These operating systems communicate together, so they can share their oven sources and services to other. In project will be used the second category of systems [1].

3. PHOTOREALISTIC RENDERING

In computer graphics the main goal of photorealistic rendering is to build image of 3D scene, which would be practically indiscernible from photography of the same scene. Photorealistic methods are trying to obtain photorealistic image, although in the end effect the image can by different from reality. Methods of photorealistic rendering are today so developed, that they are able to show very complicated and detail models of scene in a quality, which is indiscernible from photo.

The most photorealistic methods are based on algorithm ray tracing. Ray tracing is algorithm, which is based on way tracing of light ray through scene, their cross and ray reflection from objects up to environment. It exists many ways how to write such ray tracer [6].

Conception of ray tracing comes from optics. In contrast to optics in computer graphics is the process of light extension simulated in inverse progress. Ray is reflected from observer position through pixel in display plane to the scene. Therefore are divided in two kinds [6]:

- Ray tracing of first rank (ray-casting) – it is projected only point on surface of closest object.
- Ray tracing of higher rank (ray-tracing) - tracing does not end by finding of the closest point, but it carry on tracing of next rays. These rays are generated followed the measure of reflection or transparency of object.

Ray tracing can display on object surface imagery of other objects and also their shadows. For separate rays in ray tracing is used following terminology:

- Primary ray – is sending out from observer place through point of imaging window. Number of primary rays is given by pixel number of imaging window (Fig. 2).
- Secondary ray – is generated after the primary or secondary ray fetch up on object. In this fall can spring up two kinds of rays. The first one is reflected ray and the second one is refracted ray. If there is a high recursion depth, the number of secondary rays will be higher than number of primary rays (Fig. 2).

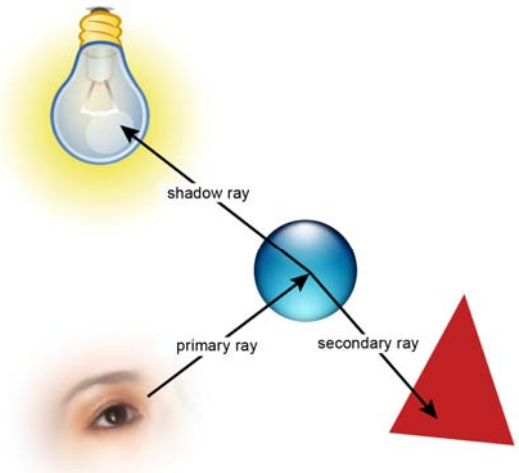

Fig. 2 Primary, secondary, shadowing ray in ray tracing

Shadowing ray – is sending from point, where fetch up primary or secondary ray (Fig. 2). It is used to detect if there is any obstacle between this point and the light source. Number of shadowing rays is generally higher than

number of secondary rays. By fetch up of ray on object surface are generated as many shadowing rays as many light sources are in scene.

Ray tracing is primarily procedure for covering calculation. It is used for layering detection of objects, which are placed in observant plain of eye. Algorithm is working with data structure, also called ray, which specifies initial point and ray course in space. For each pixel is computed ray course, which leads from eye to displayed plane. For each graphic element of scene will be detected a conjunction. In this conjunction ray intervene a graphical element and than will be computed distance from eye to conjunction. So visible is object with the smallest distance from eye [7].

```
Follow ray(ray R, recursion H)
```

- Find ray conjunction P with the nearest element
- Till conjunction P don't exist (ray leaved scene space), assign ray R background color and quit
- To each light source send from point P shadowing ray
- Evaluate lighting contributions in point P from each do not hide light sources
- Till recursion depth H did not overpass max. depth of tracing, send:
    1. reflected ray Rr calling Follow_ray (Rr, H+1)
    2. refracted ray Rt calling Rt Follow_ray (Rt,H+1)
- To ray R assign final color as summary of light result, color of reflected ray Rr and colors of refracted ray Rt

4. SPLITING SCENE METHODS

By photorealistic rendering in distributed system is required to elect right method for spliting scene. Spliting is achieveing only on one processor. Therefor is needed to choose efective method, which will be gainful from time point and also from computing sverity point. One possibility is to divide scene based on areras and thereafter to distribute this areas to procesors through static distributing. This static distributing of areas is based on areas dificullty.

Static scene distributing rests in beforeahad assesment, which prcessor will work with wich area. Therefor is needed to know how are processors powerfull and which parts of scene are dificult for computing. Following this onformation parts of scene can be assigne to processors. This assigning is based on scene parts priority and on processors power.

Projecting system for photorealistic rendering has modular structure, where under modul we understand dividing functions in to groups. This groups have fised input and output. Moduls are tougether connected and they pass informations through shared files.

System through raytracer on each node will generate scene in small graphic resolution and than it will render it in parts. During rendering it measure times, which are needed for computing of scene parcial parts.

Based on this times wil be built a table. In this table is parts sequence in agreement with time they needed for rendering (S_n). Also will be built a table of processors according to times, they needed to render the whole scene in small graphics resolution (R_p). Thereafter is each part of scene assignet to processors based on algorithm, which is shown on Figure 3:

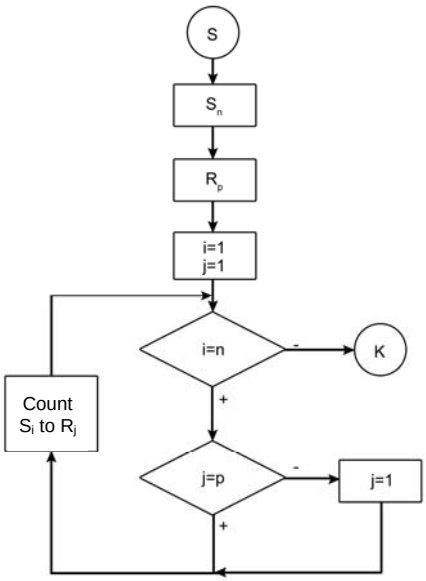

Fig. 3 Assigning algorithm of scene parts

Thereafter is started up evaluation modul, which function is to regulate node ranking in outputing file. On this modul is connecting statistic modul, which through web interface display statistics. This statistics pertain to nodes, partial results, computing time of rendered scene.

Starting-up of moduls is executing in chronological sequence (Fig. 4) and output is generated result. This result is saved in aktual file of operating system, in which is running this program.

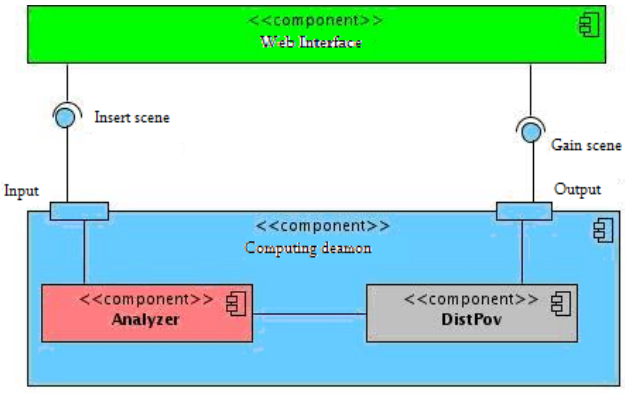

Fig. 4 Flow diagram of program

In this flow diagram introduce Web interface input and output. Analyzer is preparation part of system and DistPov is the main computing part.

Preparation part consists of 3 modules. The modules are evaluative module, preparation module and manager module. Preparation part is very important part of complex program.

Program is applied on scene Mini_demo (Fig. 5), which origin comes from author Gilles Trans, which was published on web <www.oyonale.com> under license Creative Commons By Attribution (<http://creative

commons.org/licenses/by/3.0. Scene is relatively difficult for computing, because there are many reflections from light source. Also there are some transparent surfaces, which also affect difficulty of the computing.

Fig. 5 Sample of scene mini_demo.pov©Gilles Trans, www.oyonale.com

Scene was rendered on 1, 2, 5, 10 and 15 nodes placed on Department of computers and informatics at TU of Košice with following configuration: Intel®Pentium® 4 CPU 2.60 GHz, 1 GB DDR. It was rendered by graphics resolutions 1024 x 768. Particular times of scene rendering, values of accelerating, efectivity and overbalance are introduced in Table 1.

Table 1 Results by scene mini_demo.pov 1024 x 768 (n = pixel group of incoming picture, p = number of processors)

Graphics resolution 1024 x 768					
Nodes	1	2	5	10	15
Time [s]	6002	3869	1553	764	534
Partial times [s]	6002	3869, 3868	1553, 1543, 1517, 1550, 1536	764, 754, 734, 731, 735, 762, 752, 754, 742, 737	534, 532, 469, 453, 508, 527, 479, 469, 491, 502, 506, 523, 519, 529, 533
Speedup $S(n, p)$	1,00	1,55	3,86	7,86	11,24
Efficiency $E(n, p)$	1,00	0,78	0,77	0,79	0,75
Overbalance $L(n, p)$	0,00	0,01	0,02	0,05	0,21

For ability to compare values receives from projected system , it was execute measure of scene rendering by the same conditions, but based on other method of assigning scene to nodes. This method is based on static weighing of computer workload and naive decomposition. Naïve decomposition consists in splitting picture in NUMBER_P continuous pixel segments, where P is number of processors. Than number of segments will be assign to the node for executing. In trivial situation NUMBER=1. Balancing through naïve decomposition very suffers under effect of scene locality. So considering to scene layout are some segments more difficulty for executing then other. This has for effect high overbalance.

Table 2 Values by static weighing of computer workload and naïve decomposition 1024 x 768 (n = pixel group of incoming incoming picture, p = number of processors)

Graphics resolution 1024 x 768					
Nodes	1	2	5	10	15
Time [s]	5976	4325	2622	1554	1118
Partial times [s]	5976	1891, 4325	657, 734, 1602, 2522, 1476	324, 349, 381, 356, 612, 847, 1554, 1021, 841, 521	197, 240, 220, 243, 284, 307, 362, 582, 658, 737,1118,767, 789, 526, 461
Speedup $S(n, p)$	1,00	1,38	2,28	3,84	5,34
Efficiency $E(n, p)$	1,00	0,69	0,47	0,38	0,35
Overbalance $L(n, p)$	0,00	1,29	2,84	3,80	4,67

By comparing tables 1 a 2, in which are results from scene rendering by both methods, can be point on efficiency and speedup of projected system. For better visualization are results also shown in following graphs (Fig. 6 and Fig. 7).

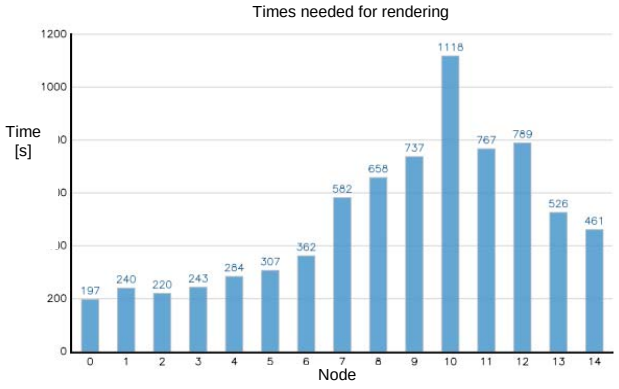

Fig. 6 Times graphical evaluation by rendering of scene by graphics resolution 1024 x 768 after applying of static rendering with naïve decomposition

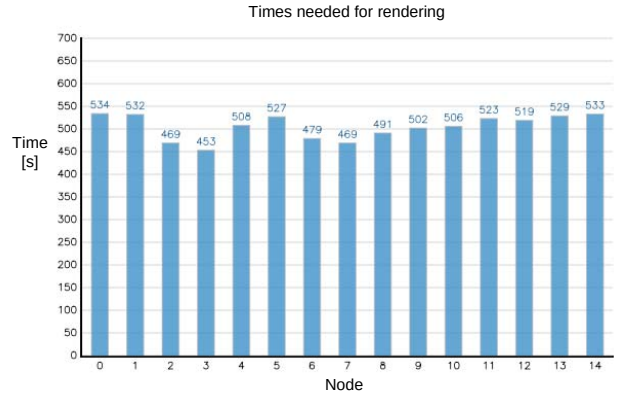

Fig. 7 Times graphical evaluation by rendering of scene by graphics resolution 1024 x 768 after applying of projected system

From graphs can be seen that after application of projected splitting based on areas, times needed for rendering of scene parts are more balanced on individual nodes. It can be also point on differences between nodes. By rendering with naïve decomposition is this difference about 10^3 s. This difference after application of projected system based on areas is less than 10^2 s. Based on these results is possible to evaluate that processors are more balanced and therefore is scene assigning more effective.

Visualization domain thanks photorealistic rendering gain big visualization quality and therefore is used in different areas as architecture, medicine, projecting, virtual reality and so on. Application of photorealistic rendering on distributed system will be expressed on efficiency of parallel process and on speed-up of computing. So that time needed for scene rendering to make smaller, it is needed to make smaller partial times on individual nodes. Algorithm created exactly for distributed system enable better to use whole system.

One of the goals was to reach better time for scene computing. This was obtaining with application of photorealistic rendering on distributed system. Within the experimental verification was achieve, that by using of projecting methods is global computing time on distributed system minimize depending on number of processors. Also it depends on methods used by scene splitting and scene assigning to nodes in system.

ACKNOWLEDGMENT

This work was supported by the Slovak Research and Development Agency under the contract No. APVV 0073-07 (34%), also the authors are pleased to acknowledge the financial support of the Cultural and Educational Grant Agency KEGA of the Slovak Republic under grant No. 3/7110/09 (33%) and of the Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120020) supported by the Research & Development Operational Program funded by the ERDF (33%).

REFERENCES

[1] VOKOROKOS, L.: Digital Computers Principles, Budapest, pp. 232, Typotex 2004, ISBN 9639548 09.

[2] JELŠINA, M.: Architecture of computer systems, Elfa s.r.o., Košice, pp.467, 2002, ISBN 8089066402.

[3] SOBOTA, B.: Computer graphics and C language KOPP, České Budějovice, 1996.

[4] TANENBAUM, S. A. – VAN STEEN, M.: Distributed systems – Principles and paradigms, Prentice Hall, 2002, ISBN 0132392275.

[5] VOKOROKOS, L.: Principles of Data Flow Controlled Computer Architectures, Copycenter, s.r.o., Košice, pp. 147, 2002, ISBN 80-7099-824-5.

[6] KAJIYA, J. T.: The Rendering Equation. In Computer Graphics, Proceedings of SIGGRAPH 86, Vol. 20, pp. 150, August 1986.

[7] GLASSNER, A. S.: An Introduction to Ray tracing, Morgan Kaufmann, pp. 327, California, 2000, ISBN 0122861604.

[8] WALD, I. – DIETRICH, A. – BENTHIN, C.: Applying Ray Tracing for Virtual Reality and Industrial Design, *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006*, pp. 177 – 185, Salt Lake City, USA, September 18 – 20, 2006.

[9] DIETRICH, A.: Massive Model Rendering Techniques, *IEEE Computer Graphics & Applications*, pp. 20 – 34, Special Issue Nov/Dec 2007.

[10] KERR, J. P. et al.: Photorealistic Volume Rendered Anatomical Atlases and Interactive Virtual Dissections of the Dissectable Human(TM)", *Engineering Animation*, Inc. 2321 North Loop Drive, Ames, IA 50010, [Cited 20.10.2009] Available from: <http://www.nlm.nih.gov/research/visible/vhp_conf/kerr/nlmpaper.htm>.

Received December 21, 2009, accepted April 23, 2010

BIOGRAPHIES

Liberios Vokorokos (prof., Ing., PhD.) was born on 17.11.1966 in Greece. In 1991 he graduated (MSc.) with honours at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He defended his PhD. in the field of programming device and systems in 2000; his thesis title was "Diagnosis of compound systems using the Data Flow applications". He was appointed professor for Computers Science and Informatics in 2005. Since 1995 he is working as an educationist at the Department of Computers and Informatics. His scientific research is focusing on parallel computers of the Data Flow type. In addition to this, he also investigates the questions related to the diagnostics of complex systems. Currently he is dean of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. His other professional interests include the membership on the Advisory Committee for Informatization at the faculty and Advisory Board for the Development and Informatization at Technical University of Košice.

Eva Danková (Ing.) was born on 6.2.1985. In 2009 she graduated at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. She is currently studying his PhD.

Norbert Ádám (Ing., PhD.) was born on 30.8.1980. In 1998 he graduated (MSc.) with distinction at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. He defended his PhD. in the field of Computers and computer systems in 2007; his thesis title was "Contribution to simulation of feed-forward neural networks on parallel computer architectures". Since 2006 he is working as a professor assistant at the Department of Computers and Informatics. Since 2008 he is the head of the Computer Architectures and Security Lab. at the Department of Computers and Informatics. His scientific research is focusing on the parallel computers architectures.