

OpenGL in Multi-User Web-Based Applications

K. Szostek, A. Piórkowski
Department of Geoinformatics and Applied Computer Science,
AGH University of Science and Technology, Cracow, Poland.
{szostek, pioro}@agh.edu.pl

Abstract — In this article construction and potential of OpenGL multi-user web-based application are presented. The most common technologies like: .NET ASP, Java and Mono were used with specific OpenGL libraries to visualize tree-dimensional medical data. The most important conclusion of this work is that server side applications can easily take advantage of fast GPU and produce efficient results of advanced computation just like the visualization.

Index Terms — server-side visualization, 3D, OpenGL

I. INTRODUCTION

Server-side solutions became more popular because of high computation power, large data storage space and security. Advanced computation of seismic field modeling [1] or Web-based DICOM viewing service [2] are examples of server-side applications which take advantage of this solution. Web-based DICOM viewer had been also described in [3]. Furthermore, graphic hardware (GPU) can increase their performance even hundred times [4]. Possibilities of DICOM data visualizations were presented in [5], [6] and [7]. Three-dimensional data, like results of computed tomography (CT)(Fig 1.) or 3D seismic survey can be visualized in real-time using OpenGL technology. These visualizations, parallel to conventional presentation methods, like presented in [2], can improve understanding complex 3D data.

II. VISUALIZATION TOOLS AND TECHNIQUE

A. OpenGL

OpenGL is a set of libraries and extensions that enables programmer to take advantage of graphic hardware [8]. These libraries were originally designed to provide users with high quality computer graphics in real-time, but today, thanks to the extensions, graphic hardware can be also used to solve complex mathematic or physic problems. OpenGL creates or *renders* visualization in *contexts*. This contexts must be initialized in system's graphic context prior to any OpenGL's function calls. What is more, there can not be more than one active render context. All of OpenGL's functions have to be called from the thread having active context, which means that only one thread can work with one context. Otherwise, context must be switched between threads, which is unfortunately slow and mostly discourage.

All the rendering occurs in *frame buffer*, which is usually the monitor display. OpenGL's extension allows to render directly into graphic's memory via *framebuffer object* (FBO), which is faster than frame buffer.

The Shader technology, an extension of OpenGL, may be useful to provide visualization with higher quality and to modularize server application. Advanced computation are also available through this technology. In this work Shader technology is omitted.

OpenGL libraries works with almost any programming language. For most of them there are OpenGL wrappers, like Tao Framework [9] or OpenTK [10] for MS .NET or Mono, Java 3D, Java OpenGL (JOGL) [11] or Lightweight Java Game Library (LWJGL) [12] for Sun Java. These libraries are usually very similar in use and code may be easily and fast reused.

To construct OpenGL server application Tao Framework, JOGL and LWJGL were used as they are very similar. OpenTK, based on Tao Framework, and Java 3D were omitted because they provide different approach to OpenGL application construction.

B. DirectX and XNA

DirectX and XNA are sets of tools and libraries from Microsoft. These tools are omitted because of their commercial character and limit of language and platform options.

C. Volume render

Volume render is a method of visualization of three-dimensional data. OpenGL with extensions includes tools to compose 3D texture, which can be sampled by planes parallel to view to display 3D data (Fig 2.). Depending on *transfer function* these planes have applied opacity to present chosen part of data.

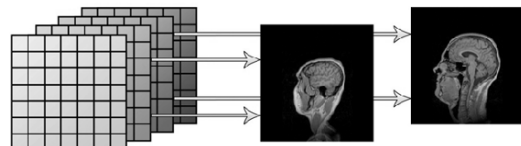


Fig 1. Example of 3D data.

This method was chosen because of its popularity, simplicity and possibility of Shader technology application. Shader technology may be used to define custom transfer function and to improve quality of visualization. [5] [6] Different approach of visualization was presented in [7].

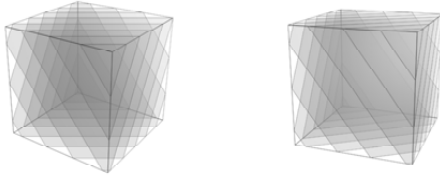


Fig 2. Box of 3D data sampled by planes in volume render.

III. WEB-BASED VISUALIZATION APPLICATION

A. Concept

General idea of OpenGL multi-user web-based application is to provide high quality visualization of 3D data to thin client architecture via web browser. The server application has to be able to produce series of visualization for multiple clients requests at one time. Method of visualization may vary depending on type of 3D data.

B. Hardware and software requirements

In order to take full advantage of OpenGL application, graphic hardware must support 3D texturing and Shader technology. Its amount of memory should be above average, because 3D data may be very large. Server's operating system must run in graphic mode as the graphic context is required.

C. Construction of application

The first request received by server application initializes OpenGL rendering context and FBO. Then 3D data are loaded from files (or one prepared file) and 3D texture is created. The visualization is ready to be rendered and the application can now receive more requests.

Then the process enters critical section. This prevents other request from calling OpenGL's function or changing active context. The context is set as active and visualization's parameters sent by client, e.g. eye-view position, are applied. The chosen visualization is rendered into FBO and copied into byte array. Rendering context become inactive. Context switching is forced by server technology, which may start new thread for new client request. The process exits critical section. Byte array is then converted into image, like BMP, JPEG or PNG and is sent to client. This sequence is presented on figure Fig 4. Analogical and more general model of communication was described in [13].

D. The applications

Application's interface is a web page, written in HTML and JavaScript (Fig 3.). It provides user with control buttons and visualization.

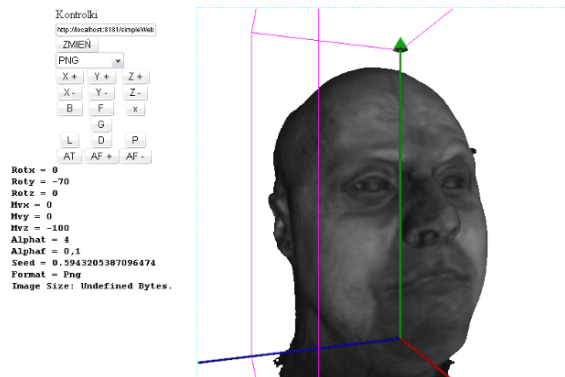


Fig 3. Web page interface with example visualization of CT images

User sends request each time he clicks any of the control button. Results of the requests are displayed next to this controls.

IV. PRELIMINARY TESTS

A. Operating systems

The application was preliminarily tested on two operating systems: MS Windows XP (SP3), and Linux (Fedora Core 10, kernel: 2.6.27.21-170.2.56.fc10.i686). There were few changes made on systems to allow running OpenGL server application and operate on large amount of data.

Windows Server 2008 and its security solution causes problems to OpenGL server application. This OS was omitted in this paper, but will be examined in future.

B. Software environment

Internet Information Services (IIS), version 5.1 for Windows XP and version 7.0 for Windows Server 2008 to run ASP .NET server application and Apache Tomcat, version 6.0.18 for Sun Java SE (1.6.0_14) were installed.

The application was written in ASP and C# for .NET Framework, version 2.0 and Mono, version 2.4; and in Java language. The Tao Framework, version 2.1.0, JOGL, version 1.1.1a and LWJGL, version 2.1.0 OpenGL's wrappers were used.

In Mono and Java libraries few issues were found and reported as bug.

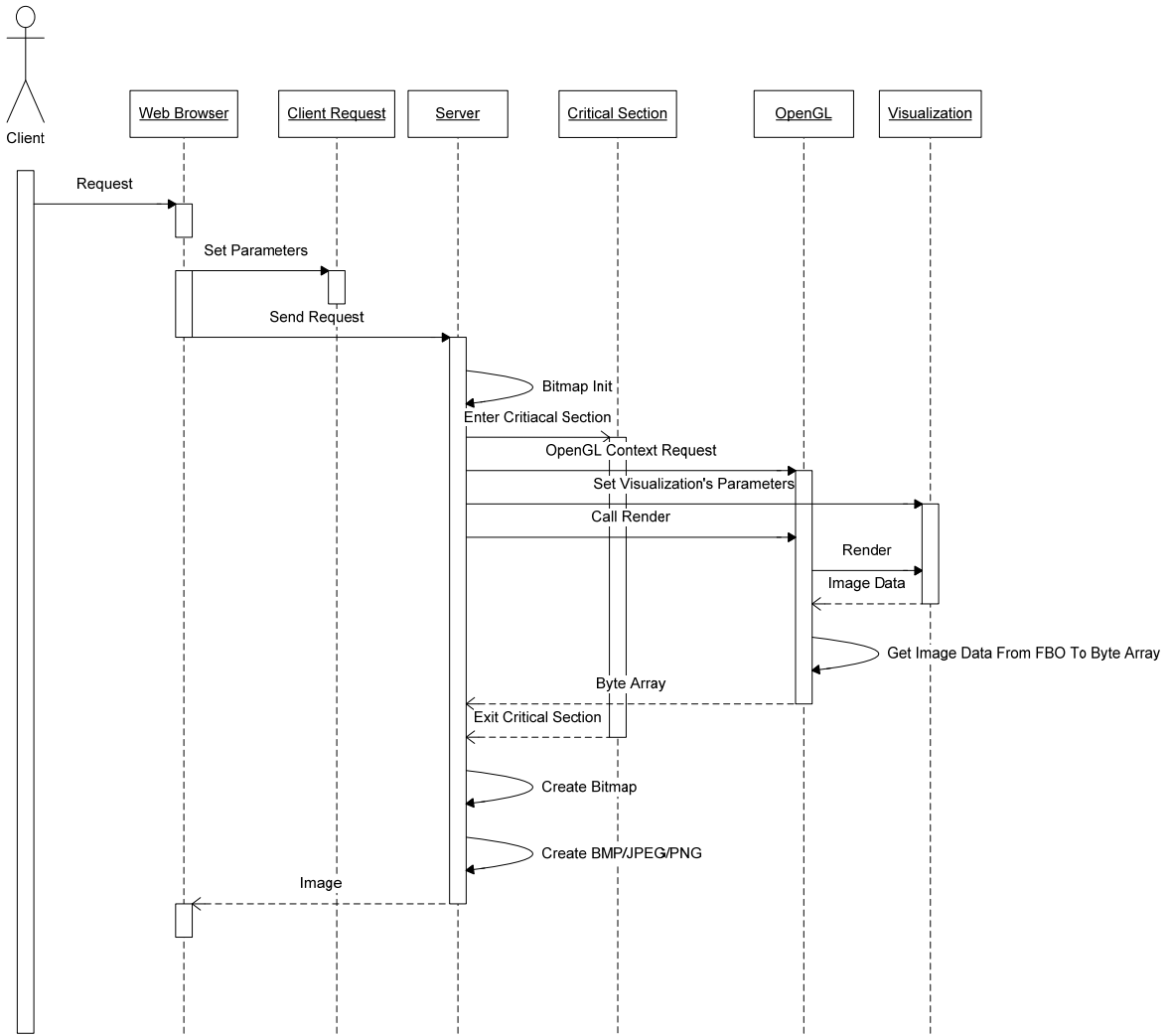


Fig 4. Sequence diagram of request handling by constructed web-based applications.

C. Hardware environment

D. The tests

The tests were performed using two computers. The server machine was single computer (laptop) which has following parameters:

- Intel Pentium III Xeon, 2500 MHz,
- 3072 MB RAM,
- NVIDIA Quadro FX 1600M (512 MB).

The second computer, known as thin client, was standard PC which has following parameters:

- Intel Pentium 4, 1600 MHz,
- 640 MB RAM.

It has MS Windows XP Professional SP3 installed.

Created applications were tested using JMeter 2.3.2 installed on the client side. Minimal response for single user and throughput for 30 users were examined. Tests were done for resolutions 256x256 and 512x512, for three common image file formats (BMP, JPEG and PNG). The results are presented in Table I and figures Fig 5. and Fig 6.

V. DISCUSSION

The shortest time of response was reached in Java and LWJGL under control of Linux Fedora Core 10. This solution seems to be the most promising for future work, which may be surprising, because LWJGL is not just an OpenGL's wrapper. It has some additional classes and mechanisms. As we can read on the official web site of the project: "The whole point of LWJGL was to bring the speed of

Java rendering into the 21st century” [12]. Comparing LWJGL and JOGL as two Java extension LWJGL was, as claimed, fastest.

Tao Framework and MS .NET were the fastest and reached the highest throughputs in generating PNG images. Depending on required quality of images PNG or JPEG compression should be chosen.

The worst results were achieved in Mono under Linux system. Mono technology is still young but fast developed environment. Although it is not free of bugs, which may be one of the reason of the poor results.

The creation of uncompressed image format (BMP), as expected, produce the worst results for all of the tested technologies, besides Java. However, if quality is the most important factor, it is suggested to choose PNG format, which results with better times and throughputs.

Considering the results it has to be taken into account, that presented times and throughputs are not only OpenGL image generation, but whole server response. Therefore, perhaps some sophisticated software environment enhancement could be made.

TABLE I
TIME PERFORMANCE AND THROUGHPUT OF WEB-BASED APPLICATIONS

				Win XP			Linux Fedora		
				Tao	JOGL	LWJGL	Tao	JOGL	LWJGL
				Response time [ms]		256x256		512x512	
Response time [ms]	256x256	JPG	35	32	30	45	24	23	
		BMP	77	73	81	122	38	36	
		PNG	50	74	70	92	59	60	
	512x512	JPG	84	84	79	109	69	66	
		BMP	232	269	268	262	130	129	
		PNG	97	167	165	142	134	131	
Throughput [req./sec]	256x256	JPG	31,3	30,4	31,4	15,4	27,2	29,7	
		BMP	14,4	14,2	14,4	8,3	28,1	30	
		PNG	30,7	26,3	26,7	13,7	24,9	27,4	
	512x512	JPG	15,7	15,4	15,5	15,8	14,3	15,2	
		BMP	4,2	4	3,9	4,2	13	13	
		PNG	15,6	9,6	9,2	10,8	10,8	10,6	

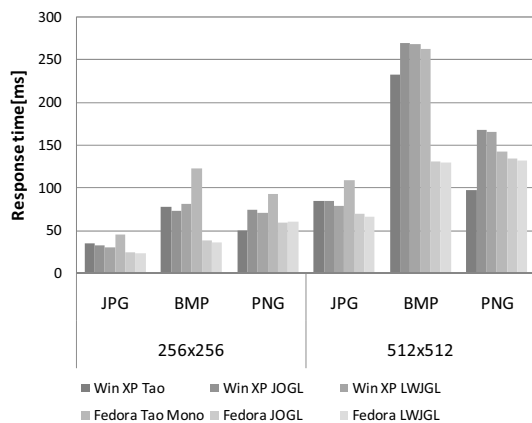


Fig 5. Time performance of web-based applications

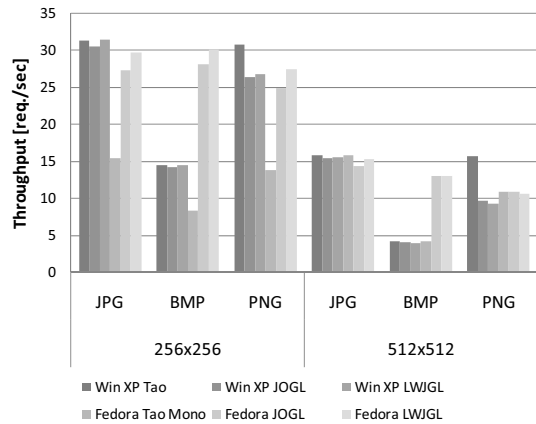


Fig 6. Throughput of web-based applications

VI. FUTURE WORK AND OPTIONS

This work is an introduction to more complex web-based application. In future work OpenGL web-based application will be tested on IBM HS21 Blade Server. The Shaders Technology will be introduced to improve speed and visualization quality [4],[5]. The visualization method will be adapted to different 3D data, like seismic surveys’ data. What is more, OpenGL will be used to improve quality of web-based viewing service for DICOM images.

VII. SUMMARY

This work proved that server-side applications can take advantage of high performance of graphic hardware to produce high quality visualizations for thin client architecture. These visualizations may be important for students and professors to learn and explain more convenient presentation’s method of 3D data, like CT or MRI scans.

ACKNOWLEDGMENT

This work was financed by the AGH - University of Science and Technology, Faculty of Geology, Geophysics and Environmental Protection as a part of statutory project number 11.11.140.561.

REFERENCES

- [1] A. Kowal, A. Piórkowski, T. Danek, A. Pięta, “Analysis of selected component technologies efficiency for parallel and distributed seismic wave field modeling,” in Innovations and Advances in Computer Sciences and Engineering, Springer, 2010.
- [2] A. Piorkowski, L. Jajesnica, K. Szostek, “Creating 3D Web-Based Viewing Services for DICOM Images”, in Computer Networks, 16th Conference, CN 2009, Wisla, Poland, June 16-20, 2009, Communications in Computer and Information Science, Springer Berlin.
- [3] J. Fernandez-Bayó, O. Barbero, C. Rubies, M. Sentís, L. Donoso, “A DICOM Web Server and a DICOM Java Viewer,” in Distributing Medical Images with Internet Technologies, RadioGraphics, 2000.
- [4] T. Danek, “Seismic wave field modeling with graphics processing units,” in Lecture Notes in Computer Science, vol. 5545, 2009.

- [5] O. Kuttera, R. Shamsb, N. Navaba, "Visualization and GPU-accelerated simulation of medical ultrasound from CT images," in Computer methods and programs in biomedicine 94, Elsevier, 2009.
- [6] J. Krüger and R. Westermann, Acceleration Techniques for GPU-based Volume Rendering, Computer Graphics and Visualization Group, Technical University Munich, 2003.
- [7] P.N. Ancuta, "3D Object Modeling and Visualization Software for Surgery Preoperative Plan," in 6th Workshop on European Scientific and Industrial Collaboration on promoting Advanced Technologies in Manufacturing, WESIC'08, 2008.
- [8] The OpenGL Graphics System: A Specification (Version 3.1 - March 24, 2009) [a:] <http://www.opengl.org>
- [9] The Tao Framework documentation, [a:] <http://www.taoframework.com/about>
- [10] The Open Toolkit Manual, [a:] <http://www.opentk.com/doc>
- [11] J.X. Chen and E.J. Wegman, Foundations of 3D Graphics Programming Using JOGL and Java3D, Springer Verlag, New York, 2006.
- [12] Home of the Lightweight Java Game Library. [a:] www.lwjgl.org/
- [13] C. W. Arnold, A. A. T. Bui, C. Morioka, S. El-Saden, and H. Kangaroo, "A Prototype Web-based Reporting System for Onsite-Offsite Clinician Communication," in Radiographics July 2007 27:1201-1211