Professional paper
UDK: 655:004.738.52:004.773

Evaluating Web browser graphics rendering system performance by using dynamically generated SVG

Darko Avramović[1], Željko Zeljković[1], Neda Milić[1], Gojko Vladić[1]
University of Novi Sad, Faculty of Technical Sciences, Department of Graphic Engineering and Design, Serbia

Corresponding author: Darko Avramović
email: adarko@uns.ac.rs

Abstract:
This paper evaluates common web browsers graphics rendering system performance by using SVG on Windows platform. Performance was measured using two extreme case applications: application which generates simple SVG objects (called “simple application”) and application which generates complex SVG objects (called “complex application”) based on random data often used for science purposes. Both simple and complex applications were hosted on local web server. Performance tests were performed using dedicated chamber environment.

Key words: SVG, Web browser, performance, graphics

Introduction

World Wide Web first appeared about two decades ago as a medium used to render plain HTML documents that were stored on a specially configured PC machines called web servers running same named software. The client-server transfers were performed using Hyper Text Transfer Protocol (HTTP) (Apte et al., 2002).

About twenty years later World Wide Web evolved to serve as a global data sharing space for large number of users. There are now about 1.9 billion users worldwide (Miniwattz marketing group [Online], 2012) and it is supporting technologies to advance rapidly. The ubiquity of the client and easy accessibility to other data sources make the Web a very attractive means of information sharing (Kim et al., 2012).

Scalable Vector Graphics (SVG) is a language for describing two-dimensional graphics in XML (W3C, 2012). The SVG format is a new XML grammar for defining scalable vector-based 2D graphics for the Web and other applications and usable as an XML Namespace (Peng, 2000). SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered ob-

First received: 30.03.2012.
Accepted: 30.05.2012.

jects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects (W3C, 2012).

SVG documents inside are pure XML and can be created by hand or can be easily generated by some kind of server- or client side technology. Most commonly used graphic formats today used in web are pixel based (bitmap) and during zoom operations their picture quality drops. That is absent in case of SVG.

Related work and motivation

Similar research was performed by Moreno and de Olivera, 2008 who measured CPU and memory utilization and response time of a web application. They tested the application using simple and complex SVG graphics. The tests were performed in 2008 but its shortage is that very old browsers were used for testing. Internet Explorer 6 was released in 2001, Firefox 1.5 was released in 2005 and Opera 9.02 was released in 2006. (Release Histories for all Major Browsers [Online], 2012) That is quite a big age difference. The tests performed in this paper were performed using latest web browsers versions.

Also, there is a trend of SVG usage in modern printing technologies – Web-To-Print also known as Web2Print or remote publishing is a commercial prepress process that bridges the gap between digital content online and commercial print production. This process allows a print house, a client, and possibly a graphic designer to create, edit, and approve computer-based online templates during the prepress phase. This process increasingly calls for a Portable Document Format (PDF) workflow environment with output provided by digital printing; although there is certainly no requirement that fulfillment be accomplished using digital production equipment; Web-to-print is also used today by printers with both offset and digital production facilities. Web-To-Print software solutions often use SVG technology and thereby allow customer to make changes or even create printing job from scratch. These facts served as motivation for the tests performed in this paper.

Experiment and experimental environment

Experiments were taken using latest web browsers: Google Chrome 18.0, Mozilla Firefox 12.0, Opera 11.64 and Apple Safari 5.1.7. All browsers operated with caching option disabled and in private/incognito mode where available. Table 1 presents experimental environment.

Table 1. Experimental environment

Hardware used	
CPU	Intel Pentium 2.5GHz
Memory	3GB DDR2 1066MHz
OS	Microsoft Windows XP SP3 5.1
GFX	nVidia GTX560Ti 1024MB GDDR5

For the purpose of this experiment two applications were created. Both applications prevented browser caching using meta data. First (called “simple”) was named circle and it had a task to create simple red circle using SVG, as presented in Figure 1.

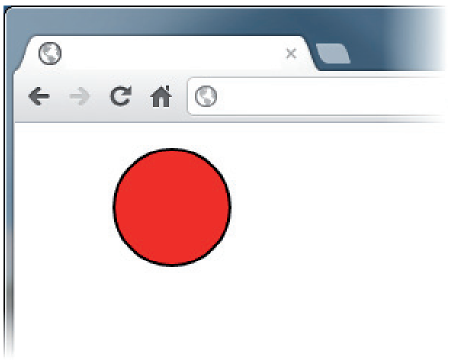

Figure 1. Circle application sample

Other application (called “complex”) got name plot and represents application which generates random data plots, as presented in Figure 2.

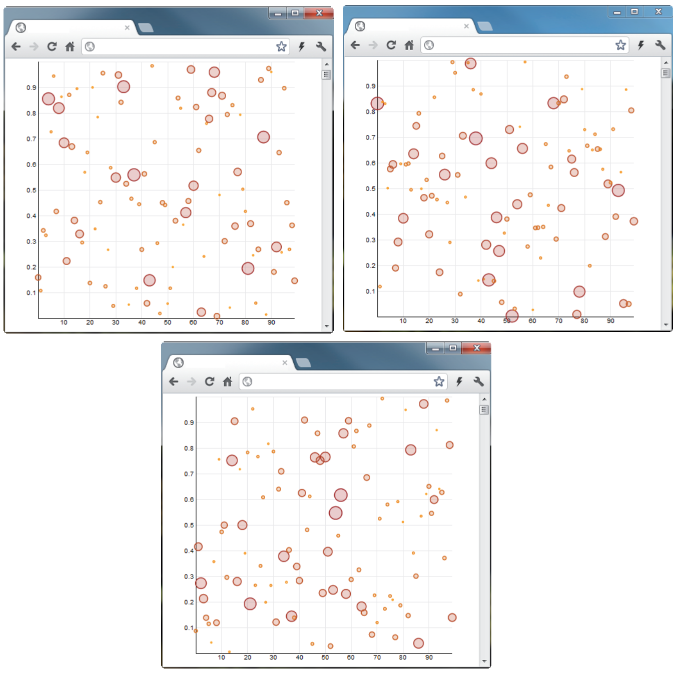

Figure 2. Plot application samples

There were four tasks (cases) where each application generated different number of objects (circles or plots). These tasks are presented in Table 2:

Table 2. Four application tasks

Task	No of objects generated
1	50
2	250
3	500
4	750

Each web browser has been tested to perform all four tasks. For each task 30 tests (samples) have been performed. That means that each browser has been tested 120 times per application, 240 times for both applications. Time to perform each test has been measured using specially made chamber application. The chamber application initiates test application loading and measures the time until test application page load completion, which is described in Figure 3.

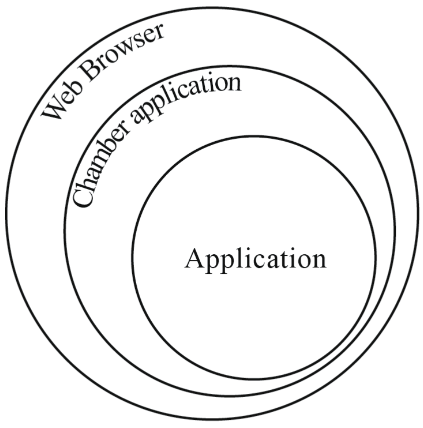

Figure 3. Testing method

At the end resources bottleneck analysis shall be performed to see how good or bad each browser resource management is. CPU bottlenecks, technology bottlenecks and page file usage shall be monitored. All tests have been performed on the operating system running only 4 mentioned web browsers and no antivirus, firewall or any other non-system services running or installed.

Results and Discussion

Simple application results and bottleneck analysis

Results gathered from simple application (Figure 1) tests shall be reviewed in the first place. Figure 4 shows total test completion time for each web browser. It is visible that all browsers except Opera achieved equally matched times. Figure 5 shows that Google Chrome and Apple Safari present almost the same curve slope, while Mozilla Firefox gave smallest curve slope. Opera tests on the other hand returned the worst results as presented in Figure 4. These results tell that Opera graphics rendering system is least optimized for simple SVG requests and web developers should have that fact in mind during application development. Apple Safari on the other hand returned best results of all. Figures 6 to 9 present actual test completion times for each web browser for all 30 samples made. Opera case presents relatively balanced curves but long completion time relative to other browsers.

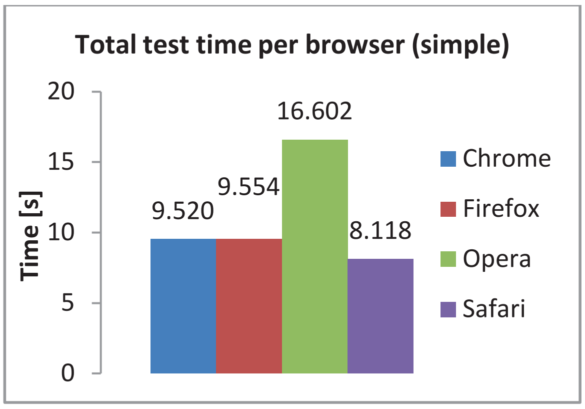

Figure 4. Total test time per browser (simple application).

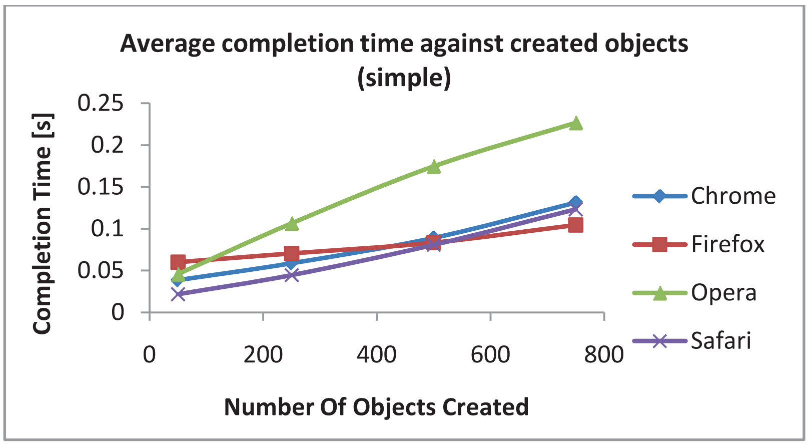

Figure 5. Average test completion time against number of created objects for each web browser (simple application)

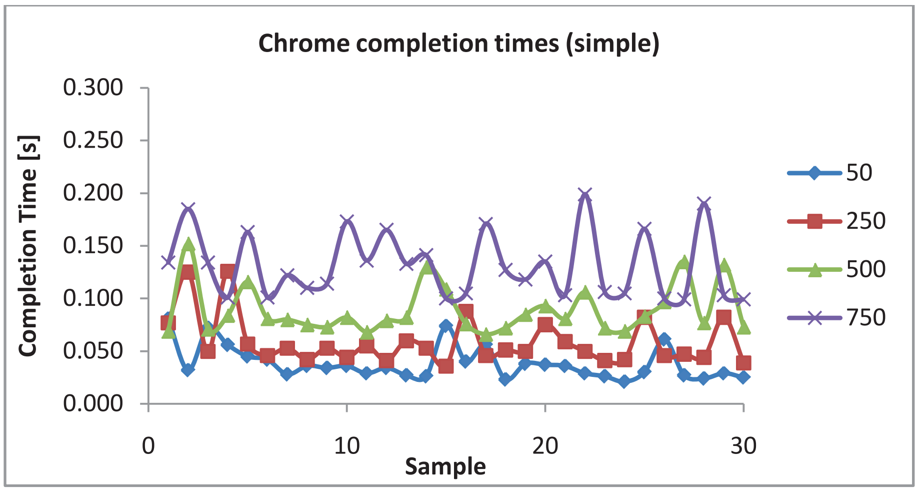

Figure 6. Google Chrome test completion times (simple application)

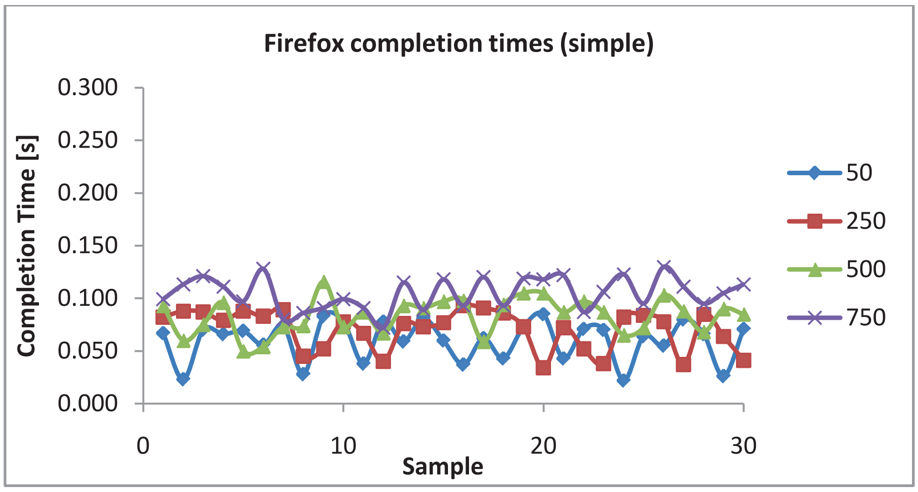

Figure 7. Mozilla Firefox test completion times (simple application)

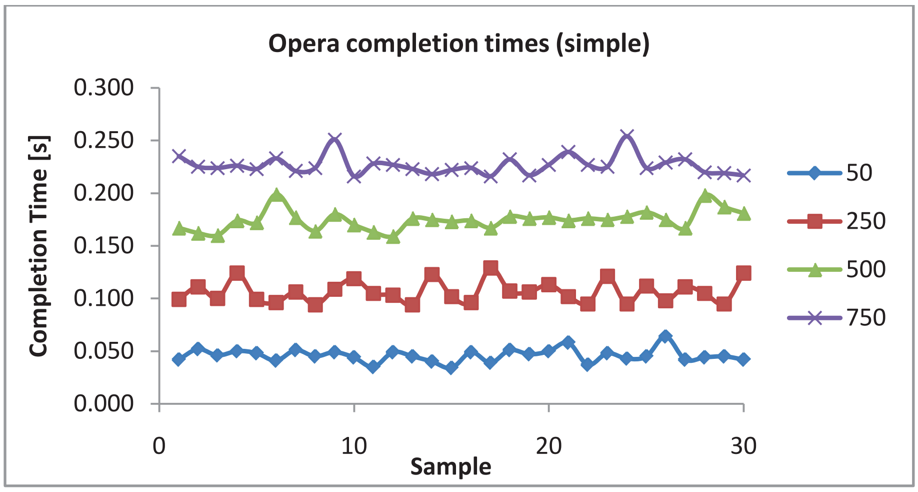

Figure 8. Opera test completion times (simple application)

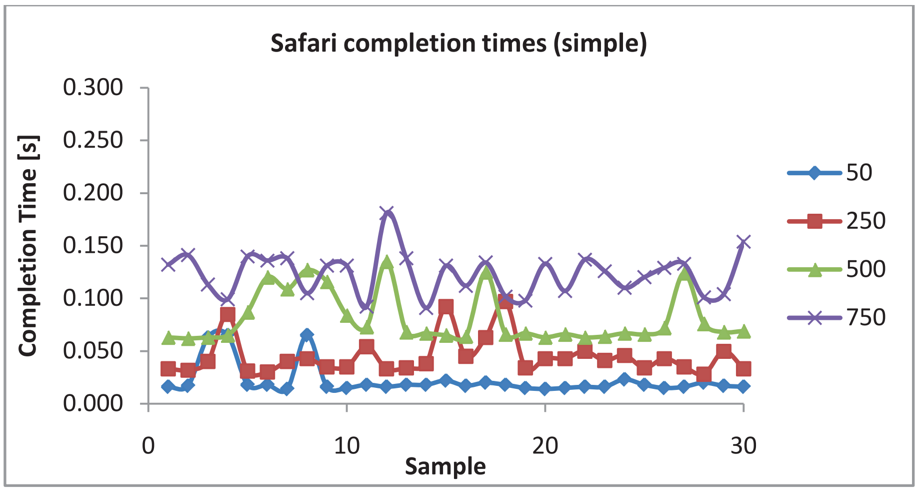

Figure 9. Safari test completion times (simple application)

Complex Application results and bottle-neck analysis

In case of complex application (Figure 2) total test times gathered are similar to simple application case (Figure 4) which is presented in Figure 10. Safari web browser achieved slightly better results in comparison to simple application results. Figure 11 shows slightly different information than Figure 5 (average completion times for simple application). In this case Apple Safari took the lead in category of speed. In Figure 5 Google Chrome and Apple Safari curve slopes were closely matched but in this case Chrome and Firefox are now way behind Safari in terms of speed. Figures 12 to 15 present actual test completion times for each web browser for all 30 samples made. These graphs are slightly different then ones connected to simple application. Due to time scale balance is present in a larger scale, but in this case there is a difference in browser behavior. Apple Safari achieved best results and curve balance with no process lockups. Mozilla Firefox achieved almost equally good curve balance but a bit higher test completion time. Google Chrome achieved test completion times as good as Mozilla Firefox but curve oscillations were noticeable (frequent process lockups were noticed).

Opera, at the end, achieved lowest score in terms of test completion times but showed stability with no huge curve oscillations. In terms of test completion times,

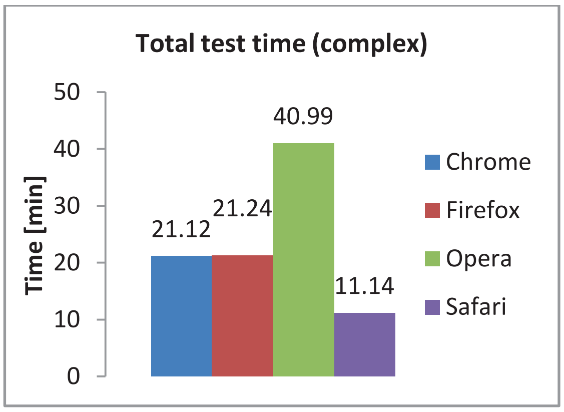

Figure 10. Total test time per browser (complex application)

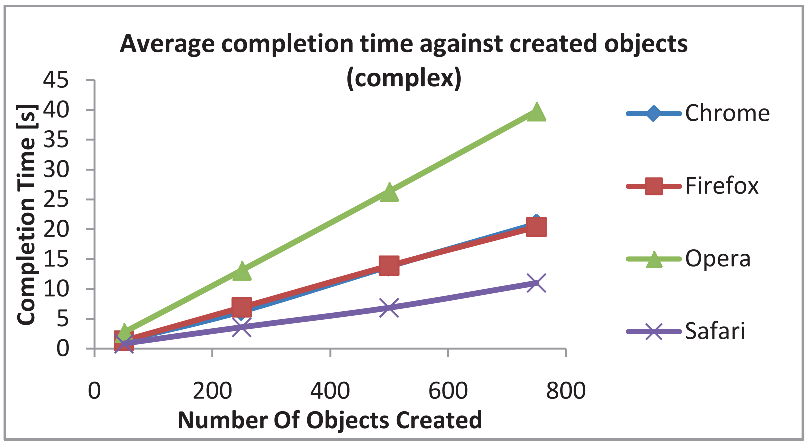

Figure 11. Average test completion time against number of created objects for each web browser (complex application)

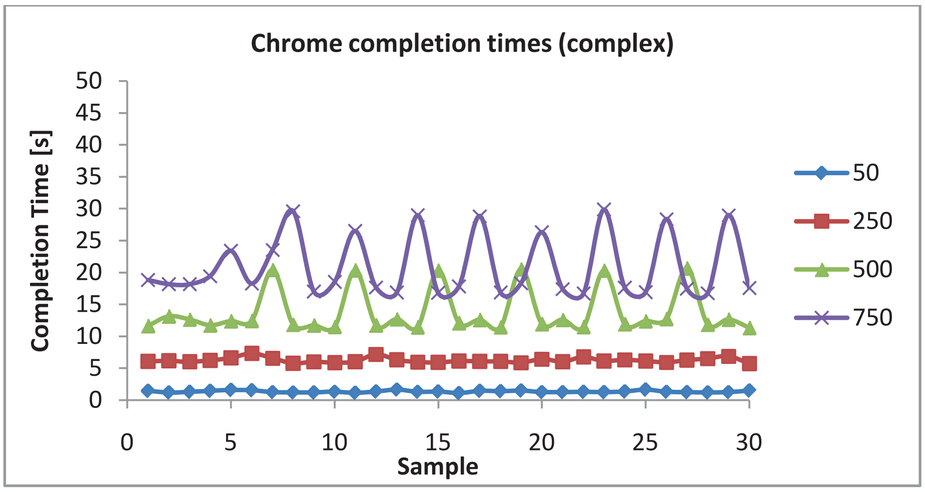

Figure 12. Google Chrome test completion times (complex application)

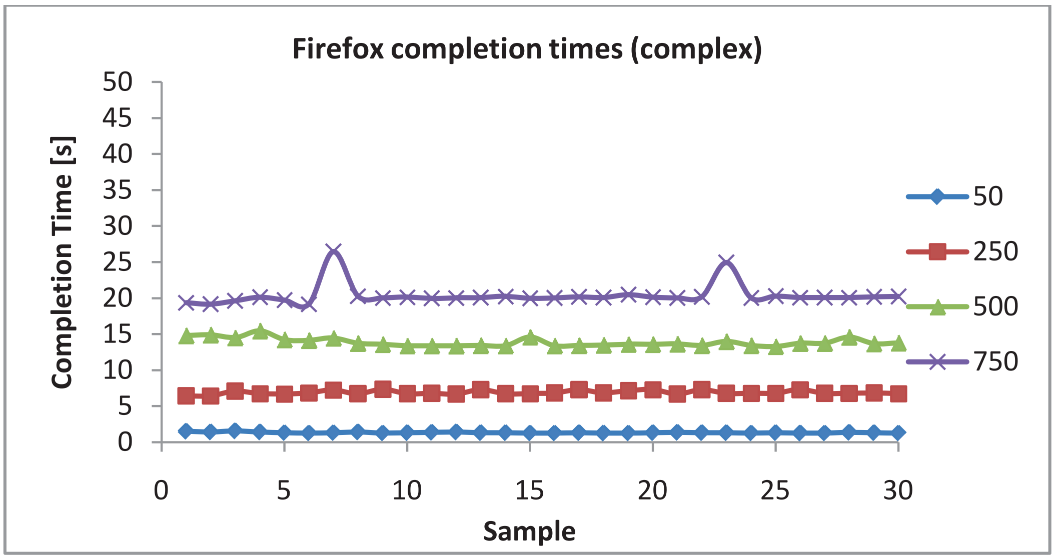

Figure 13. Firefox test completion times (complex application)

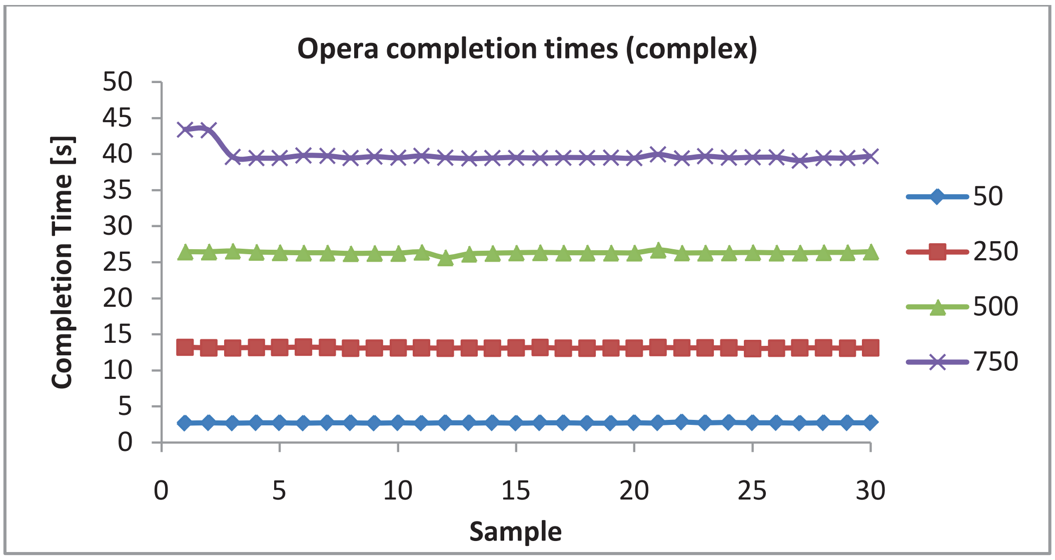

Figure 14. Opera test completion times (complex application)

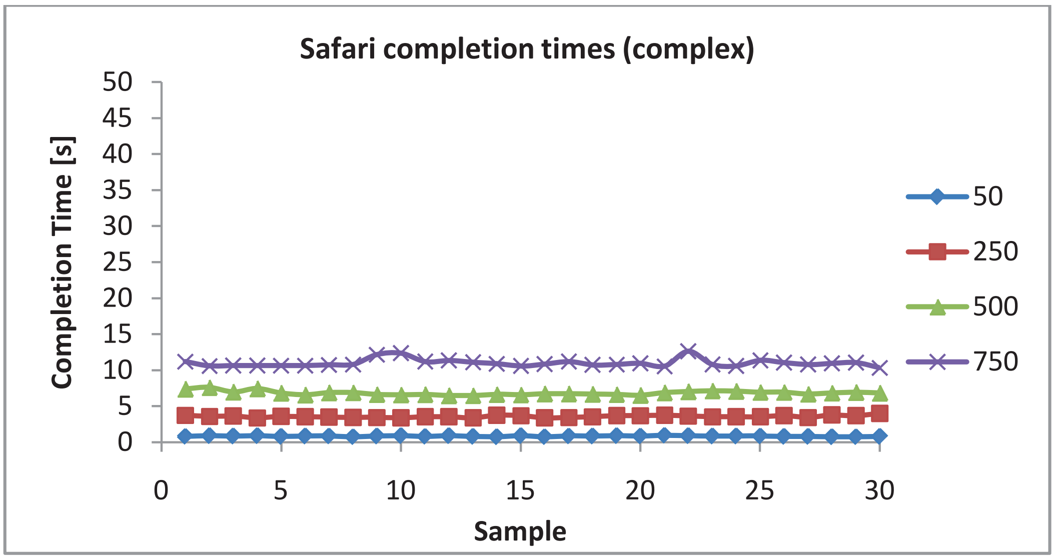

Figure 15. Safari test completion times (complex application)

when Opera and Mozilla Firefox are compared, Moreno and de Olivera concluded the same thing: Opera takes longer to generate SVG. In terms of CPU utilization the situation is opposite, Opera takes more of CPU than Mozilla Firefox. This conclusion is based on CPU bottleneck analysis.

Table 3. CPU bottleneck appearance against number of objects created per test

No. of objects	Chrome	Firefox	Opera	Safari
50	0	0	1	0
250	1	1	1	1
500	1	1	1	1
750	1	1	1	1

Table 3 presents CPU bottlenecks detected during tests against number of objects generated per test. It is visible that all four web browsers cause CPU bottlenecks for 250 and more objects generated. In case of 50 objects task only Opera generates short lasting CPU bottleneck which indicates that the browser spends more CPU resources than others. CPU utilization levels were not measured during this research.

Memory bottlenecks were not detected for complex nor simple applications.

Table 4 presents page file values. These are operating system level values gathered during each browser test. In case of page file usage Opera web browser took this minor victory over other browsers. Apple Safari took second place and the other two browsers divide third place (Figure 16)

Table 4. Page file values [Gigabytes]

No. of objects	Chrome	Firefox	Opera	Safari
50	0.5	0.7	0.7	0.7
250	0.9	0.9	0.8	1
500	1.6	1.4	0.6	1
750	2	2	1	1.5

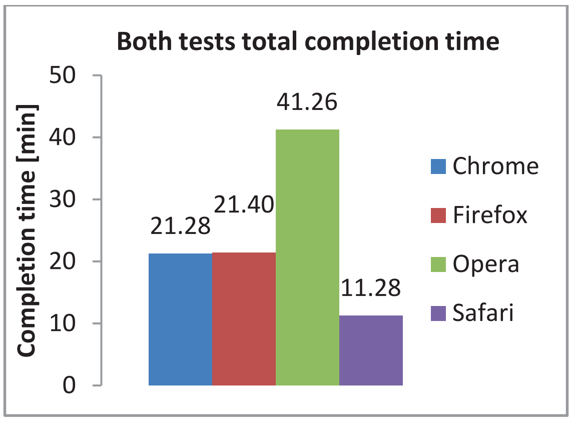

Figure 16. Average page file value during tests per browser

At the end the best browser shall be elected out of these four. In terms of speed and performance (Figure 17) Apple Safari shown best optimization level for SVG tasks with only 11.28 minutes required to complete both tests with total of 93000 objects created (both simple and complex). Chrome and Firefox are equally matched at the end with 21.28 minutes and 21.40 minutes needed for mentioned number of objects generated. Opera has poorest performance with little over 40 minutes needed for both tests completion.

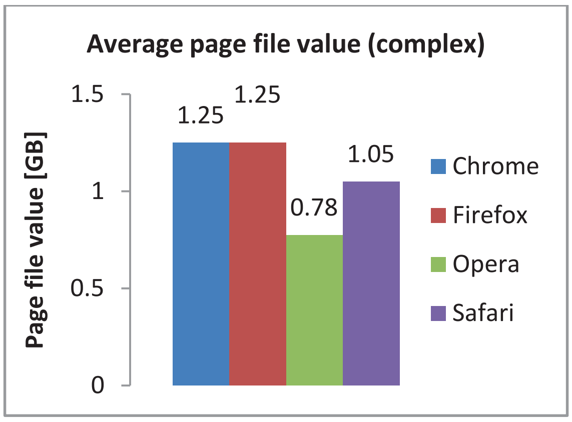

Figure 17 Both tests total completion times per browser (both applications)

Conclusion

Most important thing users seek for today is speed. No user likes to wait for its browser to perform a particular operation. Software developers and web developers are pressed by this problem as well. Every developer has to optimize its application to behave equally good in any environment (environment assumes operating system or in this case browser). Also each software solution should perform same tasks equally fast. This research shown that the situation is good for some web browsers and not that bright for others.

In case of graphics rendering system performance Apple Safari is the most reliable solution with best results achieved during tests. If a person needs speed and comfort during operations with SVG, Apple Safari web browser presented its self as the best choice in terms of speed, stability and resource consumption. Also one should avoid Opera browser because of poor performance shown during tests.

On the other hand, software developers may look things from a little different perspective. Namely, their goal is to make application which shall behave equally good in every browser but this research shows that it is not simple task. Developers should pay more attention to the application optimization for these problematic browsers like Opera and try to achieve best possible performance out of, evidently, not enough optimized engine.

Results presented in this paper are also important for Web-To-Print software developers because these systems rely heavily on browsers graphics rendering system to complete complex tasks presented before the system.

References

1. Apte, V, Hansen, T, Reeser, P (2003) Performance comparison of dynamic web platforms, Computer Communications 26 888–898
2. Impressive Webs, Release Histories for all Major Browsers [Online] Available from: http://www.impressivewebs.com/release-history-major-browsers/ [Accessed 14th May 2012]
3. Kim, E.N., et al. (2012) Web-based (HTML5) interactive graphics for fusion research and collaboration, Fusion Eng. Des. http://dx.doi.org/10.1016/j.fusengdes.2012.03.041 [Article in press]
4. Miniwatts Marketing Group, World Internet Usage Statistics News and World Population Stats [Online] Available from: http://www.internetworldstats.com/stats.htm [Accessed 5th May 2012].
5. Moreno, E. D, de Oliveira, J. I. F., (2009) Architectural impact of the SVG-based graphical components in web applications, Computer Standards & Interfaces 31 1150–1157
6. Peng, C, (2000) SCALABLE VECTOR GRAPHICS (SVG), Tik-111.590 Research Seminar on Interactive Digital Media
7. World Wide Web Consortium (W3C), SVG 1.1 (Second Edition) [Online] Available from: http://www.w3.org/TR/SVG/intro.html [Accessed 12th April 2012].