A Comparison of Programming Platforms for Interactive Visualization in Web Browser Based Applications

Tim Lammarsch, Wolfgang Aigner, Alessio Bertone, Silvia Miksch, Thomas Turic
Department of Information and Knowledge Engineering, Danube University Krems
www.donau-uni.ac.at/ike

Johannes Gärtner
XIMES GmbH
www.ximes.com

Abstract

Recently, web browser based applications have become very popular in many domains. However, the specific requirements of interactive Information Visualization (InfoVis) applications in terms of graphics performance and interactivity have not yet been investigated systematically in this context. In order to assess browser-based application platforms, we provide a systematic comparison of server-based rendering, Java applets, Flash, and Silverlight from several points of view. We aim to aid InfoVis developers in choosing the appropriate technology for their needs.

1 Introduction

Running the client side of distributed applications inside a web browser brings a great deal of platform independence and inherent connectibility. It also lowers the barrier for its introduction. Among others, these facts have led to an increasing popularity of browser-based applications. Most of them use classical user interfaces (UIs) consisting of controls, like buttons, textfields, and scrollbars. In InfoVis, there are several powerful and versatile applications that are well known among experts, but designed to run natively in operating systems (OSs) only.

The goal of this paper is the comparison of a selection of programming platforms for developing browser-based InfoVis applications. Firstly, we identify the most important requirements needed for InfoVis with a focus on interactive visualizations. Secondly, we evaluate selected platforms by (1) implementing and running an example technique, (2) performing benchmark tests regarding drawing speed, and (3) making a feature assessment. Our aim is to provide a thorough assessment of currently available programming platforms that enables InfoVis developers to choose the one most suitable according to their needs.

2 Related Work

Much information about browser-based programming technologies has been published on conventional websites or blogs. A particular example is the article by Garrett [4] who describes the combined use of asynchronous loading of web pages using ECMAScript[1] and introduces the term "Ajax" for this. In a comparison of Java and Flash, Burnette [2] claims that stability and the size of the client plugin are the main reasons for Flash having become more important for browser-based applications than Java. A feature comparison of Flash and Ajax is done by Pasztory [7] discussing applications at which Ajax is superior. A feature comparison of Flash and Silverlight is done by Ezell [3] who points out that Silverlight has solutions for all the problems of Flash, but does not mention the OSs issues. A speed benchmark of algorithms in different implementations of ECMAScript, Java and Flash is done by Lyda [6]. The results show that while ECMAScript implementations are similar in speed among browsers, Java is much faster. Flash version 9 is somewhere in between.

Wohlfart [9] points out that scientific papers which compare tools and techniques are focused either on the creation of taxonomies for describing InfoVis tools or the empirical testing of them. We will focus on the latter in order to give an overview of what is necessary in an interactive InfoVis application. Kobsa [5] compares Eureka, InfoZoom, and Spotfire. Wohlfart [9] evaluates Tableau, Spotfire DXP, Xmdv Tool, ILOG Discovery, and CViz. Wohlfart also describes the development and visualization tools OpenDX, AVS/Express, GeoVista Studio, IRIS Explorer, DeVise, RefViz, and IN-SPIRE. However, none of the systems or tools is a browser-based application and the graphics libraries, while being adequate for developing this kind of applications, are not tested under this point of view. In order to fill this gap, we present a systematic review of the use of browser-based application platforms.

3 Technologies and Comparison Design

The main distinction of browser-based applications is the difference between code running on the client and on the server. We analyzed:

[1] Throughout this paper, we will use the term *ECMAScript* for what is usually called JavaScript (the name of its implementation in Netscape, Mozilla, and FireFox) because other implementations with different names work the same way.

Server-Based Rendering[2]**:** Many technologies are available and our benchmark includes tests for server-based .NET and Java.

Java Applets[2]**:** Developed by Sun, executed using the Java Virtual Machine (VM) and written in the Java language; many free libraries are available for Java and the VM is widespread.

Flash[2]**:** Developed by Macromedia and now owned by Adobe, executed using the Flash browser plugin and written in a language called ActionScript; the plugin is very widespread.

Silverlight[2]**:** Developed by Microsoft, executed using the Silverlight plugin which is a smaller version of the .NET framework and written in various languages supported by .NET, particularly C# and Visual Basic; the .NET framework is very powerful and has spread in the commercial area very quickly.

We analyzed the listed technologies based on the following criteria. We did not define priorities among them because the priority depends strongly on the tasks of developers and users.

Run Capability: Multi-platform development means that no definite assumptions about the client system can be made. Therefore, the visualization has to be developed so that it can be shown on any supported client system. We describe the software requirements on the clients for each platform and our opinion on how easily these needs can be fulfilled.

Rendering Graphics Primitives: It should be possible to render any type of graphics primitive. This is the main reason for (X)HTML approaches requiring server-side rendering. Other platforms have this ability, but there are different design philosophies.

Animation: Animation enables the use of time as visual variable.[3] An important improvement towards timeline-based animation is user-controlled animation with frame selection.

Interactivity: If possible, user interaction should happen in real-time. To be sensed as an immediate response, the time between cause and effect of a user action should not exceed 50–150 ms [8].

Communication with a Server: The basic idea of a web-based application is that a webserver provides the sites and the user data is stored on the server-side. To make this work, the data has to be transferred to the client. Many advantages of interactivity also require that new data can be loaded at runtime.

Stability: Stability is an important factor for all applications. It is especially worth mentioning as the tripartite construct of any web browser, any plugin and code increases the possibility of instability, even more so if the used technology is very new.

Guaranteed Future: The dependency on browsers and in most cases plugins also leads to the importance of checking if there is a future for the technology.

Development Experiences: While developing is still possible with a text editor, integrated development environments (IDEs) can improve the process significantly. The availability of libraries also helps much in the development process. A good documentation and help on the web are also important for software development. While we can list what is available for each platform, our evaluation of the development experience has to remain subjective.

The analyzing process consisted of three parts. (1) We developed an interactive visualization using each platform. This application shows an interactive mass-spring-based model of a social network (Figure 1) similar to the GraphView demo by Heer from the Prefuse gallery.[4] We implemented the spring-based model in a .NET desktop application using the GDI+ renderer as a reference, in Java, Flash, and Silverlight. Here, we excluded server-based rendering because the application would basically be written in ECMAScript and executed on the client, giving no new insights about server-based rendering. Table 1 shows how many nodes could be rendered with reasonable speed.

(2) We wrote a benchmarking application, which consists of three parts, shown in Figure 2. (2.1) Polygons are drawn with 3 to 66 vertizes with line widths of 1 to 16 for a total of 1024 polygons. (2.2) Filled rectangles are drawn with 32 different sizes and also line widths of 1 to 16 for a total of 512 rectangles. (2.3) Line arcs are drawn with 20

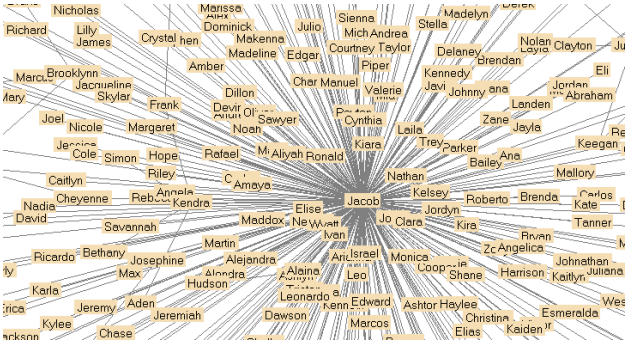

Figure 1: Reference Application

Node movement in this interactive visualization results from simulated spring forces but the user can also manually drag nodes.

[2].NET: `http://www.microsoft.com/net`; Java: `http://java.sun.com`; Flash: `http://www.adobe.com/products/flash`; Silverlight: `http://www.microsoft.com/silverlight` (all accessed on April 24th, 2008)

[3]Visual Variables are a specified set of modifications that can be applied to objects in order to encode information (from `http://www.infovis-wiki.net/index.php?title=Visual_Variables`, accessed on April 24th, 2008). The term was introduced by Bertin [1].

[4]`http://www.prefuse.org/gallery` (accessed on April 24th, 2008)

Figure 2: Graphics Benchmark
Consisting of polygons (left), rectangles (middle) and arcs (right).

different combinations of size and start/end angle and also line widths of 1 to 16 for a total of 320 arcs. Because of different rendering philosophies among the platforms, development and comparability were difficult. We decided to measure the time needed for executing the programming commands to draw objects as the basis for our benchmark. An overview of the times we measured is given in Table 2.[5] (3) We analyzed available information and made a theoretical examination of the platform with consideration of the presented criteria. We did all this using the four platforms.

4 Results

To compare the performance of the interactive InfoVis application, we counted the number of nodes that could be used with the implementation still running at a frame rate at which separate frames could not be distinguished from each other with the naked eye. The Java applet could handle even more nodes than the reference application, Silverlight and especially Flash were considerably slower (Table 1). When judging these results, you have to keep in mind that the Silverlight plugin was an alpha version while for other VMs and plugins a final one could be used. Because of the bad performance in drawing polygons, we also tried to improve the speed of the Flash version by disabling the drawing of the links, but there was only a minor, hardly judgeable improvement.

Implementation	Nodes
Java	1,000
.NET Reference	700
Silverlight	350
Flash	150

Table 1: Maximum Number of Nodes
The number of nodes which allowed a sufficient frame rate for different technologies.

The results of the benchmarking application are shown in Table 2. Any type of platform can be used for server-based rendering. We present rendering into memory done in .NET/GDI+ and Java as examples. Using fast anti-aliasing is not significantly slower than disabled anti-aliasing in GDI+. However, if you also use transparency, there is not much difference in also using good anti-aliasing. Using Java for rendering into memory is not too different from GDI+. However, using transparency or good anti-aliasing is more costly. Surprisingly, when drawing polygons or arcs in an applet, anti-aliasing optimized for quality is faster than disabled or optimized for speed. When drawing a huge number of rectangles, switching off anti-aliasing makes sense in combination with disabling transparency. The benchmarking results for Flash are partly below the measurable limit, but only the drawing times for rectangles and arcs were really faster than visible. For polygons, while the code finished very fast, we had to wait approximately nine seconds (29 with transparency) for the visualization to be actually shown. Silverlight is slightly slower than Flash, but the times are more realistic, as all visualizations were immediately shown, a fact which is very impressive for the polygon part. Combining the benchmark results with the ones from the application, Java seems to outperform other client-based technologies when much calculation beyond the pure graphics has to be done. Otherwise, Silverlight and Flash are faster.

Implementation	Alpha	AA	Polygons (ms)	Rectangles (ms)	Arcs (ms)
.NET Reference	No	None	22,403	8	4557
		Fast	22,364	8	4541
		Good	46,426	888	9467
	Yes	None	46,883	980	9446
		Fast	47,163	993	9485
		Good	50,492	1,002	10,277
.NET server-side	No	None	445	53	152
		Fast	451	53	153
		Good	1,694	151	437
	Yes	None	972	220	275
		Fast	988	222	274
		Good	1,819	227	447
Java server-side	No	None	329	32	94
		Fast	312	32	94
		Good	3,766	188	1,001
	Yes	None	2,953	234	656
		Fast	2,968	234	656
		Good	4,016	172	1,047
Java Applet	No	None	8,766	16	2,687
		Fast	8,876	31	1,922
		Good	7,860	484	1,828
	Yes	None	46,737	656	14,251
		Fast	51,003	563	13,393
		Good	8,281	485	2,063
Flash	No	Good	141[6]	0	0
	Yes	Good	110[6]	0	0
Silverlight	No	Good	182	78	422
	Yes	Good	182	78	422

Table 2: Benchmark of Drawing Speed
.NET reference rendered on the Windows desktop, .NET server-side rendered into memory, Java applet rendered into the browser window, Java server-side rendered into memory, Flash rendered into the browser window, Silverlight rendered into the browser window; Alpha channel for transparency used or not used; anti-aliasing (AA) modes are disabled (none), optimized for speed (fast) and optimized for quality (good); timings are given in milliseconds.

As a popularity measure which is important for people who wish to develop InfoVis applications in a scientific environment, we counted search results on SourceForge[7]

[5] As test system, we used a Pentium D CPU running at 3.0 GHz under Windows XP SP 2. The system was equipped with a Radeon X1300 GPU and 2.0 GB RAM. It had installed the Java Virtual Machine version 1.6.0_04 and Silverlight 1.1.20926. For tests inside of a browser window, Internet Explorer 7.0.5730.11 was used. The Flash plugin used was version 9.0.45.0.

[6] Additional waiting after the code had finished: 9000/29000 ms

[7] `http://www.sourceforge.net` (accessed on April 24th, 2008) SourceForge is the world's largest Open Source software development web site.

for some search terms. The number of Java projects is vast compared to the others. We have searched for "Ajax" to assess the support for interactivity because there are no projects dedicated especially to server-based rendering (Table 3).

Term	Results	Term	Results
Java	15,694	Java AND Visualization	76
Ajax	1,284	Ajax AND Visualization	0
Flash	839	Flash AND Visualization	3
Silverlight	350	Silverlight AND Visualization	0
SVG	258	SVG AND Visualization	10
ActiveX	121	ActiveX AND Visualization	0
VRML	50	VRML AND Visualization	2
X3D	28	X3D AND Visualization	2

Table 3: Searching Terms at SourceForge

4.1 Feature Assessment and Comparison

An overview of the platforms is given in Table 4. Only server-based rendering can be expected to run on any client, all other plaform require a VM or plugin to be installed. This is usually an easy task, but some policies might prevent it in corporate environments. Animation is difficult to achieve using server-based rendering. All frames need to be rendered before the animation starts or lags in the animation are possible because the speed of network connections depend on many unpredictable factors. If the code runs on the client, no network connection has to be considered. It is save to render new frames while the animation is shown if the rendering process is fast enough. User interaction is very difficult with server-based rendering because modifications usually have to be transferred to the client (see section 4.2). It is far easier with client-based rendering. Modifications result in a new image which is immediately drawn directly on the client. The dynamic loading of content is restricted to already rendered images and the dynamic overlaying of text with server-based rendering. Client-based platforms can load any data dynamically. ECMAScript only supports http as protocoll, other languages are more powerful and all protocols can be implemented. Java, Flash, and Silverlight come with huge programming libraries. The one for Java seems to be the biggest, but only a small part is helpful for visualization. We will now present more detailed results on server-based rendering, Java, Flash, and Silverlight.

4.2 Server-Based Rendering

This technique provides complete independence of the client system. To provide interactivity, ECMAScript and the official W3C Document Object Model (DOM) need to be supported. This is the case for all major browsers.[8] Several browsers have their own scripting dialects, but as they are all based on the ECMAScript standard, one can avoid problems by sticking to it. For rendering graphics primitives into raster images and saving them in formats, like GIF, JPG, or PNG, numerous tools are available, targeting virtually any platform. The only format well-supported by web browsers for showing animations consisting of one file is GIF. An alternative is using a JPG or PNG file for each frame. The files can be preloaded for a smooth animation using ECMAScript. Frame selection by the user is not possible with animated GIF but relatively easy using the ECMAScript approach. User interaction is the main weakness of server-based rendering. All graphic primitives that might be modified by the user have to be preloaded or only very simple effects need to be chosen (like drawing a box). Depending on the complexity of the effect, a considerable amount of ECMAScript is necessary which modifies the DOM of the page. Recently, libraries like Ext JS[9] and converters from other platforms like AjaxSwing[10] have been developed that simplify the ECMAScript development but are mainly focused on standard UI controls. Modern web browsers are much more stable than back in the 1990s. Therefore, server-based rendering without additional plugins can be seen as reference for the stability of a possible environment. Of course, the code of the visualization application also has to be stable and browser-specific lacks of standard-compliance have to be accounted for. It will always be possible to render images on the server, but the format being used to transfer the images might no longer be supported by the browsers. For GIF, JPG, and PNG, this is not likely because they are all well established with open communities as well as Microsoft. On the scripting side, ECMAScript and DOM are standardized by Ecma International and the the W3C respectively, and also used by open communities as well as Microsoft, making them as future-proof as the mentioned raster image formats. The ECMAScript documentations from the W3C are fairly usable, but the total amount of documentation is limited, making it difficult to learn the language.

4.3 Java Applets

Java requires the application to do the rendering on its own, targeting the client area in pixels. However, among the free libraries available for Java, several renderers exist which free the developer of this work. Some renderer libraries for Java provide features supporting animation, like timeline objects that keep track of animation automatically. When providing user frame selection, it is possible to fire a manual drawing event, but caching the frames is also an option. Java on its own has reached a very stable state. However, the installed version of the VM differs among clients. Incompatible versions can prevent the Java applet from running. There are also problems embedding applets in some operating system or browser configurations. The

[8] As "major browsers" we consider Internet Explorer, Firefox, Safari, and Opera.

[9] `http://extjs.com` (accessed on April 24th, 2008)

[10] `http://www.creamtec.com/products/ajaxswing` (accessed on April 24th, 2008)

	Server-based Rendering	Java	Flash	Silverlight
Run Capability	Everywhere	VM; wide availability	Plugin; wide availability	Plugin; limited availability
Rendering	Render manually or use one of many libraries	Render manually or use one of many libraries	Draw manually; rendering by plugin	Define graphics primitives; rendering by plugin
Animation	Difficult; animated GIF or ECMAScript tricks	Easy; render frames using timer events or some library	Easy; draw frames using timer events or timeline	Easy; modify objects according to timeline
Interactivity	Very difficult	Easy	Easy	Easy
Communication	Rendered images, labels; http	Any data; any protocol	Any data; any protocol	Any data; any protocol
Stability	Very High	Varies	High	Average
Graphics Speed	Varies (http connection)	Average	Fast	Average
Overall Speed	N/A	Fast	Slow	Average
Future	Very good	Good	Good	Average
Development	Difficult learning; documentation cluttered; no real IDE	Easy learning; good documentation, good IDEs	Average learning; documentation cluttered; good IDEs	Easy learning; documenation cluttered; very good IDE

Table 4: Overview of Technologies. Explanation in sections 4.1–4.5.

use of Java applets has declined during the last years with the increasing flexibility of Flash. Still, it has many supporters and a lot of software is written in Java. The amount of work which has been put into Java libraries (Table 3) will presumably keep the platform alive for many years. For Java, several sophisticated IDEs like Netbeans[11] and Eclipse[12] are available. A good documentation is available as well as countless other information sources, making it fairly easy to learn the language.

4.4 Flash

While the Flash drawing functions are similar to those of Java, they do not actually perform the drawing directly. In fact, they cache the graphics primitives and the drawing is performed in a renderer thread independent of the user code. Flash not only supports timer events but also has timeline support already built in the platform. One has to keep in mind, though, that calculating a frame is independent of the frame being drawn by the renderer task. As the output of the renderer cannot be cached, user frame selection has to fire manual draw events. Because Flash was developed to integrate into a web site in the first place, it is very stable inside the browser. There are still many different versions installed among clients. Flash checks the version, but if the installed plugin is too old, the application will not run either. In the web of 2008, Flash can be found on more sites than any other technology that requires a plugin. Therefore, it is most likely to prevail installed on client systems for the next years. For Flash there are also development environments comparable to those for Java, but debugging is more complicated because the compilation does only run in the plugin and a special debugging plugin is needed. ActionScript is a language based on the ECMAScript specification but greatly improved so coding is much easier. However, the ECMAScript compatibility causes several weaknesses, like poor type safety. The provided library of Flash seems to have more predeveloped solutions, like classical types of charts than the ones provided by Java or Silverlight. Unfortunately, those are not flexible enough to facilitate complex interactive visualizations. Flash is currently third regarding free libraries (Table 3) and has a lot of documentation, but it is severely cluttered. This might be a result of recent design changes done by Adobe. The fact is also visible by the large amount of information available for ActionScript 2.0 whilst not many resources are available for the current version 3.0 up to now. This also complicates learning ActionScript.

4.5 Silverlight

The Silverlight plugin is only available for computers running Windows or MacOS. A free alternative plugin named Moonlight is being developed by the Mono project[13], but as of April 2008, a simple way of installing is not provided. Silverlight takes the approach of Flash even further: Graphics primitives are defined in a description language. They can also be modified or complemented with additional graphics primitives using code. In Silverlight, there are no timer events but timeline support. Drawing separate frames is possible as in Java and Flash, but you can also modify existing objects instead of drawing new ones. The renderer will show the modifications in its next cycle. Like in Flash, caching the renderer output is not possible, so user frame selection requires firing a manual draw event. Silverlight is a much newer technology than its competitors. Version 1.0 has quite limited functionality and version 2.0 is still in alpha state. This alpha version has crashed and jammed the Internet Explorer several times during our test development phase, but the clean code structures of the .NET environment are enforced for Silverlight, so it is likely that the final version will be quite stable for such a new technology. While most analysts grant Silverlight a comparison to Flash on the same level, a quick research with Google reveals that for each site that requires Silverlight, there are thousands of sites requiring

[11] `http://www.netbeans.org` (accessed on April 24th, 2008)

[12] `http://www.eclipse.org` (accessed on April 24th, 2008)

[13] `http://www.mono-project.com/Moonlight` (accessed on April 24th, 2008)

Flash. This is due to the fact that it is very difficult to introduce a new technology into an established medium, especially, if something has to be installed that normally is not part of the environment. Still, it is possible that Silverlight will gain ground soon as more people start using it. In summary, the future of Silverlight is very hard to predict at the moment. Silverlight can be developed in Visual Studio which in our opinion is superior to all other IDEs currently available. Debugging also needs to connect to a browser, but it can be done with the regular plugin. The provided library for Silverlight has powerful functions to deal with graphics primitives, but some of them are more difficult to use than their counterparts in Java or Flash. Although being very new, there are very interesting free libraries currently in development for Silverlight. The documentation for Silverlight is still work in progress and it is difficult to find in-depth information. However, the class library is similar to the Windows Presentation Foundation (WPF) and can be accessed in languages like C# and Visual Basic, reducing the learning effort.

5 Conclusion and Future Work

No technology is superior to all others in all situations. Developers need to consider the environment and user group they address as well as their requirements. These prerequisites define the priorities. Therefore, even the platform-independence of web applications is limited. Server-based rendering will work in conjunction with almost any client, but because of the availability of Flash and Java they can be seen as equal for most targets. In some company cultures, however, the chances for getting Silverlight installed on the clients might be better. If developers are worried about the data transfer infrastructure, especially on the server side, server-based rendering is the easiest solution. However, this is a tradeoff with severe response time problems. With regards to stability, the expertise of the developer seems to be more important than the platform used—however, the use of a beta product, like Silverlight, always entails many uncertainties. In centrally administrated environments, keeping a platform functional over a longer period of time is less problematic. Otherwise, only server-based rendering, Flash and Java seem to be safe bets.

We have focused on the facet of interactive visualization. Other aspects might be important if movies and/or audio have to be included. There is also the possibility of using not the common consoles but other UIs. For developing further benchmarks, the Bubblemark Animation Test[14] could be a good starting point. While our survey encompasses a broad range of the market, we have not investigated niche products or technologies specialized for 3D-graphics, like X3D. We did not include scalable vector graphics (SVG) dynamically created on the client using ECMAScript, because ECMAScript is the least powerful language of the field, making it very time-consuming to develop a complete InfoVis application. New technologies like the canvas tag which implements graphics rendering directly in HTML are emerging, but it is difficult to decide at which time a technology is ready for use in an application without a large team to keep pace with development changes. We also have focused on clients that are running in the browser. There are also intermediate solutions with local applications being installed on-demand by web sources, e.g., Java Web Start and Adobe AIR. So the range of compared products might need broadening. At the moment we are taking a look at the next level of development, comparing different graphics libraries.

[14] `http://www.bubblemark.com` (accessed on April 24th, 2008)

Acknowledgements This work was supported by the program "FIT-IT Visual Computing" of the Federal Ministry of Transport, Innovation and Technology, Austria. Project number: 813388.

References

All web references have been accessed on April 24th, 2008.

[1] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[2] E. Burnette. Is Flash better than Java?, April 2007. URL `http://blogs.zdnet.com/Burnette/?p=286`.

[3] J. Ezell. Silverlight vs. Flash: The Developer Story, May 2007. URL `http://weblogs.asp.net/jezell/archive/2007/05/03/silverlight-vs-flash-the-developer-story.aspx`.

[4] J.J. Garrett. Ajax: A New Approach to Web Applications, February 2005. URL `http://www.adaptivepath.com/ideas/essays/archives/000385.php`.

[5] A. Kobsa. An empirical comparison of three commercial information visualization systems. *Proc. of IEEE Symp. on Info. Vis. 2001 (INFOVIS 2001)*, pages 123–130, 2001.

[6] M. Lyda. Flash ActionScript Performance vs. JavaScript, November 2006. URL `http://www.oddhammer.com/actionscriptperformance/set4`.

[7] A. Pasztory. Flash vs. Ajax. URL `http://www.pasz.com/articles/FlashVsAjax.html`.

[8] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.

[9] E.M. Wohlfart. A Detailed Comparison of Information Visualization Tools Using a Reference Data Set. Thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, November 2007. URL `http://ieg.ifs.tuwien.ac.at/projects/infovis-compare`.