Eurographics Conference on Visualization (EuroVis) (2014)
N. Elmqvist, M. Hlawitschka, and J. Kennedy (Editors)



# FluidDiagrams: Web-Based Information Visualisation using JavaScript and WebGL

Keith Andrews[1] and Benedict Wright[2]

[1]Institute for Information Systems and Computer Media (IICM), Graz University of Technology, Austria
[2]Institute for Software Technology (IST), Graz University of Technology, Austria

**Abstract**

*Much attention has been focused on the provision of information graphics and visualisations inside a web browser. Currently available infovis toolkits produce graphical output by either injecting SVG nodes into the DOM or using the JavaScript Canvas 2D API. FluidDiagrams is a prototype information visualisation framework written in JavaScript which uses the WebGL 3D JavaScript API for its output, falling back to Canvas 2D as necessary, via the Three.js library. Six visualisations are currently implemented: bar chart and line chart, scatter plot and parallel coordinates for multidimensional data, and cone tree and hyperbolic for hierarchies.*

*Anecdotally, visualisations using SVG nodes in the DOM for output can become rather sluggish when displaying more than a few dozen items. Visualisations using Canvas 2D exhibit similarly slow performance. WebGL utilises hardware acceleration where available and promises much better performance for complex visualisations, potentially in the order of many thousands of items without becoming unresponsive.*

*A comparison of parallel coordinates visualisations with 100 records in 20 dimensions compared three implementations: FluidDiagrams (WebGL), FluidDiagrams (Canvas 2D), and D3 (using SVG nodes). They achieved 62, 6, and 10 frames per second respectively. The FluidDiagrams (WebGL) implementation was able to render 1,000 records in 20 dimensions at 18 frames per second, compared to 1 and 6 respectively.*

Categories and Subject Descriptors (according to ACM CCS): I.3.4 [Computer Graphics]: Graphics Utilities—Graphics Packages

## 1. Introduction

As applications of information visualisation migrate from company and university research labs into business applications and more everyday use by the general public, a need to be able to provide visualisations embedded inside web pages and web applications is emerging.

Implementations of visualisations inside the web browser rely on one of three underlying technologies to produce graphics: Java applets, Flash, or JavaScript. Java applets [GJS*13] are clunky and often require the user to install a Java plugin, which are not available for mobile web browsers. Flash [Ado10] also requires a browser plugin, which comes pre-installed on many desktop browsers, however support for Flash on mobile browsers has been discontinued. This leaves JavaScript [Fla11] as the technology of choice for new developments requiring graphics which are to be run inside a web browser, particularly in the light of the ever-expanding use of mobile web browsers.

## 2. Browser-Native Graphics Technologies

Given the assumption that JavaScript is the technology of choice for web-based visualisations, there are essentially three ways to draw graphics natively inside the web browser with current JavaScript and web standards:

- *SVG injection*: Scalable Vector Graphics (SVG) [W3C11, DFS12] nodes are inserted (injected) into the Document Object Model (DOM) [W3C14] using JavaScript and are then interpreted and drawn by the web browser.
- *Canvas 2D*: The HTML5 Canvas 2D Context [W3C13, Gea14] allows developers to create graphics directly to an area of the web page using a set of 2D drawing primitives.
- *WebGL (Canvas 3D)*: WebGL [Khr13] is an immediate



DOI: 10.2312/eurovisshort.20141155

mode 3D rendering API, providing a 3D Context for the HTML5 Canvas element. WebGL is based on the widely used standard for 3D graphics, OpenGL ES 2.0 [ML10].

## 3. JavaScript Graphics Libraries

There are also a number of JavaScript graphics libraries which build on top of one or more of the browser-native graphics technologies mentioned above and provide simple graphics primitives such as line, circle, square, sphere, or cone, and transformations. They include:

- *Raphaël*: Raphaël [Bar14] is a JavaScript library which provides basic 2D graphical primitives such as line, rectangle, and circle. They are drawn by injecting corresponding SVG nodes into the browser DOM.
- *Pixi.js*: Pixi [Gro14] is a 2D drawing library which renders using WebGL for fast performance, with a fallback to Canvas 2D if WebGL is not available. Primitives include line, rectangle, and ellipse.
- *Three.js*: Three.js [Cab12b] is a powerful JavaScript 3D rendering library, which uses WebGL where available and also has fallbacks to Canvas 2D and SVG injection. A 3D scene graph is constructed by adding primitives such as cubes and spheres and applying textures.

## 4. InfoVis Toolkits Using JavaScript

A number of infovis toolkits implemented in JavaScript which provide fully interactive charts and visualisations have also emerged:

- *gRaphaël*: gRaphaël [Bar12] is a simple JavaScript library which provides four basic 2D charts: bar chart, pie chart, line chart, and dot plot. It draws by SVG injection using the Raphaël library.
- *JIT*: The JavaScript InfoVis Toolkit (JIT) [Bel14] provides a suite of information visualisations and generally draws its graphics using the standard JavaScript Canvas 2D Context. JIT also provides some support for WebGL inside a 3D Canvas.
- *D3*: Data-Driven Documents (D3) [Bos11, BOH11] is a widely-used suite of information visualisations drawn by SVG injection. It also supports animated transitions between related visualisations.

## 5. FluidDiagrams

FluidDiagrams [AW14] is a new, web-based information visualisation framework which uses JavaScript and WebGL (and Canvas 2D) through the Three.js library. To take advantage of any available graphics hardware, FluidDiagrams uses WebGL by default for all rendering, both 2D and 3D. Canvas 2D is used as a fallback, in case WebGL is not available. FluidDiagrams demonstrates that responsive, high-performance information visualisations inside web pages are achievable.

The internal architecture of FluidDiagrams is shown in Figure1. A parser factory `FDParserFactory` determines which kind of input parser is used. Currently, parsers are available for tabular data in CSV and hierarchical data in JSON format.

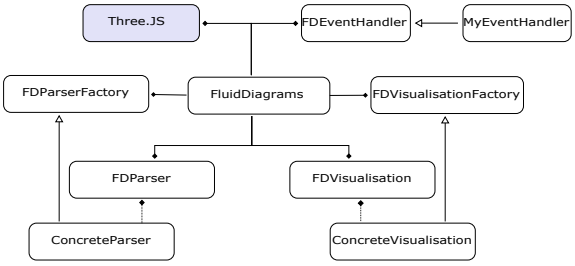


**Figure 1:** *The internal architecture of FluidDiagrams.*

A visualisation factory `FDVisualisationFactory` determines which visualisation is used for the current dataset. Visualisations themselves are implemented using the primitives provided by the underlying Three.js library. Three.js then does the required drawing using either WebGL or Canvas 2D. The SVG performance of Three.js proved insufficient for our applications and hence is not currently used in FluidDiagrams.

## 6. FluidDiagrams Visualisations

Six visualisations have so far been implemented within FluidDiagrams. Interactions typical for each visualisation are also provided: for example hovering over a data point in a Bar Chart and filtering records and reordering and inverting axes for Parallel Coordinates.

The FluidDiagrams Bar Chart (Figure 2) and Line Chart (Figure 3) illustrate simple information graphics. In FluidDiagrams, however, 2D visualisations are also constructed with 3D graphics primitives (with $z = 0$) so as to take advantage of WebGL's superior perfomance wherever available.

The FluidDiagrams Hyperbolic browser (Figure 4) [LRP95] and Cone Tree (Figure 5) [RMC91] visualisations are for visualising information hierarchies. As before, the 2D Hyperbolic browser is constructed with flat 3D graphics primitives. The Cone Tree, being a 3D visualisation, uses actual 3D graphics primitives.

The FluidDiagrams Parallel Coordinates (Figure 6) [Ins85] and Scatter Plot (Figure 7) visualisations are for multi-dimensional (tabular) data. Again, flat 3D graphics primitives are used to construct these 2D visualisations.

## 7. Performance Comparison

To investigate the anecdotally higher performance of information visualisations using WebGL, a performance comparison was run. Even though the FluidDiagrams visualisations

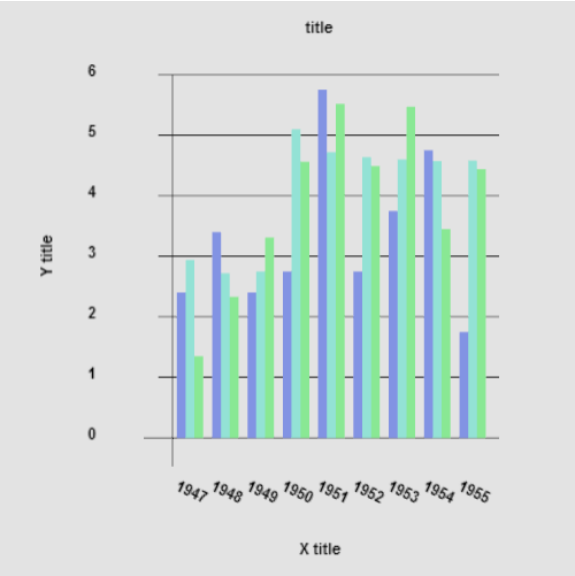


**Figure 2:** *FluidDiagrams Bar Chart.*

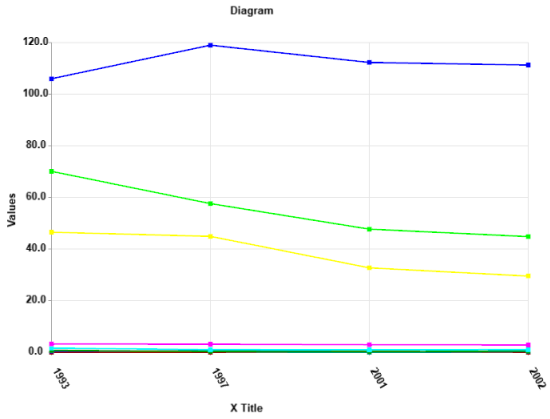


**Figure 3:** *FluidDiagrams Line Chart.*

have not yet been explicitly optimised for performance, clear performance advantages can be observed.

The FluidDiagrams Parallel Coordinates implementations running WebGL and Canvas 2D were compared to a D3 implementation [Bos13] using SVG injection. Five datasets were randomly generated to contain 10/50 (dimensions/records), 10/100, 20/100, 20/1000, and 20/10000 dimensions and records. Figure 8 shows the FluidDiagrams Parallel Coordinates visualisation with the comparison dataset of 20 dimensions and 100 records.

The comparison was carried out on a desktop PC with a 2.8 GHz Quad Core Intel I5 processor, 8 gb of RAM, and an Nvidia GeForce GTX 460 graphics card. Firefox 27.0.1 was

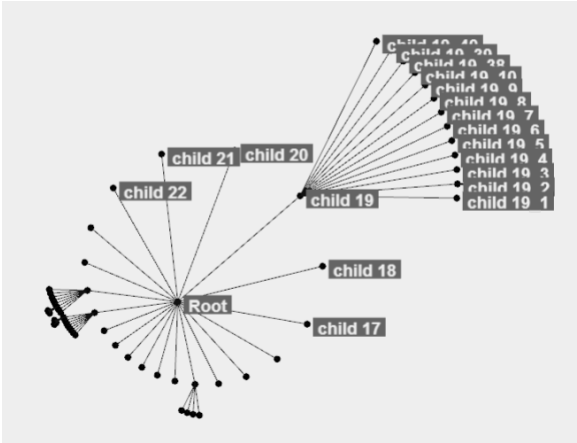


**Figure 4:** *FluidDiagrams Hyperbolic browser for hierarchies.*

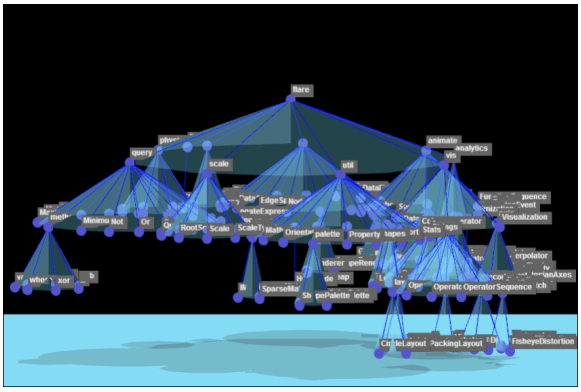

**Figure 5:** *FluidDiagrams Cone Tree.*

used as the browser and the Stats.js package [Cab12a] was used to record frame rates in frames per second.

The results of the performance comparison are shown in Figure9. WebGL rendering performance is consistently much higher than both Canvas 2D and SVG injection. The FluidDiagrams Parallel Coordinates WebGL implementation is able to handle several thousand records of 20 dimensions without becoming unresponsive.

## 8. Concluding Remarks

FluidDiagrams is a new web-based information visualisation toolkit. It demonstrates the practicality and potential performance gain attainable by using WebGL for underlying graphics output.

The graphics library used by FluidDiagrams, Three.js, contains numerous features (mostly pertaining to 3D graphics rendering) which are not needed in most information vi-

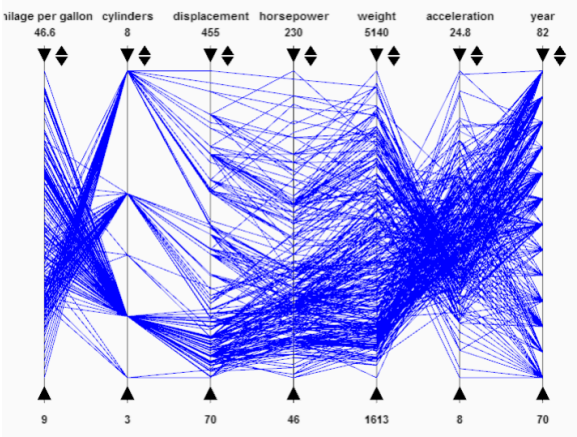


**Figure 6:** *FluidDiagrams Parallel Coordinates visualisation.*

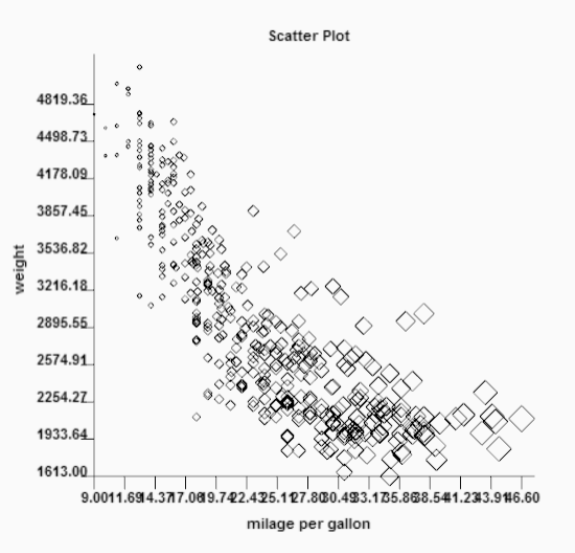


**Figure 7:** *FluidDiagrams Scatter Plot visualisation.*

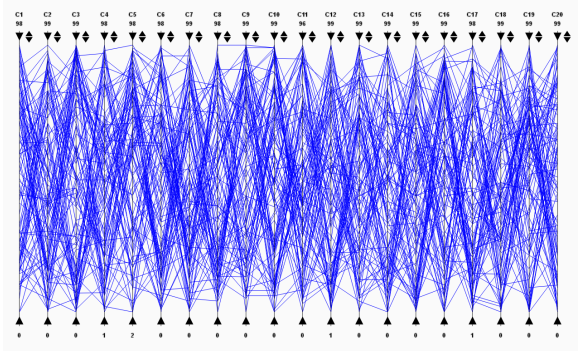

**Figure 8:** *The FluidDiagrams Parallel Coordinates visualisation with one of the comparison datasets, in this case with 20 dimensions and 100 records.*

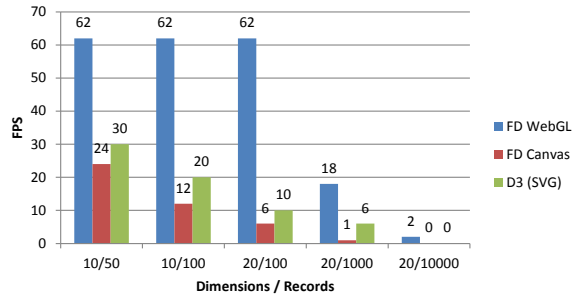


**Figure 9:** *Performance comparison results for Parallel Coordinates visualisations using WebGL (FluidDiagrams), Canvas 2D (FluidDiagrams), and SVG injection (D3) respectively in terms of average frame rate in frames per second (fps).*

sualisations. In addition, its support for SVG injection rendering was sometimes very slow (and hence is not used in FluidDiagrams). We are currently experimenting with a custom slimline graphics library called FluidDiagrams Graphics Layer (FDGL), which translates the small set of 2D and 3D primitives necessary for most information visualisations (line, polygon, circle, text cube, sphere, cone, group, etc.) into corresponding calls to WebGL, Canvas 2D, SVG for DOM injection, and SVG for file export. We are also experimenting with Pixi [Gro14] as a potential underlying graphics library for a future version of FluidDiagrams.

FluidDiagrams is available as open source code under the liberal MIT licence from the project web site [AW14].

## 9. References

### References


[Ado10] Adobe: *ActionScript 3.0 for Adobe Flash Professional CS5*. Adobe, June 2010. 1

[AW14] Andrews K., Wright B.: Fluiddiagrams, 2014. URL: http://projects.iicm.tugraz.at/fluiddiagrams/. 2, 4

[Bar12] Baranovskiy D.: gRaphael, Aug. 2012. URL: http://g.raphaeljs.com/. 2

[Bar14] Baranovskiy D.: Raphael, Feb. 2014. URL: http://raphaeljs.com/. 2

[Bel14] Belmonte N. G.: JavaScript InfoVis Toolkit, 2014. URL: http://thejit.org/. 2

[BOH11] Bostock M., Ogievetsky V., Heer J.: $D^3$: Data-driven documents. In *Proc. IEEE Information Visualization Conference 2011 (InfoVis 2011)* (Oct. 2011), IEEE. URL: http://vis.stanford.edu/files/2011-D3-InfoVis.pdf, doi:10.1109/TVCG.2011.185. 2

[Bos11] Bostock M.: D3 Data-Driven Documents, Nov. 2011. URL: http://d3js.org/. 2

[Bos13] Bostock M.: D3 Parallel Coordinates Example, Nov. 2013. URL: http://bl.ocks.org/mbostock/7586334. 3

[Cab12a] Cabello R.: stats.js JavaScript Performance Monitor, Sept. 2012. URL: https://github.com/mrdoob/stats.js/. 3

[Cab12b] Cabello R.: Three, Aug. 2012. URL: https://threejs.org/. 2

[DFS12] Dailey D., Frost J., Strazzullo D.: *Building Web Applications with SVG*. Microsoft Press, July 2012. 1

[Fla11] Flanagan D.: *JavaScript: The Definitive Guide*, 6 ed. O'Reilly, May 2011. 1

[Gea14] Geary D.: *Core HTML5 Canvas: Graphics, Animation, and Game Development*. Prentice Hall, May 2014. 1

[GJS*13] Gosling J., Joy B., Steele G., Bracha G., Buckley A.: The java virtual machine specification, Java SE 7 edition. Oracle, Feb. 2013. URL: http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf. 1

[Gro14] Groves M.: Pixi.js, Feb. 2014. URL: http://pixijs.com/. 2, 4

[Ins85] Inselberg A.: The plane with parallel coordinates. *The Visual Computer 1*, 4 (Dec. 1985), 69–91. doi:10.1007/BF01898350. 2

[Khr13] Khronos: WebGL specification 1.0. Khronos Group Working Draft, Nov. 2013. URL: http://www.khronos.org/registry/webgl/specs/latest/1.0/. 1

[LRP95] Lamping J., Rao R., Pirolli P.: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. CHI'95* (Denver, Colorado, USA, May 1995), ACM, pp. 401–408. URL: http://sigchi.org/chi95/Electronic/documnts/papers/jl_bdy.htm, doi:10.1145/223904.223956. 2

[ML10] Munshi A., Leech J.: OpenGL ES common profile specification version 2.0.25. Khronos Group, Nov. 2010. URL: http://www.khronos.org/registry/gles/specs/2.0/es_full_spec_2.0.25.pdf. 2

[RMC91] Robertson G. G., Mackinlay J. D., Card S. K.: Cone trees: Animated 3D visualizations of hierarchical information. In *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI'91)* (New Orleans, Louisiana, USA, May 1991), ACM, pp. 189–194. doi:10.1145/108844.108883. 2

[W3C11] W3C: Scalable vector graphics (SVG) 1.1. W3C Recommendation, Aug. 2011. URL: http://w3.org/TR/SVG11/. 1

[W3C13] W3C: HTML Canvas 2D Context, Level 2. W3C Working Draft, Oct. 2013. URL: http://www.w3.org/TR/2dcontext2/. 1

[W3C14] W3C: DOM4. W3C Last Call Working Draft, Feb. 2014. URL: http://www.w3.org/TR/dom/. 1