Declarative Language Design for Interactive Visualization

Jeffrey Heer, Stanford University

Summary

Successful visual analytics technologies must address a number of technical challenges, not least of which is the increasing heterogeneity of commodity hardware and interactive devices. Next-generation visualization tools should ideally support a variety of interfaces, from traditional desktop applications to browser-based web clients to multi-touch enabled mobile devices. Furthermore, they should effectively capitalize on hardware trends such as multi-core computing and specialized graphics hardware. While point designs exist for each of these areas, we currently lack a coherent framework for visualization design and deployment across heterogeneous platforms.

To address these issues, we argue for a break with current component-model architectures for visualization, and instead propose to design and develop declarative, domain-specific languages for visualization design. Much how languages such as SQL and MDX have insulated database users from database implementation specifics, we believe that a declarative language for visualization design will allow designers to focus on creating customized, interactive visualizations while letting the underlying implementation handle behind-the-scenes optimization and address platform-specific details such as rendering and UI event handling.

Our experiences creating Protovis [1]—a declarative, JavaScript-based domain specific language for visualization—has shown us that a well-crafted language can simplify visualization specification while maintaining a high-degree of visual expressiveness, including support for customized data graphics, maps, and network diagrams. We will extend this line of work to other platforms, beginning with an implementation in the Java programming language intended to enable desktop and mobile applications, and back-end support for cloud-based visual analytics. For example, we will investigate:

- **Optimization.** By decoupling specification from implementation, we can explore a variety of optimizations without interfering with the work of designers. Optimizations that we plan to implement include: runtime-compilation of visualization specifications, multi-threaded parallel processing, and hardware-accelerated rendering.
- **Format agnostic processing.** Most visualization platforms require developers to corral their data into a particular in-memory representation. Protovis-Java will visualize data elements represented by any arbitrary Java *Object*, so long as they are aggregated in *Iterable* collections. This will simplify data formatting requirements and facilitate visualization of the output of varied statistical and data mining routines.
- **Multi-platform support.** We plan to develop rendering and event-handling infrastructures that abstract above the host windowing system. This will enable a variety of renderers (e.g., OpenGL via JOGL, Java2D, SWT, Android / OpenGL ES for mobile devices) and interaction paradigms (e.g., mouse-based or touch-based interaction). Designers will be able to retarget applications with minimal effort.

To date, we have built prototype implementations of these ideas, and have achieved an order of magnitude performance improvement over existing toolkits such as Prefuse [2] and the InfoVis Toolkit [3]. Going forward, our research plan will consist of **(a)** continued design and development of the Protovis language, **(b)** systematic benchmarking of Protovis performance to identify the key contributions to improved scalability, and **(c)** public release of the software under an open-source license and publication of our results in major Information Visualization research venues. We will evaluate our system through in-house design critiques, performance benchmarks, and feedback from domain users, including VACCINE partners and collaborating researchers at Stanford University. Success will be measured in part by the utility of our system as a framework for VACCINE-related visual analytics applications. The primary research outcome will be new insights for visualization system design, instantiated in novel software architectures for visualization.

We will collaborate with VACCINE partners through the sharing of our software systems, development and support of example applications leveraging our systems, and distribution of our architectural design decisions. The reduced threshold for visualization creation using our language may also prove valuable in teaching visualization design to students. We have and will continue to use these tools in our visualization courses and will further develop corresponding tutorials and teaching materials.

References

1. Bostock, M. and Heer, J. Protovis: A Graphical Toolkit for Visualization. IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis'09), 15(6), pp. 1121-1128, Nov/Dec 2009.
2. Heer, J., Card, S. K. and Landay, J. Prefuse: A Toolkit for Interactive Information Visualization. ACM Human Factors in Computing Systems (CHI'05), pp. 421-430, Apr 2005.
3. Fekete, J.-D. The InfoVis Toolkit. IEEE Information Visualization (InfoVis'04), pp. 167-174, Oct 2004.