

Declarative Visualization of Stream-based Data

Joseph Cottam*

Indiana University[†], Bloomington, IN, USA

1 Introduction

Although analysis and visualization of dynamic data are becoming more important, most visualization tools provide little support for the unique requirements of such data. This deficiency extends all the way to the visualization models; most models imply that all data for a process are available at its start. As dynamic data are forced into static frameworks, additional problems appear with refining and updating the visualization code-base: once it works, it's hard to change.

Our research is targeted at improving the process of visualizing dynamic data. We are developing a conceptual model and tools to deal with the unique challenges that dynamic data presents. Our prototype system, called Stencil, draws from research on declarative languages to extend the abilities of visualization frameworks. By allowing visualization practitioners to focus on the issues of core issues of visual representation (and not mechanical issues of data flow and storage), this work will reduce the conceptual requirements of dynamic data visualization and improve the representations eventually produced.

2 Stencil

The Stencil system is composed of three parts, inspired by compiler generators (e.g., YACC/Lex, see figure ??): Model, Language and Compiler. The Stencil model is responsible for bringing the issues of dynamic data and graphic refinement to the forefront. This is first accomplished through the use of data streams as the only communication medium. Data streams capture many important concepts of dynamic data sources. All data must be moved in a data stream; therefore, all processors must handle dynamic data. Graphic refinement is addressed by borrowing concepts from graphic design. First, a layers concept combines visual grouping with its process derivation. Second, a formal glyph concept handles rendered elements. Explicitly including these elements keeps the Stencil model close to the graphic domain while providing flexibility to include other domain models. Other elements of the model assist with stream processing and interaction with the visual display.

Although compiler generators are based on conceptual models, the languages they employ mediate the boundary between conceptual purity and pragmatics. The Stencil language similarly must make concessions for both common

*jcottam@cs.indiana.edu

[†]This work was supported in part by a grant from the Lilly Endowment.

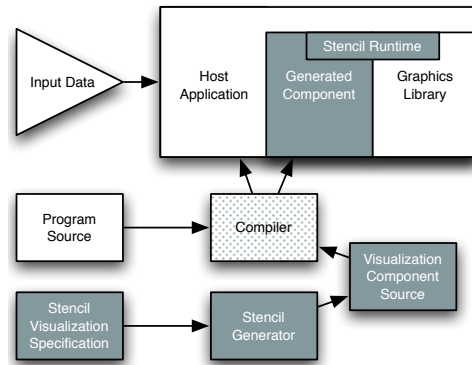


Figure 1: The relationship of Stencil to the development environment mirrors that of compiler generators.

visualization patterns and to interface with the wealth of existing visualization code. The first key to the Stencil language is a declarative basis, where straight-forward program semantics tends to keep updates to the program accessible. Second, the layers metaphor is believed to provide a natural decomposition strategy for many graphic operations. Finally, the ability to interface with existing visualization algorithms in a way similar to *ad hoc* syntax-directed transformations lets step a program out of the Stencil model when appropriate without major cost. For example, this external function call ability allows an existing spring-force embedding routines to be directly employed instead of re-implemented.

3 Research Directions

The driving research question is “What abstractions does Stencil need to handle dynamic data visualization?” For example, by including canvas, view and stream elements in the model, many interaction techniques become approachable as data transformations (where user input is represented as an additional stream). Elements related to stream analysis are the focus of our current investigations (e.g., implicit sequence counters).

A second vein of research in the Stencil system is that of visualization optimization. The Stencil language is declarative and necessarily goes through a transformation before being included with a host application. This gives opportunities to automatically tailor the visualization control flow for space and speed efficiencies in ways beyond those offered by traditional libraries.

Our initial prototypes have strongly influenced the provisional Stencil model. We continue to explore this space by developing language/compiler iterations and subsequently applying them to real-world problems. We have looked at social network analysis, large-scale software testing results and sensor network data. We continue to look at new data sets for fresh analysis and representation ideas. We believe that having a conceptual model that explicitly acknowledges both the unique characteristics of dynamic data, as well as the concerns of graphic design will improve the visualization of dynamic data. Subsequent work on the Stencil system promises valuable insight into the nature and requirements of dynamic data visualization.