End User Programming of Visualisations

Mariana Mărășoiu
Computer Laboratory, University of Cambridge
Mariana.Marasoiu@cl.cam.ac.uk

***Abstract*—Visualisations are a useful tool for helping people understand and communicate data. However, flexible and powerful tools for creating visualisations are mainly accessible only to programmers. My proposal is to explore the use of programming by example and direct manipulation as the main paradigms for building visualisation tools for end user programmers.**

I. Introduction

Data is more available than ever and it has the potential of helping us and others. However, data in itself is not useful - it needs to be analysed and understood to become useful, and visual representations are a useful aid in this process (Card et al., 1999). Thus, making and consuming information visualizations have become common activities for many people. Journal articles discussing data often have charts that enhance the explanation given in the text. An increasing use of infographic posters reflects the desire to inform and be informed. Scientists use visualisations, often customized ones, for exploring data and communicating results.

Given the very different groups of people that make use of visualizations, the ways of creating these visualizations are diverse as well, reflecting the skills of visualisation makers. Victor (2013) classifies these ways in three paradigms: using, drawing and coding, each embodied in different software applications. Representative examples are Microsoft Excel[1], Adobe Illustrator[2] and D3.js[3] respectively. However, the flexibility and expressiveness of the software often correlates inversely with accessibility and ease of use: there is a large group of people who can use Excel to create charts, but they only have a reduced set of chart types available. Conversely, the number of people who can take advantage of the power of a tool like D3.js is limited to those who can code.

Discussions with visual data analysts suggest that there is a space where direct manipulation of visual elements could offer benefits, even for expert users. For example, visualisations can enhance conversations informed by previous data without necessarily relying on a current dataset that might not be available yet (Mărășoiu et al., 2016). Existing visualisation tools are not suitable for this purpose, as they assume that there is always a data source. In such cases, analysts use drawing in PowerPoint and sketching on whiteboards, but when data becomes available, they have to recreate the visualisation from scratch.

This research project proposes to look at data visualization tools through the lens of of end user programming. For a large number of users, learning a programming language for making visualizations can be considered too expensive. According to the Attention Investment model (Blackwell, 2002), the cost and investment may be perceived as too high, and the pay-off too low. Thus, end users prefer to use software that allows them either to manipulate the data more directly, for example using spreadsheets over databases, and are more comfortable with a more direct manipulation of the graphics, for example using Illustrator over Processing[4]. The overarching question that this project asks then is "How can research in end-user programming help create more accessible and flexible data visualisation tools?"

In this paper I briefly review the current state of the art in data visualisation and charting software and argue why an end-user programming approach can be fruitful. I describe the current state of my work and offer a proposal for answering the research question above.

II. Current visualisation tools

Most current toolkits for creating dynamic visualisations rely on textual programming languages (e.g. D3.js relies on Javascript, Processing relies on Java). These are often not accessible to people who don't know how to code.

Unfortunately, there are very few environments where non-programmers can easily build dynamic visualisations. For example, Excel is mostly a data manipulation tool, and whilst it offers a set of chart types and ways of customizing them, the user can only easily interact with the underlying data. Creating dynamic behaviour in charts requires knowledge of Visual Basic (for example, adding hover labels to data points can only be done by coding with VBA). Tableau[5] is somewhat more flexible, offering more options for adjusting the visualisation and direct manipulation through the drag and drop of "pills" to bind data to the visualisation. It has however some of the same limitations as Excel: the user doesn't interact directly with the chart, but with menus that expose its properties, and it doesn't allow creation of chart types outside the combination of the templates offered.

Victor (2013) offers a proposal for how an interactive visualisation system might look. He demonstrates a tool which allows shapes to be directly manipulated, and exposes the underlying computational loop in a manner resembling programming by demonstration. Lyra (Satyanarayan & Heer,

[1] https://products.office.com/en-gb/excel
[2] https://www.adobe.com/uk/products/illustrator.html
[3] https://d3js.org/
[4] https://processing.org/
[5] http://www.tableau.com/

2014) also offers a direct manipulation interaction, combining Tableau-style drag and drop for data binding and some direct manipulation of graphical marks. The user mainly interacts with the side panels which describe the visual properties of the chart instead of being able to adjust the marks manually.

III. Research proposal

As mentioned in Section I, my proposal is to complement the existing research in information visualisation toolkits by taking an end user programming perspective. I believe that this would make creating data visualisations more accessible to non-programming end users.

As a first exploration of the main research question, I am currently asking the following "Can programming by example be used by a system for creating data visualisations?" I aim to answer this more particular research question through an ethnographic study of visual data analysts, the results of which would inform the building of a visualisation toolkit prototype.

A. Ethnographic study

I am currently organizing an ethnography with a small visual data analytics company, whose purpose is to inform the building of further visualisation tools (Sharp et al., 2016). I aim to gather an understanding of the work practices of analysts working with non-expert clients. I will take particular interest in the data visualisation tools they're using and how they communicate through visualisations with their clients. I propose to organize the ethnographic study in two phases.

The first phase will be exploratory and I will aim to get an overall feel for the organization. I will be observing the analysts as they work and as they discuss with each other. Interviews will complement the observational methods as necessary. This will provide us with insight into the kinds of analytical and presentational problems the analysts are trying to solve through visualisations, how they're solving them and the obstacles they have to work around along the way.

For the second phase, I propose to follow one specific project from beginning to end, across multiple dimensions: analytic processes, communication, finished artefacts. The project will be selected after the data from the first phase has been analysed, in order to find the most appropriate one for the research project. Since I hope that for this phase I will be able to study the clients of the projects as well as the analysts, I will aim to ensure that the clients are representative for the end user population. I will look in particular at their understanding of visualisations and previous experience with analysing data and creating charts.

B. Programming by example for visualisation tools

I am currently building the groundwork for a visualisation toolkit prototype that uses programming by example (PbE) as the fundamental interaction paradigm. I describe here the general ideas behind it in broad brush strokes as the details of this prototype will be informed by the qualitative study described above.

In a PbE system, "the user provides several examples of the required program behavior, and the system applies an inference algorithm in order to infer a generalized program that will operate in accordance with the user's intentions in other situations not yet seen" (Blackwell, 2006). In the context of data visualisation tools, one can imagine an interaction where the user is drawing a set of shapes on the screen, connects some of the properties of those shapes to the data, the system observes that there is currently data than shapes on the canvas and it then creates corresponding shapes for the rest of the data.

Blackwell (2006) also mentions some of the difficulties of PbE systems: a possible need for a large number of examples, over-generalization and generating negative examples. However, a visualisation toolkit seems to have few of these problems. The inference process is helped out by the graphic marks being connected to the data - which limits the need for generalisation. As such, the user only needs to generate a couple of examples, which are also part of the final visualisation - the impression given then is not that the user is *teaching* the computer, but that the computer is *helping* the user. Further, the inference is presented visually to the user - via the visualisation - which makes debugging easier.

This may seem, at least at first glance, similar to the strategy used by several other prototypes (e.g. Lyra (Satyanarayan & Heer, 2014) and Drawing Dynamic Visualizations (Victor, 2013)). However, I argue that approaching the subject from an end-user perspective will offer interesting and novel design solutions. For example, I am looking at using the layering metaphor of Palimpsest (Blackwell, 2014) for representing the different components of the visualisation, and propose to take a drawing oriented approach to direct manipulation, (e.g. drawing and manipulating graphic marks similar to the techniques used by graphic editing tools like Illustrator). A further inspiration is the Smalltalk programming language, and in particular the development environment associated with it. The set of visual elements whose visual characteristics have associated data can be viewed as a set of "objects" and their "properties".

REFERENCES

[1] Blackwell, A. F. (2002). First steps in programming: A rationale for attention investment models. Proceedings - IEEE 2002 Symposia on Human Centric Computing Languages and Environments, 2–10.

[2] Blackwell, A. F. (2006). Psychological Issues in End-User Programming. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), End User Development (Vol. 9, pp. 9–30). Dordrecht: Springer Netherlands.

[3] Blackwell, A. F. (2014). Palimpsest: A layered language for exploratory image processing. *Journal of Visual Languages & Computing*, *25*(5), 545–571.

[4] Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Publishers.

[5] Mărăşoiu, M., Blackwell, A., Sarkar, A., & Spott, M. (2016). Clarifying hypotheses by sketching data. In EuroVis 2016 - Short Papers.

[6] Satyanarayan, A., & Heer, J. (2014). Lyra: An Interactive Visualization Design Environment. Computer Graphics Forum: Journal of the European Association for Computer Graphics, 33(3), 351–360.

[7] Sharp, H., Dittrich, Y., & de Souza, C. (2016). The Role of Ethnographic Studies in Empirical Software Engineering. IEEE Transactions on Software Engineering, PP(99), 1–1.

[8] Victor, B. (2013). Drawing Dynamic Visualizations. Retrieved May 30, 2016, from https://vimeo.com/66085662