DataChopin
Designing Interactions for Visualisation Composition in a Co-located, Cooperative Environment

Daniel Filonik, Markus Rittenbruch, and Marcus Foth

Queensland University of Technology, Brisbane, Australia
{daniel.filonik,m.rittenbruch,m.foth}@qut.edu.au
https://www.qut.edu.au/

Abstract. This article presents our interaction design for *DataChopin*, based on an extensive survey and classification of visualisation for exploratory data analysis. Its distinctive characteristics are the use of a large-scale display wall as a shared desktop, as well as flexible composition mechanisms for incremental and piece-wise construction of visualisations. We chose composability as a guiding principle in our design, since it is essential to open-ended exploration, as well as collaborative analysis. For one, it enables truly exploratory inquiry by letting users freely examine different combinations of data, rather than offering a predetermined set of choices. Perhaps more importantly, it provides a foundation for data analysis through collaborative interaction with visualisations. If data and visualisations are composable, they can split into independent parts and recombined during the analytical process, allowing analysts to seamlessly transition between closely- and loosely-coupled work.

Keywords: Co-located Collaboration, Exploratory Data Analysis, Visualisation Composition

1 Introduction

The need for better understanding of abstract data is not new, although it is exacerbated by the growing ease and speed of data acquisition. When automatic methods fail, human background knowledge and intuition are required. Yet as Norman [13] points out, human cognitive abilities are highly constrained and our real ingenuity lies in the ability to devise external aids that enhance them. While our cognitive capabilities for storing and manipulating data may be limited, we have evolved to perform many analytical processing tasks visually. Therefore, a common approach is to devise visual representations of data. However, the choice of representation is not trivial and depends heavily on the task at hand. Consequently, a wide range of visualisation systems have been developed, designed specifically for certain tasks. While such systems with predefined components are typically effective in their intended area of application, they are often too limited

for open-ended, exploratory analysis, which requires the ability to manipulate and tailor representations based on emerging questions and insights. In order to address this, researchers have systematically studied the structure of graphical representations, along with the rules by which visualisations are constructed. Our proposed system continues this line of research, building on existing theoretical and formal models to arrive at a practical implementation and explore suitable interaction techniques and metaphors for cooperative visualisation specification.

A key promise of novel interaction technologies are better ways for people to work cooperatively. Concepts that were previously only explored in research – such as large-scale environments augmented with interactive capabilities – are becoming technologically feasible. We aim to utilise such technologies to create engaging experiences that put multiple analysts in a shared environment and elicit contextual knowledge from these analysts. The importance of contextual knowledge for the analytical process was already pointed out by Cleveland:

> "Conclusions spring from data when this information is combined with the prior knowledge of the subject under investigation." [6, p. 5]

When multiple analysts join efforts, and are given effective tools to share and communicate their visions, the potential for more diverse and unexpected insights stands to grow accordingly.

2 Related Work

As part of this review, we examine existing systems that allow flexible visualisation specification for exploratory data analysis. Since a multitude of such systems have been developed, we do not attempt to create an exhaustive survey. Instead, we aim to highlight conceptual differences based on notable examples. Furthermore, we identify and categorize the predominant interaction methods, highlighting their commonalities and differences. Since our goal is the design of an interface for ad-hoc end-user composition and collaboration, we place particular focus on how the different conceptual models relate to interface mechanisms and metaphors for visualisation construction.

In order to develop flexible systems for visualisation specification it is necessary to understand the integral components and structure of graphics. The systematic analysis graphical representations, lies at the core of an important set of visualisation theories, sometimes referred to as structural theories of graphics. Much of this research can be traced back to the *Semiology of Graphics* by Bertin [2], which represents one of the first attempts to interpret graphics as a language with formal rules. Another prevalent theoretical model is that of a *visualisation pipeline*, as popularised by Card et al. [5]. This model provides a description of the visualisation process as a sequence of transformation steps.

Depending on the chosen theoretical perspective, certain paradigms for composing visualisations naturally lend themselves. Perhaps the most obvious form of visualisation specification is an imperative algorithm that issues drawing commands to produce graphical primitives. Such specifications provide fine grained

control, but require familiarity with programming concepts, such as variables and control flow. However, the approach as its strengths, as evidenced by the popularity of *Processing* [7]. In contrast to imperative algorithms, more recent approaches employ a declarative paradigm, placing emphasis on what to display rather than how to produce it. Such a specification features a description of a graphical scene, and allows connecting data attributes to visual attributes of graphical elements. Libraries based on the *data binding* model are *Protovis* [3] and its successor *D3* [4]. On another end of the spectrum, researchers have created automated presentation tools eliminate the need for a specification entirely. Such research advanced the study of graphic primitives and pioneered key ideas, such as composition algebras for graphical marks [11,14]. Extending further on the ideas of structural theories of graphics, researchers have developed sophisticated graphics grammars [21,20], resulting in specification languages that allow the assembly of statistical graphics from fine-grained, modular units of composition. These ideas also provide the foundation for *Vega* [15], the low-level declarative language behind *Lyra* – as well as *VizQL* [8], the query-based language behind *Tableau*. Meanwhile, the prominence of the visualisation pipeline model has lead to the adoption of a data flow paradigm in many systems. The pipeline structure provides a blueprint for the implementation of reconfigurable visualisation components, such as those in *VTK* [16].

In addition to underlying conceptual models, there are different options for exposing them through graphical interfaces. These aspects are closely related, since a good interface effectively communicates the conceptual model to help users develop their mental model. The following classification aims to characterise the predominant interaction concepts in visualisation software today, including select examples beyond the domain of visualisation as guideposts for future developments.

Chart Typology Novice friendly programs are often limited to a predefined set of charts, often represented as a catalogue of icons in the graphical interface. However, chart typologies have been heavily criticised by researchers like Wilkinson, who claim that choosing from a limited charts gives "the user an impression of having explored data rather than the experience" [21, p. 2]. This appears problematic especially for exploratory analysis, as it offers no way for users to produce visual representations beyond those explicitly supported by the system.

Text and Preview A very basic form of graphical support is an environment where text-based specification is accompanied by a preview window. This approach is adopted by software like *Processing Development Environment* [7] and *GPL* [21]. The former employs an imperative programming style, whereas the latter is a proprietary implementation of Wilkinson's graphics grammar. A modified version of the grammar is publicly available in the form of the *ggplot2* [20] module for the R statistical computing environment. While text input is often challenging on touch interfaces, it can be assisted through auto-completion or direct manipulation of the parse tree.

Tokens and Slots More sophisticated graphical interfaces allow data binding via property sheets, or by dragging and dropping tokens on designated regions of the interface. Such a model is realised in software like *ILOG Discovery* [1], *Polaris* [17], and its successor *Tableau* [8]. These interfaces often incorporate abstract representations of the variables in a data set, which can be manipulated via drag-and-drop gestures. This commonly represents *data binding*, whereby variables are bound to properties of the visual representation. Furthermore, *Polaris* and *Tableau* also inherit ideas from *visualisation grammars*, such as a compositional algebra to specify combinations and nestings of data variables.

Boxes and Wires Another interaction concept are boxes and wires, also known as the node-link model of visual programming, which is a natural fit for the *data flow* model of visualisation. This approach combines a visual notation with expressive power of text-based specification languages. Complex flows are created by placing processing units that act as operators on the data, and subsequently connecting their inputs and outputs to create a graph. Such a model employed by *VTK* [16] in the form of *VTK Designer*.

Pipeline Stages Such interfaces are characterised by high-level abstractions focussed on the application domain. For example, they might be restricted to pipelines with a limited set of stages, which are directly represented within the interface as text or icons. This style is followed by the *Lark* [18] application, which uses a pipeline with customisation points at three stages: analytical abstraction, spatial layout, and presentation. Furthermore, *LIVE Singapore! Data Browser* [10] and *Datacollider* from MIT's SENSEable City Lab apply similar models. Outside of the visualisation domain, other notable interfaces using very specialised programming models are *Reactable* [9] for musical composition and *Kodu* [12] for specifying simple behaviours in games.

Drawing Canvas Generally, the interaction model of graphical applications is not a good fit for the task of visualisation specification, as manual manipulation of marks quickly becomes becomes repetitive and tedious. However, paired with facilities for automation and *data binding*, this interaction style can become feasible. In this respect, the web-based *Lyra* [15] application is worth mentioning, as it is inspired by Victor's interface for drawing dynamic visualisations [19]. The users create and arrange visual marks on the canvas through direct manipulation. Subsequently, data variables are dragged onto various anchors in order to bind data to visual properties.

3 Design Process

Based on our review, we assessed the identified interaction concepts for use co-located, collaborative settings. Ultimately, this process informed the design our final artefact, named *DataChopin*. The system was designed from the ground up for co-located, multi-user interactions. Therefore, data sets and visualisations are associated with user accounts. Once a user authenticates with the system, their presence is indicated by a top-level menu element on the shared desktop, which features an avatar and provides access to personal content.

Data Interactions The interaction metaphor for selecting and manipulating data attributes was inspired by poker chips. Our design intuition was that these elements would introduce interesting dynamics to a collaborative analysis process. Poker chips are employed in a variety of tabletop games, enabling playful mechanics and social interactions. They carry associations with collecting, exchanging, and negotiating. Our goal was to capture the affordances of poker chips, while enhancing their digital counterparts with capabilities for data analysis.

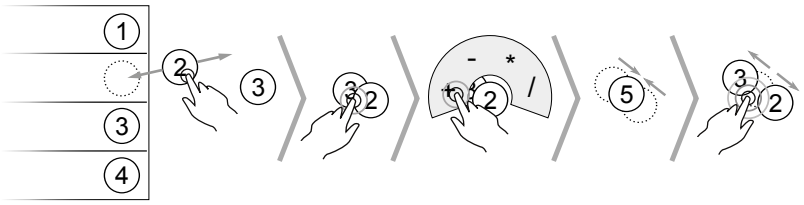

Fig. 1: Selecting and combining data attributes to compose expressions.

Piling tokens on top of each other provides a mechanism for mapping functions over the data, as illustrated in figure 1. Tapping on a pile of tokens opens a radial auto-completion menu of applicable functions based on the types of the tokens in the pile. If we limit ourselves to unary or binary functions, we can display the menu as soon as a user drops one token onto another, if there are applicable functions based on their types. If tokens originate from different data sets, they are considered to be incompatible. In this case we can introduce a repulsive force pushing the tokens apart to indicate that two tokens are incompatible. Once a function is applied, the token pile merges into a single token representing the composite expression. The action can be undone by repeated tapping on a composite token to recover the constituent parts.

View Interactions The primary interaction mechanism for creating and manipulating graphical marks is through direct interaction with the visualisation canvas, along with a drag-and-drop mechanism for data binding. Interactive elements in the drawing canvas are the *origin*, *axes*, *marks*, and *background*. They are animated to change colour and size as the user drags data tokens to indicate whenever meaningful actions are possible, based on the type of the expression that constitutes the dragged payload. Initially, the visualisation canvas is empty with only the origin visible. Placing a token on any of the interactive elements creates or modifies the mark layer associated with the data set. Dropping tokens on the *origin* spawns new coordinate axes, binding the data attribute to the respective positional component of the marks. Placing tokens on any of the *axes* creates or replaces the existing binding. Dropping on tokens on any of the *marks* binds the data attribute to a visual attribute of the mark as determined by the type. Finally, dropping tokens on the *background* prompts the system to automatically choose an appropriate mapping.

Cooperative Analysis The top-level interface elements adopt a classical window metaphor and can be freely positioned in the large-scale, shared-desktop environment. As different users have access to different data sets through their personal profiles, it introduces the need to exchange tokens and work together to achieve the desired visualisation results. The system allows users with different areas of interest and expertise to build personal repertoires of data, and subsequently share and divide responsibilities as they work together.

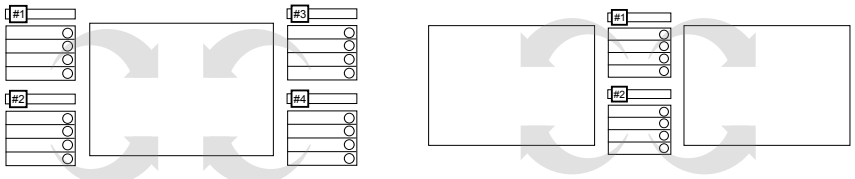

Fig. 2: Collaborative configurations based on arrangements of interface elements.

A common starting configuration is a single data repository and visualisation canvas for each user. This supports a loosely coupled style of cooperation, during which the users mostly work independently on their own canvas. Nevertheless, due to the nature of the shared desktop, even such configurations frequently bring about interactions, such as users glancing over to see other visualisations and asking for instructions. Often times, once users recreate somebody else's visualisation with their own data, they are eager to compare the results.

Another configuration that we explored was a single, shared drawing canvas that multiple users gathered around, as illustrated by figure 2a. This configurations supports a closely coupled style of cooperation, and is either based on a single data of shared interest, or on multiple data sets from the repositories of different users, resulting in a combined, layered visualisation. The latter is made possible due to composability being an integral feature of the system. Finally, as a single view can lead to contention, we also experimented with compromises between a single, shared and multiple, independent views. One configuration that appears promising are two drawing canvases, with a number of shared data repositories in between them, depicted in figure 2b. That way, two groups can work independently, both having access to the same data sets. If the data used by both groups comes from the same data sets, the visualisations are always compatible, meaning that the groups can interchange and combine parts of their visualisation specifications at any given time.

4 Discussion

Our work lead us to survey the spectrum of conceptual models and interaction concepts for visualisation specification, weighing their associated trade-offs in the specific context of exploratory analysis in collaborative environments. Early

on, we ruled out static chart typologies, due to concerns that their rigidity would stifle creativity. Furthermore, piece-wise and iterative specification is considered beneficial for open-ended analysis, and forms an important cornerstone for collaboration in our proposed system. Text-based specifications have proven effective, especially for seasoned users who are familiar with the syntax and semantics of the underlying specification language. However, they are in conflict with direct manipulation principles and pose challenges with regard to text input on touch-based interfaces. While boxes and wires provide a visual notation capable of modelling general-purpose programming languages, their generality comes at the cost of usability. In our experience, we found the domain of exploratory analysis sufficiently constrained to employ a special-purpose abstraction. In our classification, *DataChopin* is a hybrid of tokens and slots, combined with a visualisation canvas for declarative data binding.

So far, we have conducted informal evaluations our system, and our initial experiences have been positive. Cooperating in a co-located setting successfully elicited discussions about the data and participants were quick to share their interpretations. The use of a multi-user, shared-desktop environment was commonly regarded as beneficial. In contrast to the single-user, personal systems that participants were accustomed to, the idea of multiple analysts working in tandem was perceived as empowering. Rather than a single person being in charge and driving the analytical process, the interface enabled them to perform actions in parallel and pursue smaller tasks independently. Therefore, we continue to focus our efforts on placing participants in shared interaction environments, aiming to leverage the implicit and explicit communication channels to stimulate creativity and assist analysis. In future work, we are planning more formal evaluations to assess the expressiveness and effectiveness of the proposed compositional model.

5 Conclusion

Our review has shown that HCI research on cooperative visualisation specification is still lacking. While some systems support distributed, asynchronous collaboration, few focus on co-located, synchronous settings. With the exception of *Lark*, the majority of existing interfaces were designed for single-user, personal environments. This article represents another step towards closing the research gap. We have classified predominant interaction methods for visualisation specification, and derived a design specifically aimed at facilitating cooperation in a shared interaction environment. The result is *DataChopin*, a system for large-scale, shared-desktop environments, based on the premise of composable visualisations. Often, formal visualisation models have been studied in theory and divorced from HCI considerations. In contrast to that, our work presents a practical approach, covering the design and implementation of a working prototype.

Acknowledgments. We would like to thank the Visualisation and eResearch (ViseR) team at QUT for their technical support. Illustrations in this article contain icons by GestureWorks, released under CC BY-SA.

References

1. Baudel, T.: A canonical representation of data-linear visualization algorithms. Tech. rep., IBM, Software Group, ILOG products. (2010)
2. Bertin, J.: Semiology of Graphics: Diagrams, Networks, Maps (Translated by Berg, W.J.). The University of Wisconsin Press, Madison, WI, USA (1983)
3. Bostock, M., Heer, J.: Protovis: A graphical toolkit for visualization. Visualization and Computer Graphics, IEEE Transactions on 15(6), 1121–1128 (2009)
4. Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. Visualization and Computer Graphics, IEEE Transactions on 17(12), 2301–2309 (2011)
5. Card, S.K., Mackinlay, J.D., Shneiderman, B.: Readings in information visualization: using vision to think. Morgan Kaufmann (1999)
6. Cleveland, W.S.: Visualizing data. Hobart Press (1993)
7. Fry, B.: Visualizing Data: Exploring and Explaining Data with the Processing Environment. O'Reilly Media (2007)
8. Heer, J., Mackinlay, J.D., Stolte, C., Agrawala, M.: Graphical histories for visualization: Supporting analysis, communication, and evaluation. Visualization and Computer Graphics, IEEE Transactions on 14(6), 1189–1196 (2008)
9. Jordà, S., Kaltenbrunner, M., Geiger, G., Alonso, M.: The reacTable: a tangible tabletop musical instrument and collaborative workbench. In: ACM SIGGRAPH 2006 Sketches. p. 91. ACM (2006)
10. Kloeckl, K., Senn, O., Ratti, C.: Enabling the real-time city: Live singapore! Journal of Urban Technology 19(2), 89–112 (2012)
11. Mackinlay, J.D.: Automating the design of graphical presentations of relational information. ACM Transactions on Graphics (TOG) 5(2), 110–141 (1986)
12. MacLaurin, M.B.: The design of kodu: a tiny visual programming language for children on the xbox 360. In: ACM Sigplan Notices. vol. 46, pp. 241–246. ACM (2011)
13. Norman, D.A.: Things that make us smart: Defending human attributes in the age of the machine. Basic Books (1993)
14. Roth, S.F., Mattis, J.: Automating the presentation of information. In: Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications. vol. 1, pp. 90–97. IEEE (1991)
15. Satyanarayan, A., Heer, J.: Lyra: An interactive visualization design environment. Computer Graphics Forum (Proc. EuroVis) (2014)
16. Schroeder, W.J., Lorenson, B.: Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn. (1996)
17. Stolte, C., Tang, D., Hanrahan, P.: Polaris: A system for query, analysis, and visualization of multidimensional relational databases. Visualization and Computer Graphics, IEEE Transactions on 8(1), 52–65 (2002)
18. Tobiasz, M., Isenberg, P., Carpendale, S.: Lark: Coordinating co-located collaboration with information visualization. Visualization and Computer Graphics, IEEE Transactions on 15(6), 1065–1072 (2009)
19. Victor, B.: Drawing dynamic visualizations. `http://vimeo.com/66085662`, accessed: 2014-08-31
20. Wickham, H.: A layered grammar of graphics. Journal of Computational and Graphical Statistics 19(1), 3–28 (2010)
21. Wilkinson, L.: The grammar of graphics. Springer (2005)