Graph Sketcher: Extending Illustration to Quantitative Graphs

Robin Stewart*
MIT CSAIL
32 Vassar Street,
Cambridge, MA 02139, USA
rstew@alum.mit.edu

m.c. schraefel
Intelligence, Agents, Multimedia Group
University of Southampton
Highfield, Southampton SO17 1BJ, UK
mc+chi09@ecs.soton.ac.uk

ABSTRACT
Scientists, engineers, and educators commonly need to make graphs that quickly illustrate quantitative ideas yet are not based on specific data sets. We call these graphs quantitative concept diagrams (QCDs). Existing charting and illustration programs make it possible to produce such graphs, but they are so time-consuming that users tend to sketch the graphs by hand instead. To reduce the cost of creating QCDs, we developed Graph Sketcher, a quantitative graphing tool that deeply integrates the data plotting capabilities of charting programs with the direct manipulation techniques of illustration programs. We show that our integrated interface substantially reduces the time needed to create QCDs, and we further show that real Graph Sketcher users both enjoy and take advantage of the interface improvements to create QCDs in a wide range of fields.

Author Keywords
Information visualization, charting, illustration, quantitative concept diagrams, snap-dragging, constraint-based layout, planar map coloring.

ACM Classification Keywords
H5.2 Information interfaces and presentation: User Interfaces—*Graphical user interfaces.*

INTRODUCTION
Existing graph-making interfaces poorly support an important class of diagrams that are quantitative yet not based on specific data sets. Consider the lines and filled areas in Figure 1. These components are *quantitative* because their meaning depends on the scaled coordinate system of the axes. Yet they are also *"conceptual"* because they represent a simplified economics theory rather than raw data. The quantitative nature of these components demands charting functionality, while the conceptual nature demands the visual, direct-manipulation interface of illustration programs. We call diagrams containing such components *quantitative concept diagrams* (QCDs). Figure 2 situates QCDs within the larger space of diagrams.

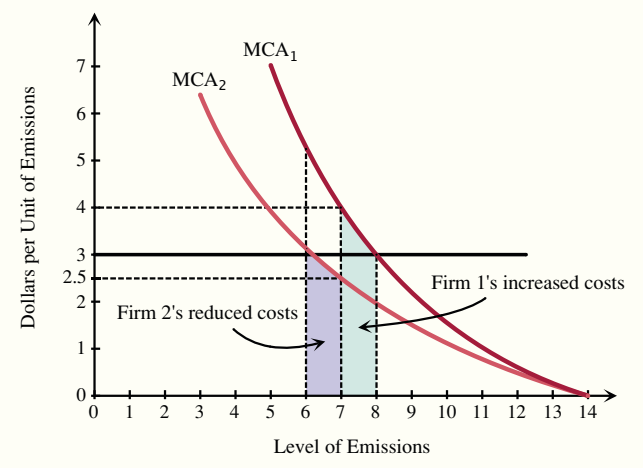

Figure 1: An Economics diagram [6] with components that are simultaneously quantitative and conceptual.

Our studies of the curriculum in Economics and other fields have found that QCDs such as Figure 1 are critical for learning quantitative concepts — yet they are so frustrating to create with existing programs that most users revert to sketching them with paper and pen, foregoing the higher-quality results and ease of editing and sharing that digital media provides. Graph Sketcher aims to make computer-based QCDs dramatically easier to create, so that their use can be expanded from formal publications into more everyday mediums such as lecture slides, essays, wikis, and blogs.

Graph Sketcher's interface goes beyond previous work by deeply integrating interface techniques that have previously been kept separated. In existing graph-making software, a given component is either an illustration component controlled via direct-manipulation, or a data object controlled via the charting system, but not both. This works fine when plotting data sets or illustrating non-quantitative ideas, but for QCDs it forces users to make a tradeoff between fast, direct manipulation and precise quantitative positioning. Graph Sketcher instead unifies data plotting capabilities and direct manipulation capabilities so that all diagram components be-

CHI 2009, April 4 - 9, 2009, Boston, MA, USA.

*Now at The Omni Group, 3257 16th Ave W, Seattle, WA 98119.

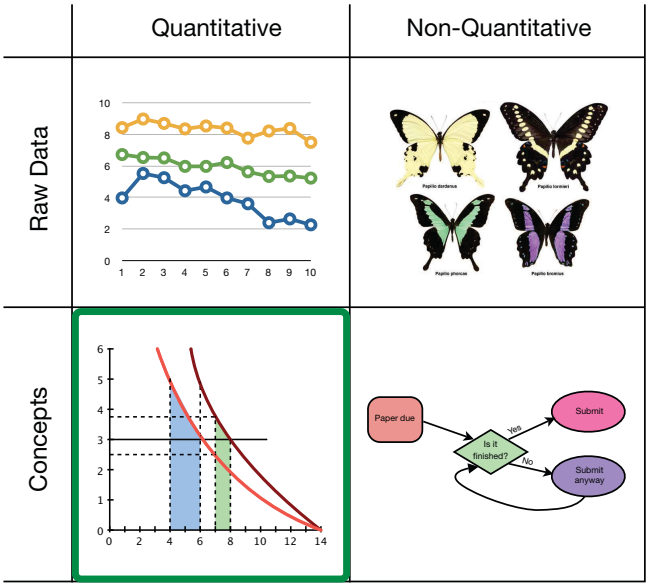

Figure 2: A taxonomy of diagrams compares QCDs to the charts and illustrations supported by existing tools.

have as both data *and* illustration. This novel interface integrates and extends snapping [3], constraint-based layout [4], and planar map coloring [1] so that users can create QCDs as easily as they can create plots and illustrations.

We further motivate the design of our tool by documenting the need for QCDs in Economics. Next, we describe how Graph Sketcher's interface extends existing interfaces, and we calculate the corresponding improvements in user efficiency. Finally, we summarize survey results showing that these improvements have allowed real users of Graph Sketcher to more easily make QCDs in a wide range of fields.

MOTIVATION

Quantitative concept diagrams are used to teach almost every quantitative discipline, including chemistry, biology, physics, math, and engineering. For example, our analysis of a typical introductory Economics textbook [6] found that 19% of its pages contain QCDs. In the 178 diagrams with axes, quantitative concepts were represented using straight lines in 79% of graphs, curved lines in 21%, and filled areas in 15%. (Figure 1 includes all three of these representation types.)

To validate our intution that existing interfaces are insufficient for creating QCDs, we asked 128 Economics professors whether their students turn in computer-generated diagrams on their problem sets or simply draw them by hand. The majority of the 19 professors who responded estimated that fewer than 30% of students use a computer to draw their Economics diagrams (Figure 3). By contrast, most college students voluntarily use word processors to complete verbal assignments, because it is faster, easier to read, and looks more professional than hand writing. For the same reasons, a much larger proportion of students would voluntarily make QCDs by computer if it actually saved them time.

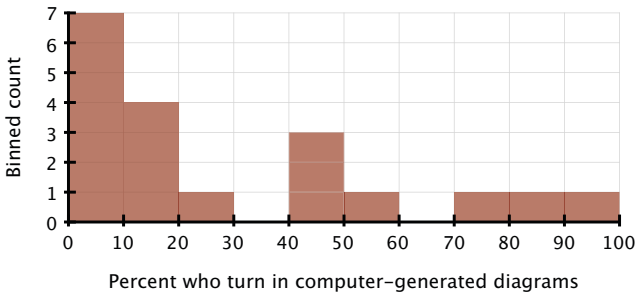

Figure 3: Histogram summarizing 19 Economics professors' estimates of the percentage of their students who use a computer to make diagrams required for problem sets.

RELATED WORK

Related research has sought to bridge the divide between charting and illustration interfaces in various specific cases. Baudel [2] supports direct manipulation of plotted data so that users can visually update their data and fix outliers. Fathom [7] lets users directly manipulate straight line functions on a graph in order to quickly see statistical implications. Our interface continues to receive the benefits of these limited approaches while going much further to support the efficient creation of QCDs.

Constraint-based layout has been extended in several relevant directions. Briar's [4] snap-based constraints could handle a wide range of geometric properties beyond line and intersection coincidence, and Igarashi [5] extended these constraints and others to work with a pen interface. The general approach has also been applied to laying out graphs of the data-structure and org-chart variety [8]. Graph Sketcher is the first tool to integrate these techniques with charting capabilities.

THE INTERFACE

We model a QCD as a set of points, lines, polygons, and labels mapped into a user-defined 2D coordinate space. Lines are implemented as a series of points connected by straight or curved segments, and polygons ("filled areas") are similarly defined by a set of points acting as corners. We provide a fairly standard vector illustration interface with a pen-like tool for drawing lines, a polygon tool for creating filled areas, and an editing tool for selecting and repositioning already-drawn objects.

To help users modify their diagrams more quickly, we also use a simple constraint-based layout system that applies to points snapped to lines or line intersections (an approach pioneered in Briar [4]). Specifically, we treat each line as a parametric function $p(t)$ and record the t-value at the location where a point has been snapped. Whenever the line is moved or reshaped, the system updates the position of the snap-constrained point by recalculating $p(t)$. Because lines and polygons are defined by a set of points, this constraint system applies equivalently to maintaining line endpoints and fill corners. For example, it would be activated in all of the connections between lines and fill corners in Figure 1.

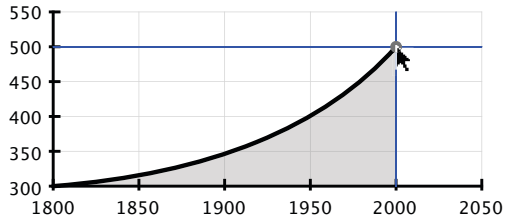

Figure 4: Our interface helps users snap line endpoints and fill corners to the user-defined data coordinate grid.

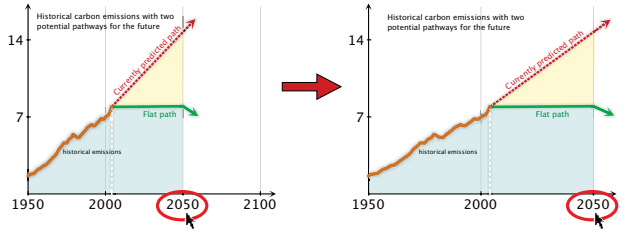

Figure 5: The charting system maintains the positions of all objects within the user-defined coordinate space.

The first way we improve upon the basic charting and illustration functionality is by **augmenting the "snapping" behavior with awareness of the user-defined coordinate grid** (Figure 4). Snapping reduces the time a user needs to precisely position the cursor by effectively increasing the size of the target (e.g., [3]). Traditional snap-to-grid interfaces are designed to help align objects on a page. Our interface differs by (a) allowing users to directly define the coordinate grid (based on the charting system), and (b) snapping the individual line endpoints and fill corners rather than whole objects (a more natural behavior in the context of quantitative diagrams). Snapping in this way makes it approximately twice as fast[1] to position most of the components in Figure 1.

The second way we unify charting and illustration functionality is by **maintaining the positions of all lines and filled areas within the user-defined coordinate space** (Figure 5). To accomplish this, we use a hybrid constraint system. By default, every point — and thus every line endpoint and fill corner — is mapped to a location in the user-defined coordinate space and maintained by the charting system. If the user snaps a point to a line or intersection, control over its position goes instead to the constraint-based layout system (described above). There is in fact no notion of a point that is *not* subject to one of these two constraint systems — but the user still maintains full control because of the ability to reposition any point by simply dragging it with the cursor.

The underlying charting system preserves the quantitative meaning of diagram components. Without it — as is the case in existing programs — whenever the user modifies the axis ranges or resizes the canvas, any illustration components just stay where they were. This is surprising for users whose mental model is of a quantitative space, and worse, it means that updating the axis ranges also requires tediously repositioning each illustration component. Even for a simple economics diagram such as Figure 1, our heuristic analyses estimate that such an update takes users 4-10 times longer in existing programs than in our interface, even if the user has access to advanced grouping and resizing features.[2]

[1]Fitt's law predicts that pointing time increases logarithmically as the target size decreases; the default target size of 2 pixels unsnapped vs. 12 pixels snapped predicts an increase of $\sim 1.8x$.

[2]We used a keystroke-level model to estimate the relative time required in Graph Sketcher (2 steps), Adobe Illustrator CS3 (8 steps), Apple Numbers 2008 (17 steps), and Microsoft Excel 2008 (23 steps).

Third, **we extend planar map illustration concepts to plotted data series**. Planar map illustration techniques color the areas between lines instead of using the traditional approach of "stacking" layers of shapes [1]. Again, we use a hybrid approach. By default, filled polygons stack traditionally. But if two successive fill corners are snapped onto a line or data series, the edge of the colored area automatically follows the line as it would in planar map illustration (Figure 6). This makes it dramatically faster to create numerically precise filled areas that depend on other quantitative concepts in a diagram.

IN-USE STUDY

Graph Sketcher is available online as shareware[3] and has been purchased by over a thousand students, teachers, professionals, and hobbyists. We surveyed the 300 users who most recently bought a license, requesting their most recent graph and asking why they used Graph Sketcher to make it. We received 31 responses (10%). 22 of these graphs (71%) included quantitative conceptual components. These 22 came from nine fields: economics (10), psychology (3), math (2), biophysics, chemistry, civil engineering, IT administration, mechanical engineering, and physics.

Written survey responses gave us a broad but shallow understanding of why users prefer Graph Sketcher to existing tools. Most respondents said Graph Sketcher was simply "easier to use," "faster," and "more intuitive" than other programs, and a few spoke directly to the importance of inte-

[3]http://www.graphsketcher.com

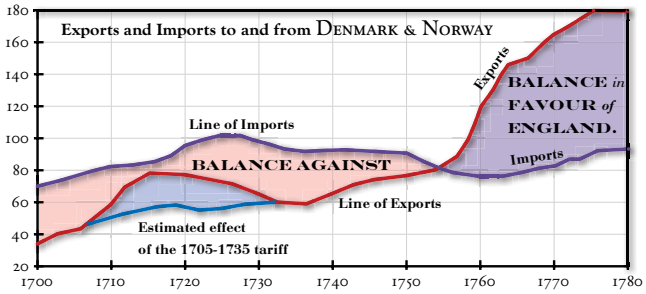

Figure 6: Our extension of planar map illustration makes it much easier to create filled areas bordered by data series. William Playfair shaded such areas by hand in many of his pioneering quantitative diagrams.

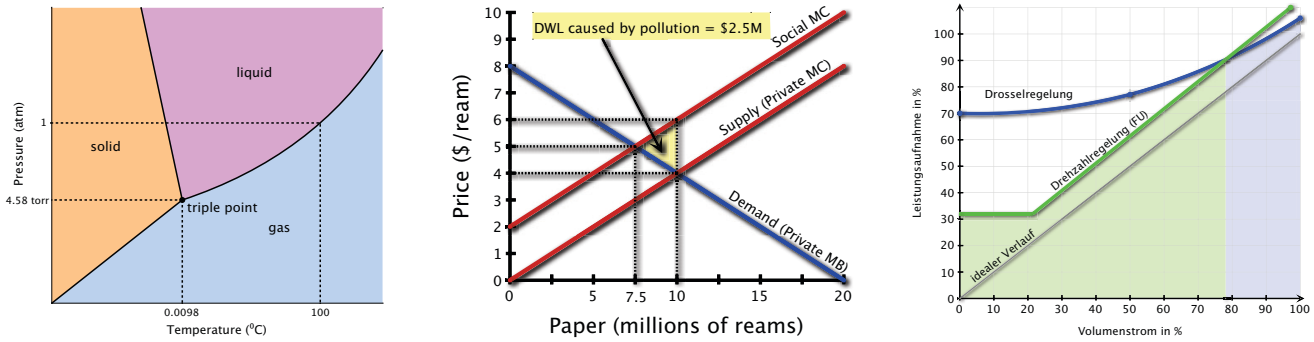

Figure 7: Diagrams created by Graph Sketcher users in chemistry, economics, and mechanical engineering.

grating charting and illustration functionality. For example:

> The only other programs I am familiar with that would make graphs like this are excel, which doesn't allow you to draw stuff, and any drawing program, which are hard to get the scales right. This is the best of both worlds.

We analyzed the 22 QCDs (some of which appear in Figure 7) to evaluate whether the novel interface features presented in this paper specifically contributed to the positive impression. Line endpoints were snapped to the user-defined coordinate system in 73% of the graphs, indicating that snapping to the scaled grid provides the largest benefit of the techniques we have presented. Without watching the evolution of the graphs over time, it is difficult to evaluate the usefulness of maintaining all positions within coordinate space. However, 64% of the graphs potentially benefitted from this because they used a scaled coordinate system. Finally, 32% of graphs used filled areas snapped to lines or data series, showing that our extension of planar map coloring is useful in the cases that require it (recall that only 15% of the economics textbook QCDs contained filled areas).

Finally, the fact that more than a third of the diagrams did *not* use Graph Sketcher's novel interface features suggests that the presence of these features does not detract from making non-QCD charts (otherwise, users would have continued to make these charts with traditional programs).

CONCLUSION AND FUTURE WORK

What distinguishes Graph Sketcher from preexisting work is the deep integration of interface techniques previously separated into charting and illustration feature sets. Using a hybrid constraint system and extensions to snap-dragging and planar map coloring, our interface treats all components as both data *and* illustration, so that whichever properties are most important at a given time can be used.

This approach can be extended to support more geometric constraints [4, 5] and more intelligent planar map coloring [1]. It can also be applied to any quantitative coordinate space, including logarithmic scales, time scales, and polar representations. And we would like to support more charting features such as mathematical functions, bar charts, and points whose size and color are also based on scaled data.

The techniques presented here will become increasingly important over time as computer-supported collaborative work demands that visualizations are computerized, shared, and contain an increasing number of inline annotations. With the ability to efficiently create quantitative concept diagrams, users can substantially improve the presentation and comprehension of a wide range of quantitative ideas.

ACKNOWLEDGEMENTS

Many thanks to Evan Miller, Michael Bernstein, Max Van Kleek, and the many other colleagues, friends, teachers, advisors, beta testers, and users who provided valuable feedback and support.

REFERENCES

1. P. Asente, M. Schuster, and T. Pettit. Dynamic planar map illustration. *ACM Trans. Graph.*, 26(3):30, 2007.
2. T. Baudel. From information visualization to direct manipulation: extending a generic visualization framework for the interactive editing of large datasets. In *Proc. UIST '06*, pages 67–76. ACM, 2006.
3. E. A. Bier and M. C. Stone. Snap-dragging. *SIGGRAPH Comput. Graph.*, 20(4):233–240, 1986.
4. M. Gleicher and A. Witkin. Drawing with constraints. *The Visual Computer*, 11(1):39–51, 1994.
5. T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: a technique for rapid geometric design. In *Proc. UIST '97*, pages 105–114. ACM, 1997.
6. R. S. Pindyck and D. L. Rubinfeld. *Microeconomics*. Prentice Hall, Upper Saddle River, NJ, 2001.
7. K. C. Press. Fathom dynamic data software 2.1. Desktop software, 2008.
8. K. Ryall, J. Marks, and S. Shieber. An interactive constraint-based system for drawing graphs. In *Proc. UIST '97*, pages 97–104. ACM, 1997.