TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs

Dominikus Baur[1], Bongshin Lee[2], Sheelagh Carpendale[1]

[1]Innovis Group, University of Calgary
2500 University Dr NW, Calgary, AB, Canada
dominikus.baur@gmail.com, sheelagh@ucalgary.ca

[2]Microsoft Research
One Microsoft Way, Redmond, WA, USA
bongshin@microsoft.com

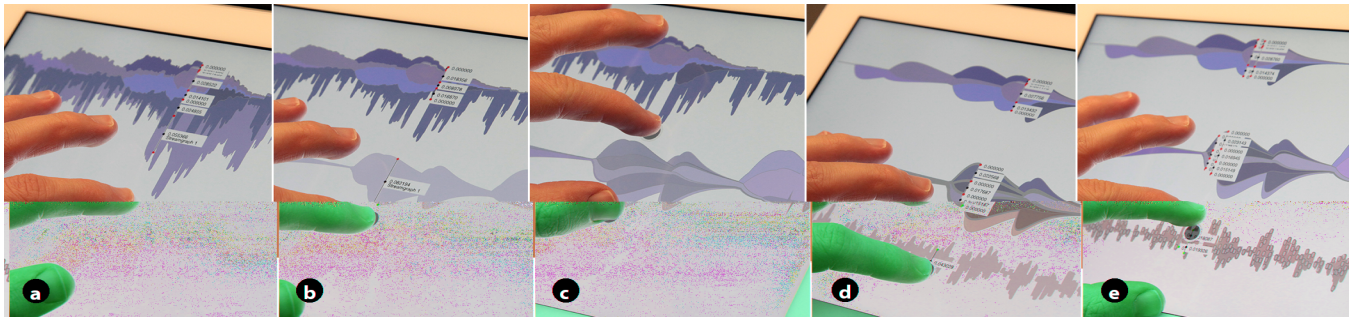

Figure 1: TouchWave: (a) reading specific values with the vertical ruler, (b) extracting a layer from its parent by dragging, (c) showing sublayers by pinching, (d) extracting a layer containing atomic items, and (e) pinching to show the underlying items.

ABSTRACT

The increasing popularity of touch-based devices is driving us to rethink existing interfaces. Within this opportunity, the complexity of information visualizations offers particular challenges. We explore these challenges to bring multi-touch interactions to a specific visualization technique, stacked graphs. Stacked graphs are a visually appealing and popular method for presenting time series data, however, they come with associated problems—issues with legibility, difficulties with comparisons, and restrictions in scalability. We present TouchWave, a rethinking and extension of stacked graphs for multi-touch capable devices that provides a variety of flexible layout adjustments, interactive options for querying data values, and seamlessly switching between different visualizations. In addition to ameliorating the main issues of stacked graphs, TouchWave also integrates hierarchical data within stacked graphs. We demonstrate TouchWave capabilities with two datasets—a music listening history and movie box office revenues—and discuss the implications for weaning other visualizations off mouse and keyboard.

Author Keywords

Stacked graphs; Visualization; Multi-touch; tablets.

ACM Classification: H.5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

ITS'12, November 11–14, 2012, Cambridge, Massachusetts, USA.

INTRODUCTION

In the last few years, multi-touch has become the default mode of interaction not only for portable devices such as smartphones and tablets, but also for interactive tabletops and wall displays. Changes are also expected to happen with desktops as touch-enabled monitors become more affordable and readily available. Controlling everything with fingers instead of a single, indirect mouse cursor drastically changes the requirements and capabilities of interfaces. These changes present both opportunities and challenges for information visualization, where interactions are still predominately mouse and keyboard; we need to rethink those interfaces. Given the wide variety and complexity of existing visualization techniques, it is probably not feasible to create a single set of multi-touch interactions that can cover all visualizations. To consider generalizing touch interactions across visualizations, we therefore need more concrete practical examples. In this paper, we add to the small set of multi-touch enabled visualizations [6,8,13,18,22,23] by extending one specific visualization technique, stacked graphs, which is commonly used and yet could significantly benefit from multi-touch interactions.

Despite their popularity, stacked graphs suffer from issues with legibility, enabling comparisons, and scalability. With the design and development of TouchWave (Figure 1), we show how bringing stacked graphs to multi-touch capable tablet computers to address their inherent issues requires a complete rethinking of the visualization, in both representation and interaction. In addition, we demonstrate how we address stacked graph issues with a simple yet comprehensive set of touch-based interactions.

The main contributions of this paper are: TouchWave, which extends capabilities of stacked graphs by introducing a set of consistent multi-touch interactions that mitigate several basic stacked graph problems; a collection of touch interaction design goals, which consider particularly the issues of creating a hands-on-data visualization experience; and the concept of *kinetic manipulation*, which encourages combining the active physical finger, hand, and body motions with the animated responses of the visualization to enhance an experience of virtual tangibility.

STACKED GRAPHS AND THEIR PROBLEMS

Stacked graphs are a popular method for visualizing socially relevant time-changing data mostly due to their organic and flowing visual qualities. Examples include news stories [11], music listening histories and movie box office rates [1], email messages [10], twitter postings [5], and names people give their babies [24]. Stacked graphs are composed of several *layers* of time series data on one common timeline (Figure 2). Each layer stands for one specific attribute of the data (e.g., the number of tweets for a given keyword or the stock price of one specific share), providing its value at a given point in time. Values are encoded as the height of a layer at a given horizontal point. Compared to other types of time series visualization such as line charts or small multiples [20], stacked graphs form one cohesive chart: all layers are stacked on top of each other without intermediary whitespace. The outer hull of a stacked graph therefore depicts the sum of the values of all contained layers. The layout of a stacked graph depends on the form of its *baseline*: a straight baseline leads to a regular stacked graph, but its shape can also be adjusted to the data at each position. This leads to layouts such as symmetrical Themerivers or Streamgraphs that optimize the variation in slope based on a layer's thickness [1]. Stacked graphs are popular, but their approach to representing the data faces three types of problems: (1) *legibility,* (2) *comparisons*, and (3) *scalability* (Figure 3).

The *legibility* of a graph describes the ease with which the human visual system is able to extract values from it and whether the graphical representation exceeds a human perceptual threshold [16]. Visualizations are based on turning numerical data into visual shapes and legibility issues can arise in the process when choosing a problematic or unsuitable chart type [20]. Stacked graphs in their non-interactive form suffer from the fact that human perception makes reading and comparing curved slopes difficult [2] (Figure 3a). Also, stacked graphs are usually based on discrete data samples, but their representation suggests that the data was continuous. Finally, stacked graphs often lack explicit scales; they might show a horizontal time axis but the vertical axis is usually missing. The reason is that explicit scales would have to be visible at every point of the graph, which would lead to overlap and visual clutter.

Comparing values is common in information visualization scenarios but problematic with stacked graphs. While they

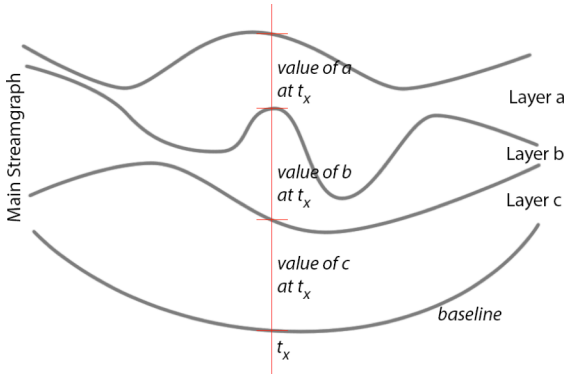

Figure 2: The structure of a stacked graph (here: a streamgraph). Each graph is made up of the sum of the values of its sublayers.

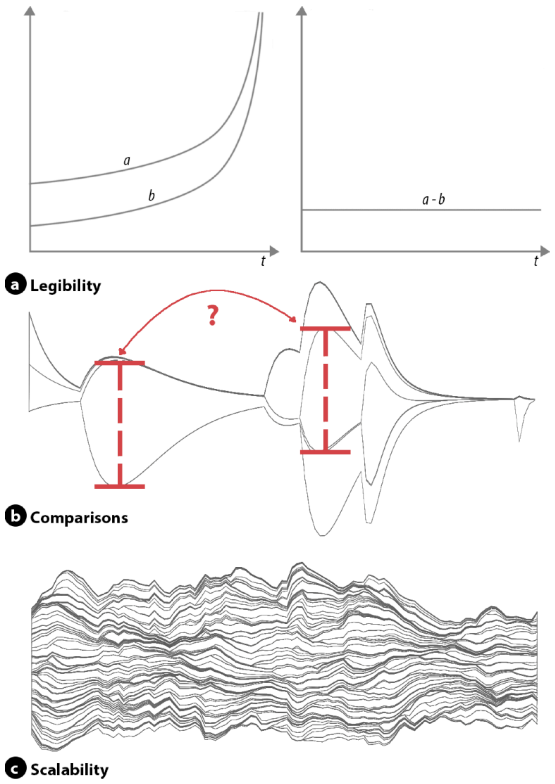

Figure 3: Three main issues of stacked graphs.

provide a good overall "feeling" for the growing and shrinking of layers over time, explicitly comparing layers is difficult (Figure 3b) and practically impossible if all layers have roughly the same size. Common comparison tasks that are difficult with stacked graphs include comparisons between layers overall, between different parts of one layer, between different parts of different layers and even between all layers at one point in time.

In common with many visualizations, stacked graphs also have *scalability* issues. Simply stacking all layers means that only a certain number of them can be visible and stay interactive at one time for a given screen resolution. Tens

or hundreds of layers quickly lead to visual clutter and makes interacting with layers nearly impossible (Figure 3c). Current solutions rely on simple panning and zooming, or hiding some layers; filtering by text query (hiding layers based on their attributes in NameVoyager [24]) or filtering by additional control widgets (BookVoyager's separate tree-control can show and hide specific layers [25]).

Only a few projects (e.g., BookVoyager [25], ManyEyes [21]) incorporate hierarchical data into a stacked graph to address scalability. Those that do support hierarchical data rely on separate (e.g., tree) views to show the hierarchy. This use of two views not only uses additional screen space but also requires the viewer to make the cognitive links between the two representations of the same data.

RELATED WORK

Interaction in Stacked Graphs and Streamgraphs

The challenges of stacked graphs caused even early projects to incorporate interaction with the visualization. ThemeRiver [11] shows the flow of document themes on a timeline and enables displaying and hiding labels, scales, and underlying numbers, and navigating the stream by zooming and panning. NameVoyager [24], an interactive stacked graph showing the popularity of baby names, enables rapid exploration of more than 6,000 time series based on prefix text search. Its filtering capability combined with smooth animations and aesthetically pleasing visual representations led to a popular response from the general public. BookVoyager extends NameVoyager to handle hierarchical time series by integrating it with an additional standard tree control used to navigate the hierarchy of the data [25]. Muse [10] provides similar filtering capabilities through additional on-screen elements for navigating email archives. Color-coding was used to make the hierarchical structure more visible in the stacked graph of ManyEyes [21]. Cui et al. merge time-series visualization with topic analysis in TextFlow [4] and create interactive and animated ways to explore this information. However, they emphasize topic development over time and relax some of the constraints of stacked graphs (e.g., having no whitespace between layers). ColourVis [15] provides considerable functionality but does not consider interaction. Finally, while sense.us [12] does not suggest new ways to directly improve on stacked graphs, it supports the social aspects of asynchronous collaborative visualizations.

Looking at the types of tasks supported by interaction with stacked graphs, they clearly focus on details-on-demand and filtering: based on Yi et al.'s taxonomy of seven high-level interaction goals in information visualization [26], all visualizations support such *abstraction/elaboration* (mostly by hovering over an item [4,11,12,21,24,25] or clicking [10]) and *filtering* (through text input [12,24,25] or clicking [4,12,21]). Yet, other tasks such as *exploration* (i.e., showing different subsets of the data, e.g., through pan & zoom [11]) or *reconfiguration* (i.e., showing the same data in a different layout [25]) are only rarely supported. Enabling similarly complex manipulations in one consistent widget-less set of simple touch-based interactions requires a different approach with a stronger focus on *reconfiguration* to keep the visualization object consistent.

Multi-touch Interactions for Information Visualization

With the rapid advances in technologies, some research projects are specifically exploring multi-touch interactions for information visualization. For example, Voida et al. present two interaction techniques, *i-Loupe* and *iPodLoupe*, and a set of design considerations to address the challenges of designing interaction techniques for information visualizations on tabletops [23]. Valming et al. explore tabletop touch interaction for 3D information visualization [22]. Spindler et al. explore the implications of a secondary, passive display device above a tabletop surface for information visualization [19] without extending touch-based manipulations.

There also have been several efforts to provide more fluid interactions for node-link graph visualizations. For example, Schmidt et al. focus on multi-touch interactions for link manipulation [18]. They broaden interaction possibilities by presenting a set of multi-touch interaction techniques (e.g., plucking, pinning, strumming, and bundling) that can be effectively combined. Frisch et al. discuss a rich set of individually-elicited pen and touch gestures for editing node-link diagrams, and provide insights into the suitability of gestures and bimanual interactions on tabletops [8]. Dwyer et al. investigate interaction techniques people use while they are optimizing node-link graph layouts [6]. Multi-touch interactions are also used to provide high freedom of expression when entering queries. For example, Facet-Streams harness the expressive power of facets and Boolean logic with tangible and multi-touch interactions without requiring people to use complex formal notations [14]. More generally, Isenberg et al. discuss how multi-touch interactions could be applied to visualizations [13].

We extend this research direction by demonstrating with TouchWave how bringing multi-touch to a visualization can go beyond mapping touches to cursor input, and start to reveal how touch interactions can improve the visualization.

TOUCHWAVE

TouchWave provides touchable stacked graphs that offer a set of integrated interaction techniques and the ability to display hierarchical data within streams. In this section, we first describe the underlying design considerations, followed by TouchWave's interaction techniques and then explain its hierarchical capabilities.

Design Goals

This section contains our main design goals for TouchWave: creating a full interaction set with kinetic manipulations and integrated interaction but without complex gestures and on-screen widgets.

Support Kinetic Manipulation

Touch-based interaction invites people to interact directly with the on-screen visuals. The idea behind *kinetic manipulation* is that a person's fingers, hands, and perhaps arms and body together with the visualization form a coherent kinetic whole. Kinetic manipulation suggests that a visualization reacts in a consistent, learnable manner to touch interaction. This consistency is usually arrived at via physically-based metaphors, but could also use a type of virtual consistency as in the concept of alternate interface physics [17]. In any case, kinetic manipulation enables exploration and learning the "rules" to which a visualization obeys by touching and dragging parts of it. The visualization becomes a tangible, virtual object that enables data exploration through its manipulation.

Kinetic manipulation also works with the concept of momentum, harmoniously linking physical touch movement to visualization movement. In this regard, kinetic manipulation follows similar ideas as the increasingly popular concept of fluid interaction [7] by avoiding abrupt switches between states and developing animated transitions. Kinetic manipulation goes further, however, in that it more strongly emphasizes the stability and existence of virtual objects. The intention here is to have consistent and active behavior and predictable malleability in touch-based adjustments.

Create Integrated Interactions

Integrated interactions mean that interactions are triggered directly on the visualization itself instead of a control panel, etc. Visualizations have had a long history of being manipulated in software where the visual representations and the interaction controls are spatially separate. Mapping these touch interactions directly to parts of the visualizations and creating a visual response that is expected is non-trivial. The intention with integrated interactions is to keep the interactions located within the visualization's screen space.

Another way interactions have been developed for visualizations is through the use of additional on-screen widgets. While this can be a powerful idea, the use of additional widgets can pull cognitive attention away from the visualization. In supporting the idea of integrated interactions, we work towards both limiting the use of additional widgets and making those that are used adhere with the kinetic manipulation concept. That is, they should be spatially situated appropriately within the visualization and their actions and reactions should be kinetically in harmony with those of the visualization.

Avoid Complex Gestures

One possible approach to creating touch-based interactions is to develop touch-traced patterns, which can then be recognized and used to trigger specific system responses. These touch-gestures can quickly become complex and can be hard to learn and remember. Instead, our goal is to keep the touch interactions simple, to when possible use established touch interactions (touch-and-drag, tap, double tap, long press, swipe, and pinch).

Consider the Viability of the Interaction Set

Creating a set of touch interactions that work together is harder than creating individual interactions. Performing the same touch interactions on the same on-screen element should lead to the same result. As a touch usually produces a reaction of the visuals even before the gesture itself is finished, interactions have to be able to handle ambiguity (touching an object can lead to touch-and-drag, swipe, or two-finger pinch gesture, etc.).

Interacting with TouchWave

In this section, we present interaction techniques available in TouchWave. For each technique we discuss the extent to which it mitigates particular aspects of the stacked graph problems. We address the legibility issues by introducing vertical rulers, supporting comparisons through adjustable layouts and temporarily extracting layers from their original positions, and provide navigation techniques for overcoming the scalability problems.

Vertical Rulers On Demand

The legibility problems of stacked graphs come on the one hand from the visual, curve-based data mapping but also from the lack of suitable scales. This is mostly due to the visual clutter that can be introduced by permanent scales. Offering scales on-demand for horizontal points allows reading explicit values while not overburdening the visualization. Our *vertical rulers* (Figure 4a) are an aid for these legibility issues. Touching the background behind a stacked graph in TouchWave creates a ruler that is bound to the position of the finger. Each ruler shows the size of all layers that it crosses at its current position by providing the numerical values. Moving the finger horizontally updates the labels correspondingly. Vertical rulers can be generated for more than one finger supporting both reading and comparing. This detail-on-demand technique can also be used to show additional information for each layer (e.g., movie names, see the box office case study).

Automatic Sorting and Adjustable Layouts

Vertical rulers allow reading absolute values, but sometimes only the relative order of layers is important (e.g., Which layer has the highest or lowest value for a given horizontal position? At which point is the layer no longer the largest?, see the box office case study). With *automatic sorting* (Figure 4b) TouchWave solves this comparison problem interactively. Press-and-hold a stacked graph in TouchWave sorts its layers based on their values at that horizontal position. This quickly shows the most influential layer for the position. Once the sorting is invoked, the finger can also be moved along the horizontal axis to continue sorting the layers based on the values at other horizontal positions.

In their discussion on stacked graphs [1], Byron and Wattenberg present several approaches that rely on changing a stacked graph's baseline to create a new layout.

Different layout approaches enable different comparisons; ThemeRivers help in comparing layers above and below the central timeline, while a basic stacked graph avoids the problems of curvature at least for the lowest layer. In order to quickly change between different layouts for a stacked graph, TouchWave enables *adjustable layouts* (Figure 4c) triggered by double tapping (the default layout is a streamgraph). As a transition between different layouts only changes the baseline, TouchWave uses an animated transition to help people keep track of points and layers.

Extracting Layers

Comparing two layers for different horizontal positions (e.g., if layer A's value at a certain point in time is larger/smaller than layer B's at a different point) can still be difficult even after adjusting the layout and using vertical rulers. TouchWave therefore makes it possible to *extract layers* from a stacked graph by dragging (Figure 4d). Touching a single layer and moving it far enough removes it from its parent. The parent stacked graph in TouchWave then adjusts the layout so that the remaining layers fill the gap. Since this works for multiple layers simultaneously and both fingers automatically create vertical rulers, rearranging and holding the layers against each other supports comparison of different sections. This interaction would be cumbersome to perform using a mouse in conjunction with keyboard modifiers which might be the reason why such deconstructions have not been done before. When a layer is dropped on the background canvas by lifting the finger, it creates a new single-layer. If a layer is dropped on an existing stacked graph it lands right above the layer it is currently over. This allows manual sorting and grouping of stacked graph, if automatic approaches such as sorting do not bring the expected results.

Navigation through Focus+Context

The main interaction technique that current stacked graphs provide to overcome the scalability problem is zooming & panning. Using this navigation for something as uniform as a stacked graph can, however, become disorienting. Also, losing the context makes it hard to perform comparisons. The concept of focus+context displays [9] addresses some of the problems of zooming & panning and works well in TouchWave through the availability of multiple input points. TouchWave bases its navigation on two-finger pinching gestures that are by now commonly used for scaling objects on multi-touch devices.

Horizontally pinching a stacked graph in TouchWave activates *horizontal scaling* (Figure 4e). The two vertical slices that the two fingers touch at the beginning of the gesture can be freely dragged towards the left or the right while the ends of the stacked graph stay fixed at the screen borders. The rest is distorted accordingly. So if, for example, the left vertical slice is dragged towards the left side of the screen, all horizontal values left of it are compressed, while all positions to its right are expanded. Similarly, moving the left vertical slice towards the right

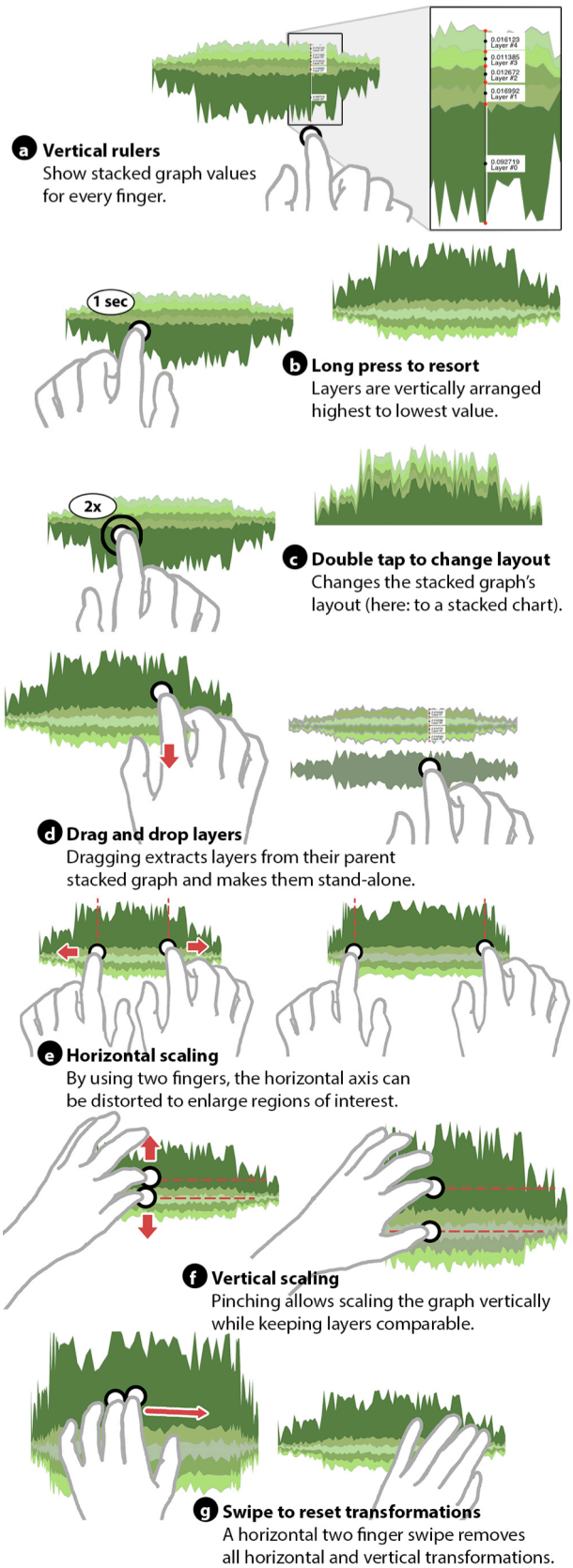

Figure 4: Interaction techniques for stacked graphs in TouchWave.

screen border compresses all values to the right of it, while expanding the ones to its left. The ends of a stacked graph in TouchWave are pinned to the left and right borders of the screen, preserving visibility of all parts at all times even though certain regions may be distorted. Thus two fingers can quickly adjust the size of a horizontal section while keeping the context intact. The view remains distorted until it is reset. All other interactions (e.g., changing the stacked graph's layout) preserve the distortion. The horizontal pinching gesture can be repeatedly applied to focus on even smaller horizontal regions.

Vertically pinching a stacked graph in TouchWave enables *vertical scaling* (Figure 4f). While using the same pinching gesture, there is one major difference between vertical and horizontal scaling. In horizontal scaling, the context is compressed but kept intact while the focus region is expanded. Using the same notion for vertical scaling would make the height of some layers bigger, while other layers would be shrunk. This would introduce a distortion of the layers' values and aggravate legibility by distorting the data representation. TouchWave therefore scales the vertical axis uniformly for vertical pinching to keep the relative sizes of the layers intact. As with horizontal scaling, other operations still work (changing the layout and reading values with the vertical rulers) and the vertical pinching gesture can be repeated multiple times to increase the scaling. This vertical scaling partially overcomes problems with too many layers and interacting with layers that would otherwise be too small because of the fat finger problem.

Both horizontal and vertical scaling introduce distortions to the graph that are non-trivial to revert using these techniques themselves. TouchWave therefore introduces a two-finger horizontal swiping gesture to *reset the scales* (Figure 4g). This removes all horizontal and vertical transformations and resets a stacked graph.

Introducing Hierarchy to TouchWave

Our set of interaction techniques addresses legibility and comparison issues of stacked graphs. However, our approach to scalability with focus+context only ameliorates some of the scalability issues. Once the number of layers crosses a certain threshold it is still difficult to see them all.

Extracting and re-organizing the layers in TouchWave helps with scalability. To take this further, introducing a hierarchy allows hiding unnecessary details; layers of interest can be expanded into sub-layers while other, less interesting layers can hide their sub-layers. Introducing a hierarchy also makes the underlying elements in TouchWave accessible, such as individual songs or tweets.

In this section, we present interaction techniques that enable conveniently working with hierarchical stacked graphs and reorganizing their layers. While BookVoyager [25] used an additional on-screen tree control for the hierarchical data structure, we build on integrated interactions with the stacked graph itself for our techniques.

Hierarchical and Item-based Stacked Graphs in TouchWave

Hierarchical stacked graphs contain multiple nested layers instead of a list of coequal layers. The underlying data structure is therefore a tree, with a single main stacked graph acting as root node and containing all other layers, each of which may in turn contain sub-layers and so on. At the lowest level, simple layers do not contain other layers but only values (leaf nodes). A layer's value at a given horizontal position is therefore either a specific numerical value (for a leaf node) or the sum of all its sub-layers' values at that position.

Previously, stacked graphs contained abstract numerical values, even if the underlying data is based on discrete items. For example, popular stacked graphs such as music listening histories or twitter data are based on atomic units (e.g., songs, tweets) but converted to numerical representations that allow displaying them in such a graph. Hierarchical stacked graphs in TouchWave, however, can display the numerical overview data and the atomic units: a leaf node can either contain discrete numerical values (e.g., stock prices) or atomic items (Figure 5). These items are bound to their horizontal position (e.g., their timestamp) but can move vertically and also have a range of influence depending on how much a single song is "worth." This influence is expressed through a radius for the item. Items have to be rearranged vertically to prevent overlap, which establishes a height value for the layout at that horizontal position. Listening to two songs in quick succession, for example, would lead to an overlap between the items. Moving one of them one item radius up prevents the overlap and creates a correct height reading (the layer is twice as high at that position as at positions with single songs, Figure 6). Displaying atomic units therefore does not introduce distortion to the graph and with the right range of influence, atomic unit and numerical value layers can be combined in the same stacked graph.

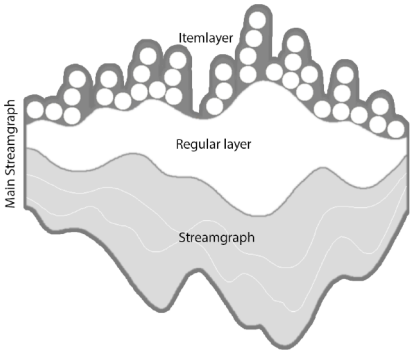

Figure 5: Hierarchical stacked graphs in TouchWave can contain regular layers, other stacked graphs or item layers consisting of atomic items (songs, tweets).

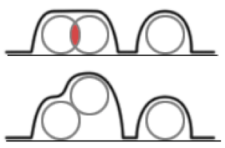

Figure 6: Resolving collisions between items in item layers.

Data Organization

Working with hierarchical stacked graphs in TouchWave requires adjustments to our existing techniques and the

introduction of new ones for managing the information. These new techniques allow showing/hiding sublayers, panning the background for more space and creating/removing copies of layers and sublayers.

Vertical scaling of a stacked graph now does not only adjust its size but also *shows its sublayers*, once a certain amount of scaling is reached (Figure 7a). Similarly, shrinking a layer's vertical size through pinching hides visible layers and reduces visual clutter. The lowest layers not only show single values but can also contain items which are represented as small, semi-transparent circles.

Panning the background (Figure 7b) by dragging it moves all stacked graphs in TouchWave up or down. Panning only works vertically, as all stacked graphs are fixed to the left and right borders of the screen. The background also starts to pan while dragging a layer if the dragged layer gets too close to the lower or upper screen borders. This allows finding an empty spot for dropping the layer without having to pan the background first. This gesture is identical to triggering a vertical ruler, which means that while panning a ruler is displayed as well. We decided to put up with the ambiguity instead of introducing a more complex gesture, as rulers are not too distracting.

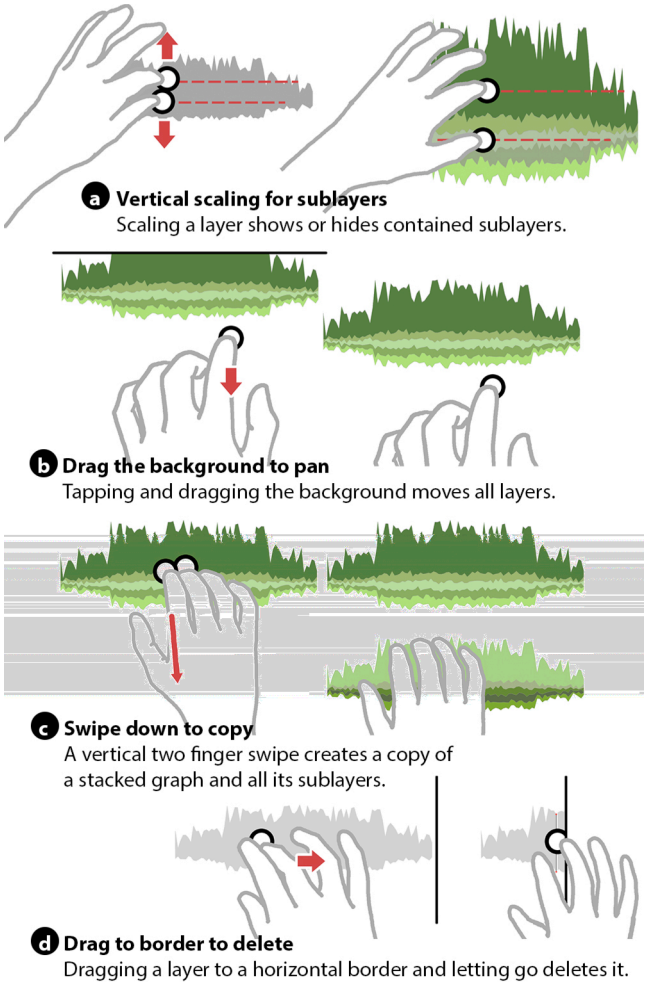

Figure 7: Extended interaction techniques for hierarchical stacked graphs in TouchWave.

Extracting a layer from a stacked graph in TouchWave by dragging and dropping is still possible and now also affects all contained sublayers; removing a layer from its parent and dropping it in another stacked graph moves it and all of its sublayers to that new position. This allows building new stacked graphs on-the-fly, simply by dropping and combining layers of interest on the background.

All layer re-organization changes the original order of a stacked graph. Therefore, a two-finger vertical swipe allows *copying a stacked graph* and all its sublayers (Figure 7c). In a fluid transition, the newly created copy is hurled in the direction of the swipe and stops at the lower or upper screen border. The way to *delete a copy* is to drag it to the vertical screen border and let it go (Figure 7d).

Implementation

TouchWave was written in Objective-C for iOS 5. We deployed TouchWave on a 3rd generation iPad with a resolution of 2048 x 1536 pixels on a 9.7 inch display. For this configuration, TouchWave supports up to 1,000 horizontal sampling points and 100 parallel layers at interactive frame rates. Reducing the number of sample points allows showing more layers and vice versa. The actual number of layers can be far greater, as non-visible layers do not have any impact on the performance.

The basic layout algorithms for our Objective-C code were derived from Byron and Wattenberg's 2008 paper [1] and their Processing drawing code (available as open-source on Github[1]). Fluid transitions between different states and layouts were enabled through the use of Apple's CoreAnimation framework, and graphical output was built using CoreGraphics. One TouchWave-specific implementation issue was creating a suitable layout for items within an item layer. A single item is defined by its horizontal position pos_x, its radius r, and a vertical position pos_y. Both pos_x and r are fixed and only pos_y can be adjusted to prevent overlaps (Figure 6). As we wanted to have this layout working on a portable device, we went for a non-optimal, but simple solution to solving collisions. Our algorithm works as follows. Each round the item with the largest radius is picked and dropped on the horizontal axis (y=0). Afterwards the item "bubbles up" (y is increased in steps whose length determines the accuracy of the layout) until it no longer collides with any other existing item. The algorithm finishes after placing all items.

CASE STUDIES

Stacked graphs are popular for various datasets, so we decided to demonstrate the possibilities of TouchWave with existing use cases from literature [1]. Two exemplary datasets we used are music listening histories and movie box office results.

[1] https://github.com/leebyron/streamgraph_generator

Music Listening Histories

One of the first widespread examples of a stacked graph is Byron's music listening history visualization. Discussions about layout approaches and implications for stacked graphs were published by Byron and Wattenberg [1]. The listening history stacked graph is based on creating a layer for each artist in the history. The values for each horizontal sampling point for one layer corresponds with the number of times a song by that specific artist has been listened to in one week. The result represents the changes in musical taste over time, but makes it difficult to read clear values and is completely static.

A music listening history in TouchWave provides multiple ways to interact with the data. Figure 8 shows a screenshot of two months of listening history data as interactive stacked graphs. Compared to the original, the data is hierarchical, with genres as the top level (visible at the top). Each genre layer consists of layers for artists that are in turn based on single songs. When enlarging an artist layer far enough, the single underlying songs become visible. In this example they appear as oval shapes, due to the non-uniform scaling (middle layer). The two most popular genres have been extracted from the overarching stacked graph and placed below it for further analysis (indie and electronic as blue and green layer, respectively in Figure 8). We work with 1,000 horizontal sample points in this example which means that each sample corresponds to roughly 80 minutes of the two months. The average song length of four minutes leads to much overlap between the song items and explains the stacked look of the middle layer. Pinching horizontally allows focusing on certain periods in time and pinching vertically controls the visible level of the hierarchy (genre, artists, songs). More detailed exploration is enabled through the possibility to deconstruct the stacked graph and drag interesting layers out. Additionally, the TouchWave version clearly shows the "burstiness" of music listening behavior: people listen to several songs before stopping again. This is apparent in the visualization as empty spots between stacked sets of songs. The free-flowing layout of the original Listening History visualization hinted at these gaps but brushed over them. The more realistic representation in TouchWave leads to worse results for the streamgraph layout however (there is no clear horizontal axis). Double tapping on the stacked graph allows switching to a different layout - ThemeRiver, for example, makes sure that all layers above and below the time axis form a symmetrical shape. Another aspect that is enabled through the hierarchical view is the comparison between different genres. It is interesting to see that the listening history is dominated by several main genres (indie, electronic, alternative (red color)) while other genres are more niche with only few songs.

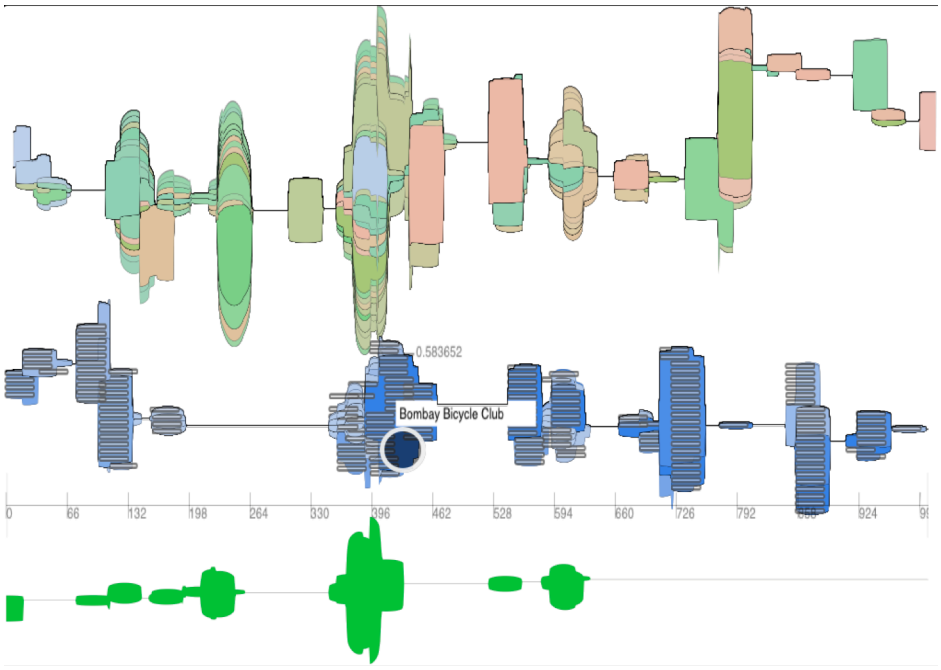

Figure 8: Two months of a listening history with genres and single artist layers.

Movie Box Office Revenues

Another popular example of a stacked graph is the New York Times' 'The Ebb & Flow of Movies' [3], a visualization of box office results from 1986 to 2008 (also in Byron and Wattenberg [1]). Each film is depicted as one layer and the stacked graph shows the development over time. Movies typically make the most money in the first few weeks and then start to slowly lose drive until they disappear from the theatres. The elongated shapes of single films fold into an overarching stream whose shape depicts the overall box office. The outer stacked graph shows seasonal shifts (summer blockbusters).

We used a similar dataset in TouchWave, showing the box office results for 52 films over 80 weeks (Figure 9). We applied a similar color coding to Byron and Wattenberg where more saturated red colors depict higher grossing films [1]. This allows a rough estimate of which films were the most successful. Interaction in the web version of the New York Times' graph[2] allows clicking on a layer to see a story synopsis and horizontal panning for navigation. While we did not integrate detail information about a movie into

[2] http://www.nytimes.com/interactive/2008/02/23/movies/20080223_REVENUE_GRAPHIC.html

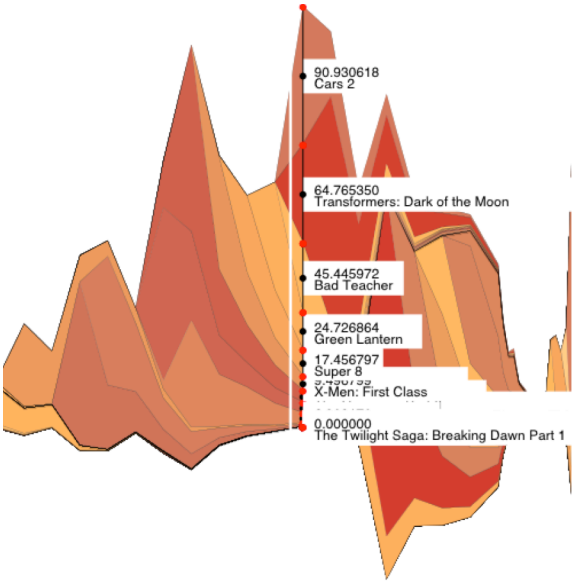

Figure 9: Section of box office results for movies. A vertical ruler allows reading the numerical values.

the TouchWave version, our two finger horizontal scaling enables focusing on specific regions of the graph while keeping the context. This allows comparing the current section to the top-grossing films of the whole dataset. Visually comparing a film's performance to others is difficult if they were not released at the same time. Relying on the color coding allows comparing the overall gross revenue, but even finding the number of weeks that both films were in the theaters is difficult. In TouchWave, two films can be compared by extracting their layers from the stacked graph with two fingers. Even if they were released at different times, both layers can be aligned to compare the films' shapes and their time at the box office. Focusing on a certain couple of weeks is possible through horizontal pinching. Additionally, while the color coding only depicts a rough estimate for the overall revenue, the vertical rulers show the specific box office results for a given week. Moving a ruler along the time axis or creating a second one with a second finger enables explicit comparisons. Finally, resorting the stacked graph at one specific horizontal position sorts all film layers by their revenue in that week.

DISCUSSION AND CONCLUSIONS

TouchWave is an example of rethinking an existing visualization technique to make a suitable, multi-touch based interaction. Our design goals can be valuable for extending other visualization techniques from the desktop:

- *Support kinetic manipulation*: endeavour to respond to a person's physical motions during interaction with harmonious interactive animation and responses.
- *Create integrated interactions*: look to spatially locate one's fingers and hands in contact with the visual representation whenever possible. The notion is to develop a hands-on data experience. Also, to leverage the full potential of touch interactions, avoid simply copying an interface with existing on-screen widgets. Instead, minimize widgets and integrate the widgets within the visual representation whenever possible.
- *Avoid complex gestures*: while the temptation might be to replace complex widget and menu systems with complex gestures, the challenge is to enable the same rich interaction with simple touch interaction.
- *Consider the viability of the interaction set*: when working towards simple, kinetic, integrated interactions it is important to pay attention to the creation of unambiguous interactions that where possible can be provided in a modeless interface.

TouchWave's kinetic interaction with the visualization uses fluid, animated transitions [7], to convey a natural, pseudo-physical feeling for the data. All interactions are integrated where touches are placed directly on the visualization. Our only additional widget, touch-based vertical rulers are also placed in situ. We adhere to simple touch actions, such as those that are becoming commonly accepted. Within the immediacy of integrated interaction TouchWave supports a mode-less approach to working with the data. Every type of manipulation and measurement can be triggered at any point without locking people into modes. This general approach has the advantage of maximizing the usage of available screen space for representing the data instead of filling some of it with buttons and other widgets. The downside is discoverability; figuring out what the system supports for the first time may not be easy as functions cannot be discovered by trying all the buttons. Suitable help mechanisms such as tutorials and in-place help would alleviate this problem and once learned, re-discovering interactions options might not be too difficult because everything is based only on simple touch gestures.

A main challenge while designing TouchWave was coming up with a consistent interaction set: a mapping between elementary touches, on-screen visuals and changes to the visualization that would lead to consistent and expected results. Ad-hoc mappings between touches and visuals usually lead to conflicts and ambiguities in the results. We found that creating a suitable combination of all these factors is a challenging problem in itself. Our approach was to first decide on a list of manipulations that we would make available. We then made two lists of all available visual categories (single layers, stacked graphs, background) and all available elementary interactions (tap, pinch, etc.). Finally, we determined what manipulation would happen for each combination of a visual category and an elementary interaction (e.g., background + touch ➔ vertical ruler and panning). This allowed us to pick suitable interactions for each manipulation without (unwanted) collisions. This process may prove useful to apply to other visualization techniques for multi-touch interactions.

We addressed the fat finger problem by initially scaling stacked graphs in TouchWave to the maximum available

screen size. Also, sublayers in each stacked graph in TouchWave are only shown for a certain size of stream. Vertical scaling allows increasing the size of a stacked graph and its layers to make reaching them easier. Still, for some cases, a single layer can just be too small and for these cases, having an on-screen widget such as a type of virtual magnifying glass [21,22] could help.

As multi-touch capable devices such as interactive tabletops, walls and tablet computers become more and more popular the demand for more complex applications such as visualization tools for those devices will also rise. With TouchWave we have demonstrated that multi-touch can provide a more fluid and modeless interaction with data that extends the existing technique. For example, TouchWave offers an integrated representation for hierarchical data in stacked graphs. Enabling interactive exploration through multi-touch gestures can ameliorate some of the issues arising from legibility, comparisons, and scalability. Even though TouchWave was developed for tablets, the proposed interaction techniques are applicable to all larger multi-touch devices (tabletops, wall displays).

With touch-capable devices available everywhere the formerly fringe case of touch-based visualization will continue to become more generally relevant. Other visualization techniques can equally benefit from the advantages of having a more immediate interaction with the data representations. Kinetic manipulation, integrated interaction and a consistent set of interactions based on simple gestures can improve the overall experience of a multi-touch visualization.

ACKNOWLEDGEMENTS

This research was supported in part by NSERC, AITF, CFI and SurfNet.

REFERENCES

1. Byron, L. and Wattenberg, M. Stacked Graphs - Geometry & Aesthetics, *IEEE TVCG (InfoVis 2008)*, vol. 14, no. 6, 2008, pp. 1245–1252.
2. Cleveland, W.S. and McGill, R. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods, *Journal of the American Statistical Association*, vol. 79, 1984, pp. 531–554.
3. Cox, A. and Byron, L. The Ebb and Flow of Box Office Sales, *The New York Times*, February 23, 2008.
4. Cui, W., Liu, S., Tan, L., Shi, C., Song, Y., Gao, Z., Tong, X., and Qu, H. TextFlow: Towards Better Understanding of Evolving Topics in Text, *IEEE TVCG*, vol. 17, no. 12, 2011, pp. 2412–2421.
5. Dörk, M., Gruen, D., Williamson, C., and Carpendale, S. A Visual Backchannel for Large-Scale Events. *IEEE TVCG (InfoVis 2010)*, vol. 16, no. 6, 2010, pp. 1129–1138.
6. Dwyer, T., Lee, B., Fisher, D., Inkpen, K., Isenberg, P., Robertson, G., and North, C. Understanding Multi-touch Manipulation for Surface Computing, *IEEE TVCG (InfoVis 2009)*, vol. 25, no. 19, 2009, pp. 961–968.
7. Elmqvist, N., Vande Moere, A., Jetter, H.-C., Cernea, D., Reiterer, H., and Jankun-Kelly, T.J. Fluid interaction for information visualization, *Information Visualization*, vol. 10, 2011, pp. 327–340.
8. Frisch, M., Heydekorn, J., and Dachselt, R. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces, *Proc. ITS 2009*, 2009, pp. 149–156.
9. Furnas, G.W. Generalized fisheye views, *Proc. CHI 1986*, 1986, pp. 16–23.
10. Hangal, S., Lam, H., and Heer, J. Muse: Reviving memories using email archives, *Proc. UIST 2011,* 2011, pp. 75–84.
11. Havre, S., Hetzler, B., Whitney, P., and Nowell, L. Themeriver: visualizing thematic changes in large document collections, *IEEE TVCG*, vol. 8, no. 1, 1999, pp. 9–20.
12. Heer, J., Viégas, F., and Wattenberg, M. Voyagers and voyeurs: supporting asynchronous collaborative information visualization, *Proc. CHI 2007*, 2007, pp. 1029–1038.
13. Isenberg, P., Hinrichs, U., Hancock, M., and Carpendale, S. Digital Tables for Collaborative Information Exploration, Tabletops - Horizontal Interactive Displays, C. Müller-Tomfelde, ed., Springer-Verlag, 2010, pp. 387–405.
14. Jetter, H.-C., Gerken, J., Zöllner, M., Reiterer, H., and Milic-Frayling, N. Materializing the Query with Facet-Streams – A Hybrid Surface for Collaborative Search on Tabletops, *Proc. CHI 2011*, 2011, pp. 3013–3022.
15. Lynch, S., Haber, J., Carpendale, S. ColourVis: Exploring Colour in Digital Images. *Computers & Graphics*, vol 36, no. 6, 2012, pp. 696-707.
16. Pentland, A. Maximum likelihood estimation: The best PEST, *Attention, Perception & Psychophysics,* vol. 28, no. 4, 1980, pp. 377–379.
17. Perlin, K. and Fox, D. Pad: An Alternative Approach to the Computer Interface, *Proc. SIGGRAPH 1993*, 1993, pp. 57–64.
18. Schmidt, S., Nacenta, M., Dachselt, R., and Carpendale, S. A Set of Multitouch Graph Interaction Techniques, *Proc. ITS 2011*, 2011, pp. 113–116.
19. Spindler, M., Tominski, C., Schumann, H., and Dachselt, R. Tangible Views for Information Visualization, *Proc. ITS 2010*, 2010, pp. 157–166.
20. Tufte, E. The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT, USA 1983.
21. Viégas, F., Wattenberg, M., van Ham, F., Kriss, J., and McKeon, M. ManyEyes: a Site for Visualization at Internet Scale, *Proc. InfoVis 2007*, 2007, pp. 1121–1128.
22. Vlaming, L., Collins, C., Hancock, M., Nacenta, M., Isenberg, T., Carpendale, S. Integrating 2D mouse emulation with 3D manipulation for visualizations on a multi-touch table. *Proc. ITS 2010*. 2012 221-230.
23. Voida, S., Tobiasz, M., Stromer, J., Isenberg, P., and Carpendale, S. Getting Practical with Interactive Tabletop Displays: Designing for Dense Data, "Fat Fingers," Diverse Interactions, and Face-to-Face Collaboration, *Proc. ITS 2009*, 2009, pp. 109–116.
24. Wattenberg, M. Baby Names, Visualization, and Social Data Analysis, *Proc. InfoVis 2005*, 2005, pp. 1–7.
25. Wattenberg, M. and Kriss, J. Designing for Social Data Analysis, *IEEE TVCG*, vol. 12, no. 4, 2006, pp. 549–557.
26. Yi, J., Kang, Y.A., Stasko, J.T., Jacko, J.A. Toward a Deeper Understanding of the Role of Interaction in Information Visualization, *IEEE TVCG (InfoVis 2007)*, vol. 13, no. 6, 2007, pp. 1224–1231.