SUPPORTING CREATIVITY IN THE CREATION OF DATA VISUALIZATIONS WITHOUT PROGRAMMING USING AN OBJECT-ORIENTED APPROACH

by

Drew Skau

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2013

Approved by:

____

Dr. Robert Kosara

____

Dr. Zachary Wartell

____

Dr. Celine Latulipe

____

Christopher Beorkrem

____

Dr. Mary Lou Maher

ABSTRACT

DREW SKAU. Supporting Creativity in the Creation of Data Visualizations Without Programming Using An Object-Oriented Approach. (Under the direction of DR. ROBERT KOSARA)

Visualizations are typically created using a pipeline approach that relies on pre-defined algorithms to map data to visuals. This prevents creativity from entering the process of building data visualizations. My work explores a new model for creating visualizations that involves designing data visualizations with an object- oriented approach. This allows the creation of novel visualizations and visual structures. I plan to explore this space further using prototype tools developed using the model. Evaluation of the tools will focus around creativity support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Creating new types of visualization invariably requires programming. Tools like Tableau create visualizations based on data and user input, but are limited to a relatively narrow selection of visualization types. Many designers are trying their hands on visualization frameworks like D3.js [6], but are not familiar enough with the programming concepts involved to be very effective. At the same time, the theory of visual representation has not advanced much since the seminal work by Bertin [3] and Mackinlay [16]. I believe that both problems can be solved by a fresh look at the nature of visual representation in visualization.

I propose a new model for the representation of data in information visualization: visualization primitives. Like graphical primitives, visualization primitives are simple geometrical objects that can be combined into more complex ones. In addition to just the graphical component, however, visualization primitives also connect to data and, in turn, produce output data. By designing simple prototypes and applying data to them, users can quickly create many different visualizations in a very short time.

The goals of this work are as follows. On the theory side, I want to develop a new model of visual data representation. I believe that current models that are based on pipelines are too limited and do not adequately define the connection between visual appearance and data. On the practical side, I believe that this model will translate into tools that will make it easier for non-technical users – such as designers, illustrators, and journalists – to create new visualizations from scratch. Designers are already accustomed to working with graphical objects, and manually change them to represent data. Using visualization primitives, they are now able to design prototypes that automatically and immediately represent the data.

I believe this approach will support creativity in visualization creation. The immediacy of the feedback promotes exploration of the visualization design space.

CHAPTER 2: RELATED WORK

Visualization primitives build on existing work on the theory of visualization, in particular Bertin's ideas and glyphs. We also consider them a tool, in particular one that is closely built on theory, like Protovis or D3. Finally, our goal is to encourage creative work with visualization, which requires ways of measuring creativity.

2.0.1 Theory

Information visualization is mostly understood as the visualization of discrete data points [17]. Each data point is represented by a visual *mark*, according to Bertin's highly influential work from the 1960s [3]. Bertin defined marks as graphical objects with visual (or *retinal*) properties.

Mackinlay's APT system [16] was directly based on Bertin's work, and the first one to create tailored visualizations for a dataset. APT's goal was the automatic generation of the visualization, however, not supporting the user's creativity. SAGE [22] provided similar functionality, but also primarily oriented at data rather than visual design. Tableau [24], which is partly based on the ideas behind APT, goes in a similar direction, though the user can construct visualizations from scratch. Tableau's expressive power is limited to a relatively small number of plot types, however. We want the user to be able to create entirely new types of visualizations quickly and easily.

Card and Mackinlay developed a notation for analyzing the visual mappings used in visualizations [7]. Their goal was to connect point designs into a coherent design space, which then could be explored more effectively. However, their system is mostly analytical, not constructive: changes in the analysis do not translate back into novel visualization designs.

The Grammar of Graphics [29, 31] describes a way of defining almost any visualization imaginable, but from a mathematical and computational perspective that does not necessarily match up with practical visualization creation or best practice. For example, the idea that a pie chart is nothing but a stacked bar chart that has been transformed into polar coordinates may be a correct mathematical explanation, but we argue that few non-technical users would even understand this; much less construct a pie chart this way.

Visualization primitives are an extension of glyphs [1], which are graphical objects whose properties (lengths of parts, angles, colors) represent data. Glyphs are typically used to show small numbers of high-dimensional data points, but we argue that the idea can be applied in a much more general way. Tools for constructing glyphs [21] have been proposed, but tend to be driven by form-based interfaces and limited in the scope of available properties and ways of combining them into more complex (and "traditional") visualizations.

2.0.2 Tools

Visualization toolkits like prefuse [11], Protovis [5], and D3.js [6] put many of Bertin's and Wilkinson's ideas into practice, and make it possible to create a wide variety of visualizations, including entirely new ones. While they elegantly abstract away many common tasks, they do require considerable programming knowledge, however. This is a major hurdle for many potential users of such tools, like journalists and designers.

Other tools that do not require actual programing, like Quadrigram [4, 19] and DEFOG [15], are still strongly influenced by computing constructs. Building a visualization using Quadrigram's data flow paradigm requires many steps that are effectively function calls, and are based on typical operations available in programming languages. DEFOG is also driven by data rather than the visual representation; that is a useful property for a data analysis tool, but it does not help designers construct new and interesting visualizations.

Several systems infer visualizations from hand drawn sketches, or abstractly created objects. NapkinVis [9] infers a common visualization type and uses Protovis to draw the resulting charts. CavePainting Visualizations by Keefe et al. [13] shows the value of having non-technical users involved in the visualization creation process, and explores how creativity can improve resulting visualizations. Brett Victor has been developing a system for Drawing Dynamic Visualizations [27] that appears to infer data connections from the structure of what has been drawn. These systems rely on expertise of technical users to create the actual visualizations by embedding it into the system, and by relying on known techniques.

2.0.3 Creativity Support

There has not been a wide array of work exploring creativity in the field of data visualization, however some work does still involve creativity. Lee et al. [14] have done some work investigating interaction with visualizations beyond keyboard and mouse. Their design considerations very closely parallel design guidelines for supporting creativity. Other significant work by Walny et al. [28] explores visualization creation on whiteboards. The study never explicitly investigates creativity, but the whiteboard format is relatively open ended, free-form, and collaborative, and the visualizations generated during the study are clearly creative. CavePainting [13] intentionally incorporates creative users as a means to explore creative techniques applied to scientific visualization creation. None of this work directly evaluates or explores creativity.

As creativity is a difficult concept to define, quantitatively evaluating creativity is also a difficult task. Evaluating a tool's support of creativity is somewhat easier, as it does not depend on the abilities or mental state of the user, only on the capabilities of the tool. The Creativity Support Index (CSI) [8] provides a system for measuring a tool's ability to support creativity. It uses a set of six factors, ranked with Likert scales, followed by a set of fifteen ranked pairs to prioritize the factors. The results of the index are a value out of 100, with 100 supporting creativity perfectly.

2.0.4

CHAPTER 3: COGNITIVE MOTIVATION

Part of the motivation for this work, both for the theoretical side and the practical uses for visualization designers, is based on the idea of designing visual objects as actual objects (rather than a pipeline). We believe that this is approach more closely matches an effective mental model that can benefit both education in information visualization and creativity in visualizations.

3.0.1 The Information Visualization Model

The theory of information visualization is built on discrete data items. Bertin's marks are individual objects, just as the data points rendered by APT [16] and the data items fed to a piece of Protovis [5] code. One of the two dimensions Tory and Möller [17] use to distinguish between scientific and information visualization is also the discrete domain of the data (the other dimension being whether the spatial layout is given or chosen).

This distinction is fundamental and important, because it leads to entirely different types of solutions than work being done in scientific visualization. Ray casting, splatting, and other techniques that assume and require a continuous data domain (where values can be meaningfully interpolated between data points) are of no use in information visualization; though some of our techniques are applicable in scientific visualization, because any real dataset necessarily consists of a limited number of data values.

Distinct data values translate into distinct, well-defined visual objects. While a volume dataset can be rendered at many different resolutions, and can even be improved by better reconstruction and interpolation techniques, the number of items to be drawn for a given dataset in information visualization is not up for debate: one visual shape for each data point (except when there is filtering or aggregation).

Many programs in information visualization, are not only written in object-oriented languages, but actually embody some of the ideas presented in this model. In particular, prefuse [11] and Protovis [5] provide classes that closely resemble primitives, as well as the means to iterate over such definitions to render data onto the screen.

3.0.2 Cognitive Models

Our perception is based on the notion of objects. Very early on, infants develop the ability to differentiate and track objects [23]. We do not see our world as a collection of pixels or surfaces, but distinct, physical objects.

This also applies to our perception of two-dimensional shapes. Pinker [20] describes a model of the mental processes behind reading and understanding a chart as an exercise in separating it into objects and examining the relationships between them. The user translates the properties of all these objects on the chart into cognitive representations that answer conceptual questions. Pinker shows that the cognitive representations of these objects are symbols with visual descriptions.

Recent work on the perceived relationships between elements of visualizations [32] has reinforced this idea: items in a bubble chart seemed to attract each other when they were similar or close to one another. Such an effect is only possible if these elements are, in fact, seen as actual objects, with mass and other physical properties.

Grammel et al. [10] calls for tools supporting iterative refinement, as well as tools to help teach novices about visualization. The model we propose can support tools that have iterative refinement. We also believe the experimentation process that this model allows can help users to learn about information visualization.

We believe that having a model that closely aligns with our perceptual and cognitive processes can assist creativity by allowing users to think in more natural ways. As a result, it may help to support creativity when using tools based on the model.

3.0.3

CHAPTER 4: MODEL

We propose that a complete set of visualization primitives along with an appropriate set of visual properties will cover the possible visualization design space for tabular data. Visualization primitives allow for the creation of prototype objects that are instantiated for data points.

Visualization primitives uses a familiar concept from object-oriented programming; prototypes and instances [25]. Prototypes contain the essential information about themselves. They are the blueprint that allow copies to be created. Prototypes in the visualization primitives model know what data they get mapped to, as well as some internal geometric relationships. A designer creates prototype primitives, and assigns values to their properties, to generate a visualization. Instances are the objects in a visualization; the specifications of an instance come from the prototype and work together with the data to create the form of the visualization.

4.0.1 Prototypes

The concept of visualization primitives is built on a system of primitive shapes and their visual properties. The shapes correspond to commonly used components of a visualization, and their properties correspond to commonly used visual properties.

There is no optimal set of primitives because the goals of different implementations may be different. It is possible to have an optimal set of primitives for a given set of goals.

One danger in implementing the model is having too much or too little complexity in a primitive. The key is to balance the complexity of the primitives with the capabilities it provides.

As a first pass, we covered five different primitives to explore the model. It should be noted that our set is not necessarily complete. For example, the line primitive in our implementation does not include assignments for endpoints. This would greatly increase the complexity of the internal geometric representation for lines, but adding this property in the future would make parallel coordinates, and perhaps other visualizations, simple to create.

4.0.2 Visual Properties

Data is represented through visual (or Bertin's *retinal*) properties of the object. The visual properties of a primitive are what defines the capabilities it has. Providing properties that closely align with the affordances of the visual object creates the most useful set of primitives. For example, a circle's size could be defined by a width, however it would be more useful to provide properties for area, radius, and circumference. These properties can encourage best practice in producing visualizations, simply by exposing the appropriate visual variables. To continue with the circle example, assigning quantity to radius creates a quadratic increase in area (a poor representation for human perception). But assigning that same quantity to area creates a linear increase, so the resulting visualization is perceived correctly.

In order to provide appropriate properties, each primitive must have internal geometric relationships. To continue with the circle example, the area, radius, and circumference properties are all interconnected. From any one of these properties, a circle's size can be calculated for drawing to screen. In addition, assigning one of the properties enables the calculation of the others. This supplies the user with an output data source that describes the existing objects. That data source can be used as input data for other objects.

4.0.2.1 Internal Geometric Relationships

Many visualization primitives have these internal geometric relationships between their properties. Embedding relationships into the software's representation of them lets the user take advantage of them when building the visual structure of their visualization. For example, this system makes it trivial to connect the left side of one rectangle to the right side of another.

These geometric relationships also mean that every primitive has a different set of properties. The differing properties help to determine what each primitive will be good at in a visualization. This is why we recommend that other implementations of the system take care that the properties be closely aligned with the affordances of the visual objects. The designer decides what data is mapped to what property, but the creator of the system decides what properties are available.

4.0.3 Iteration

We want the user to design a prototype object, not an algorithm. The visualization designer defines the loop once, and this creates each primitive seen in the visualization. When a user designs a loop, they are thinking about a data path or algorithm that gets the data to the final destination, but we want the user to be able to design an object. In visualization primitives we use implicit loops to allow the user to design a prototype object instead of a pipeline.

This iteration is built into the primitives and repeats for every item of data. The scaling and internal geometric relationships are calculated during the iteration, and the resulting data is stored in the primitive. Implicit iteration along with mappings are the key to allowing an object-oriented prototype design strategy. Together they turn a prototype primitive into the repeated instances of primitives in the visualization.

4.0.4 Data

Since iteration handles the assignment of the individual data values to each instance, in the interface for the user, our data can be reduced to fields. But not all data used in a visualization comes from the original data source.

Algorithmic approaches to generating visualizations often have hardcoded values, or sequences of values generated during the drawing process. We call these values administrative data because they set a size for line weights, or provide position information for bars in a bar chart. Visualization primitives do not use an algorithmic approach, but this administrative data is still necessary to produce a visualization. Our implementation calculates administrative data during the time of data import and provides it as a data source just like the actual data dimensions.

4.0.4.1 Mappings

Since instances of the primitive in the visualization are not designed individually, it is necessary for there to be a way to abstractly assign data to a prototype primitive's properties. Mappings provide this abstract connection. Each property knows its data source, and a scaling factor. The primitive uses this information in the generation of its own data.

4.0.4.2 Connections

Since primitives store their own sets of data, they can also be used as sources of data. Having output data is important for being able to build relationships between multiple primitives. Creating any sort of glyph-based visualization requires that their be relationships between the multiple objects.

The data primitives store has often been processed, either through scaling, or through the primitive's internal geometric relationships. Using data from a primitive provides a simple way to make meaningful spatial relationships between two different primitives. An example use case of this would be VIE-VISU, a glyph visualization using small multiples to represent data over time [12]. VIE-VISU shows up to twelve dimensions simultaneously in one graphic, with a thirteenth dimension of time represented by each multiple of the graphic. The primitives within the multiples have the same spatial relationships in each multiple. An example of this technique implemented in our system can be seen in Figure 6. Output data from each primitive provides a way to create the spatial relationships. Connecting the data of one primitive to another sets up a parent-child relationship between primitives. This allows the child to refer to the parent's data in order to generate its own data.

In some interface implementations, it is conceivable that this connection could be accomplished based on the spatial relationships between prototypes. For example, placing the left side of one prototype next to the right side of another prototype could trigger a connection between the respective data and property of the primitives. This method of defining connections only works for spatial relationships. Relationships between color, or other more abstract spatial properties (area for example) would have to be defined some other way.

4.0.5 Axes and Layout

Spatial layout is an important aspect of visualization design. Administrative data are the most common sources of data that influences layout. For example, in a bar chart the data mapped to the vertical size of the bars can only be scaled. This only impacts the height of the bars, but their horizontal positions come from a sequence, and their vertical position comes from a constant. The majority of the layout control comes from the step size in the sequence, and the constant mapped to bar width.

Axes are not in our current implementation (axes weren't critical for testing creativity), but an axis primitive could provide the labeling necessary. Its properties would include the label data source, direction, size, etc. The type of axis could also be driven by a property, switching between a spatial axes or color scale. The actual labeling could be automatically handled using the extension of Wilkinson's Algorithm by Talbot et al. [26]

Axes are less of an issue for glyphs; the size and position of the objects is entirely relative, there is no axis to compare the spatial dimensions to. In the case of multiple views (small multiples, trellis displays) that have axes in common, scales exchange information so they can use the same transforms. This exchange is possible using the output data from each primitive to create connections between the axes present in a visualization.

CHAPTER 5: EXAMPLES

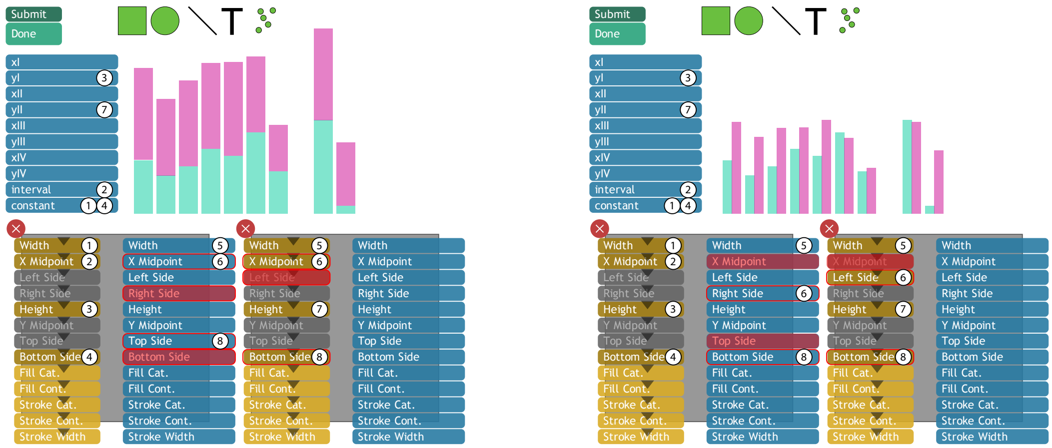
In order to help explain the property mapping process, we present a few examples. These examples show how to construct existing familiar visualization types. Some of them show interesting concepts that are highlighted when producing visualizations from a primitives based approach. Along with the description of how to construct them, we present versions of each visualization created with our current implementation of visualization primitives.

5.0.1 Bar Chart

Bar charts are created with a rectangle primitive. In a bar chart, the rectangle only has one data driven visual property, height. The rectangle's other properties are all derived from administrative data. Horizontal position is tied to a sequence, while other properties are all tied to a constant.

To create a stacked bar chart, a new rectangle primitive can be connected to the existing rectangle primitive (Figure 1(a)). To make the connection, the top side of the original rectangle is used as the input for the bottom side of the new rectangle. They share the same horizontal positions and widths, and the height comes from a separate data source. Users design not only the prototype primitive, but also the relationships between the prototype primitives.

To create a grouped bar chart, the data assignments are similar (Figure 1(b)). The two rectangles share the same bottom side position, and the same width, and the height of the



(a) Stacked Bar Chart (b) Grouped Bar Chart

Figure 1: A stacked bar chart and grouped bar chart in our implementation. The data in both is from Anscombe's Quartet [2]. Stacked bars (a) and grouped bars (b) differ only in the mapping of their sides. The numbers are added to indicate mappings, in the interface mouseover provides this information. The red highlights indicate the differences in the mappings between the two charts.

new rectangle still comes from external data. But now the left side of the new rectangle is connected to the right side of the original rectangle.

5.0.2 Scatterplot

A standard scatterplot showing only one category is easily created with a circle primitive. Horizontal and vertical position of the shape are the two data driven properties. The other properties are all administrative. For some datasets, a scatterplot can represent additional values with the color or size properties.

Scatterplots are very simple to create with our implementation (Figure 2). In this example a circle has been used, but any primitive in our implementation has the horizontal and vertical position properties that are necessary. Connecting a categorical value (in this example, number of cylinders) to the fill color property, maps the value to a member of a categorical color scale.

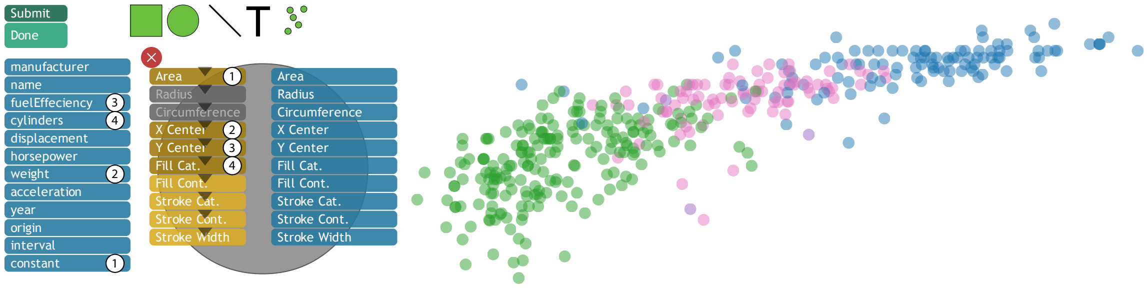


Figure 2: A scatterplot in our implementation using the cars dataset. Miles per gallon is attached to vertical position, weight is attached to horizontal position. Number of cylinders is connected to color. The numbers are added to indicate mappings, in the interface mouseover provides this information.

Other primitives designed for scatterplots could provide a categorical visual variable of shape. This variable would provide access to a set of shapes for representing categorical data.

5.0.3 Line Chart

Line charts are a unique case. They break the information visualization model slightly by indicating continuity within a data dimension. This special case requires special treatment in the visualization primitives model.

Line charts need to have the iteration of some of their data to be offset by one. This makes the two endpoints of the line instances refer to different items in the data, and creates the visual continuity between each data item. Without this connection, a line chart would simply be a series of points.

There are a number of ways this can be accomplished in the visualization primitives model. Since a line chart without connections decomposes into points, we opted to make the points primitive allow connections between each point instance. This allows the line primitive to keep an angular property for creating texture effects.

In a line chart, just like a bar chart, the primitive has one essential visual property. The

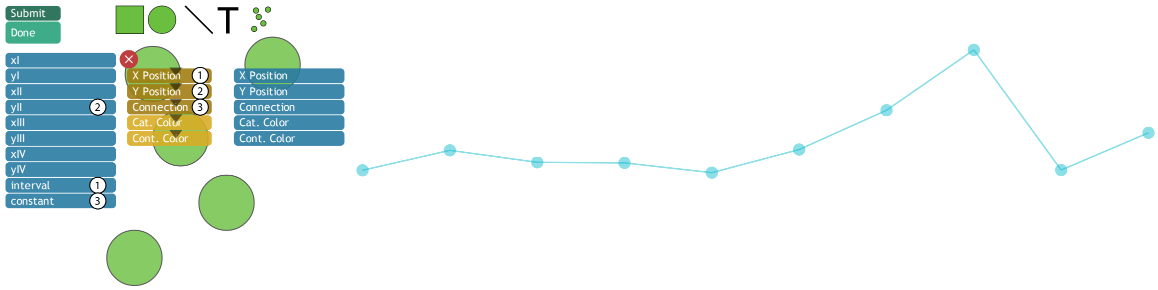


Figure 3: A line chart in our implementation is created by turning on a connection between points. The numbers are added to indicate mappings, in the interface mouseover provides this information.

remaining properties are all administrative, and only serve to create the layout.

For point primitives in our implementation, the connectedness of the points is also a property. This allows connections to be driven by data, although in a line chart, connections are mapped to a constant value to turn them on for all instances (Figure 3). When the point primitive draws its instances in the visualization view, it grabs the position for the connection's other end from its previous sibling in the data.

5.0.4 Heat Maps

Periodic data is often represented in heat map grids where color is tied to data, while time data build the grid (Figure 4). With the right breakdown of time in the data, this is also a simple visualization to create with visualization primitives. Our example uses Typical Meteorological Year 3 data for Charlotte, NC [30]. One data field indicating day of the year is mapped to the vertical position, while horizontal position comes from the hour of the day. The width and height of the primitive instances comes from a constant value. The color scale is tied to the dry bulb temperature.

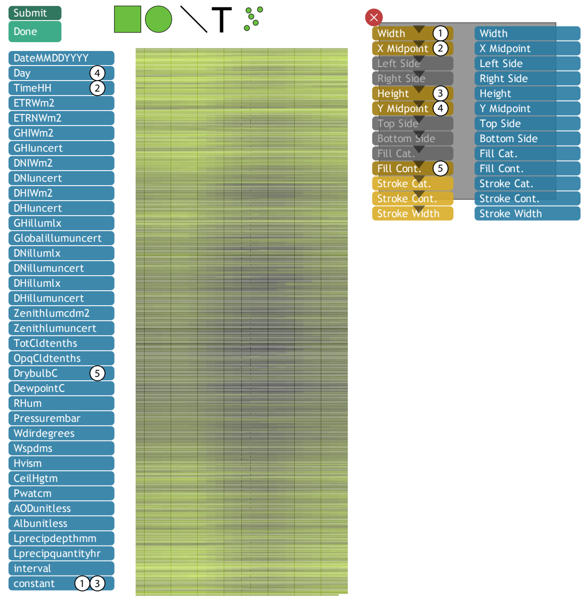


Figure 4: A heat map of temperature using TMY3 data [30]. The numbers are added to indicate mappings, in the interface mouseover provides this information.

5.0.5 Waterfall Chart

Related to the gantt chart, a less well known chart is the Waterfall Chart (Figure 5). This chart type is useful for seeing trends in timeline data. In our implementation, the chart is created using three primitives showing data on United States Presidents. One rectangle primitive shows the lifespan of the presidents. The left side is mapped to birthday, the right side is mapped to the day of their death (or the current date). The height is assigned a constant value, and the vertical position is a sequence. Another rectangle primitive displays the presidents' time in office. The left side is mapped to their inauguration, while the right is mapped to the end of their term. The height and vertical position come from the respective outputs on the other rectangle primitive. A text primitive labels the names of each president. The text string is mapped to the names field. The left side comes from the right side of the lifespan rectangle, while the vertical position and height are mapped to the

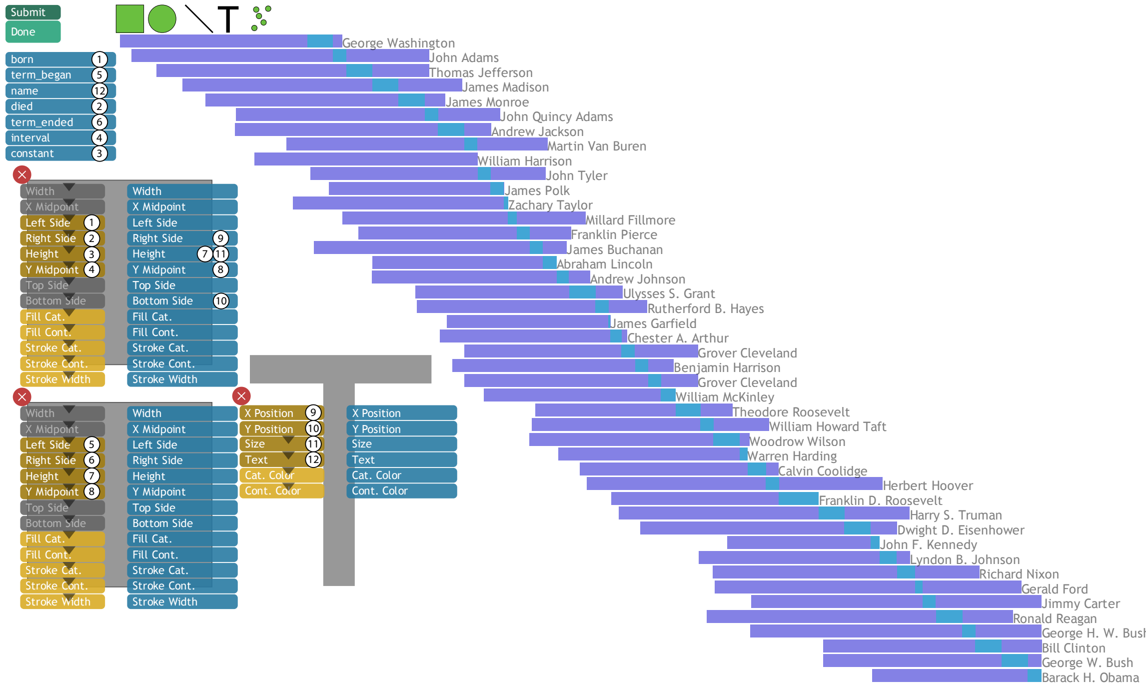


Figure 5: A waterfall chart showing the lifespans and terms of the presidents of the United States. The numbers are added to indicate mappings, in the interface mouseover provides this information.

respective properties of the lifespan rectangle.

5.0.6 Glyphs

Glyphs [12] are often used to create small multiples for comparison. An example of this technique implemented in our system can be seen in Figure 6. The visualization is built using the cars dataset. Four primitives make up each glyph, with each primitive representing a dimension of the data. Horsepower is the width of the blue rectangles, fuel efficiency is the width of the green rectangles. The yellow-green “wheels” show weight, while the blue “wheels” show acceleration.

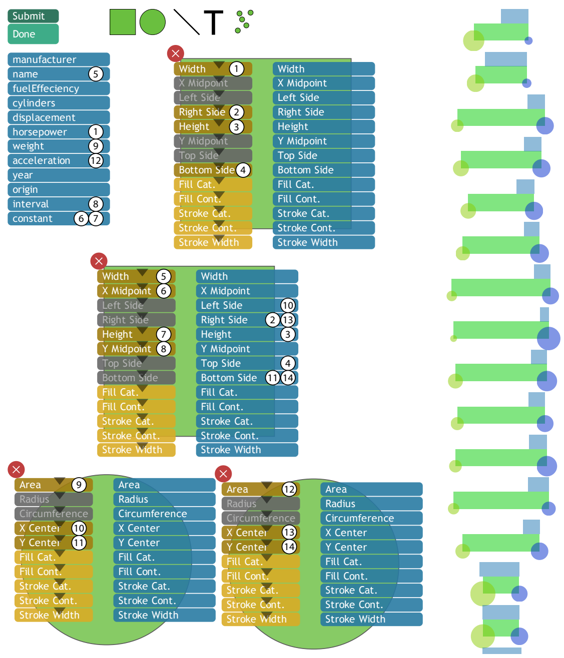


Figure 6: A glyph technique in our implementation using the cars dataset. The technique is similar to the one used in VIE-VISU [12]. The numbers are added to indicate mappings, in the interface mouseover provides this information.

CHAPTER 6: STUDY DESIGN

The aim of the visualization primitives model is to enable creativity during the creation of data visualizations. Given this goal, tests to measure the implementation need to evaluate the ability of the tool to support creativity. This means that a user study will not evaluate visualization primitives against other tools using tests of task efficiency, or other similar metrics. Instead, we employ the Creativity Support Index (CSI) [8] to evaluate the tool's ability to support creativity. The CSI is based closely on a tools ability to support a flow state for the user.

The study is intended to measure the tools ability to support creativity, not to quantify the quality of visualizations that are produced. These concepts are related, however they are not identical. Creative results are not guaranteed even with tools that support creativity perfectly. Creativity depends on several components, including the mind of the creator, and the capabilities of the tool at hand. We have included examples of "creative" visualizations generated with visualization primitives, however, these are anecdotal. Evaluating how creative they are is a secondary issue to how well visualization primitives support creativity.

The nature of creativity requires that the user study be flexible about regulations. Tasks that ask a user to find a certain thing in the data could provide inspiration, but they could also constrain thought processes. Time limits impose pressure that could help or hinder the creative thought process. We have opted to remove external pressures in the study, allowing participants to build visualizations at their own pace, setting their own tasks along the way.

CHAPTER 7: EXPERIMENT

To test the ability of Visualization Primitives to support creativity in visualization design, we conducted an online experiment. We chose an open-ended session design in which participants used Visualization Primitives freely, but with a dataset chosen by the experimenters and with questions preceding and following the session.

Specifically, we formed the following hypotheses regarding Visualization Primitives and creativity:

- More experience in visualization would lead to a higher score in the Creativity Support Index.
- More experience in visualization would lead to more visualizations generated using the tool.

7.0.1 Materials

The materials used in this experiment consist of three main elements: the dataset, visualization-specific demographics, and the Creativity Support Index.

We chose to use the Better Life Index as a dataset for several reasons. Primarily, the dataset has been used previously for a creative flower based visualization [18]. The dataset is also socially relevant since it contains information on several different countries. This means a wider group of people may find the data relevant to themselves, encouraging participation. Finally, the dataset size is large enough to provide interesting possibilities for visualization and analysis, yet small enough to run well in a browser.

Besides typical demographics such as age, education, and gender, we also included questions to measure participants' experience level with visualization tools. Specifically, participants were asked to rank their experience with other data visualization tools (using a 20-point scale), and to describe their experiences in a text area.

The Creativity Support Index (CSI) is an analogue to the NASA Task-load Index (TLX), but for measuring how well a tool supports creativity. In the CSI, participants are first given 6 20-point Likert-scale questions that measure the tool's ability to support exploration, collaboration, engagement, expressiveness, perceived effort, and the tool's ability to become transparent to the design process. Next, participants answer 15 ranked-pair questions, choosing which of the aforementioned aspects was more important to them during the activity (e.g. exploration versus collaboration).

7.0.2 Procedure

Participants were recruited via advertising on Twitter to the URL `http://visualizationprimitives.net`. No payments or incentives were given. It took approximately six days to gather all responses.

After following the link, participants were shown a briefing which described the experiment as an open-ended visualization creation tool. When participants indicated they were ready, they were asked for basic demographics (age, gender, education level).

After completing the demographics, a non-optional multiple step tutorial was started. In this tutorial, participants were required to perform the actions necessary to make a bubble chart (Figure 7). Participants were also instructed to submit "results they were happy with"

Figure 7: The bubble chart users were left with after completing the tutorial.

at any time via a submit button. That sent their current screen configuration to our server as a SVG (scalable vector graphics) file. SVGs were also sent automatically whenever a significant change to mappings was made.

Following the tutorial, participants were allowed to use Visualization Primitives with the OECD's Better Life Index [18] dataset for any amount of time, submitting notable visualizations as desired. In addition to participant-submitted visualizations, the system also auto-submitted when any new mapping was assigned.

When participants indicated they were finished with the open-ended session, they clicked a done button and were presented with the final questionnaire, which contained three parts: visualization-experience, the Creativity Support Index [8], and a space for any additional comments.

7.0.3 Results

As with many online studies, participation drastically fell off with each step. 875 participants made it past the introduction page. 674 answered the demographic information. 95 participants completed the tutorial. 44 completed the CSI, and only 31 of those built anything beyond the tutorial before answering the CSI. Participants who answered the CSI and completed the tutorial made an average of 14.31 (median of 8) mappings. Participants who did not answer the CSI, but completed the tutorial, made an average of 9.07 (median of 6) mappings. For the participants who answered the CSI, their average time with the primitives was 20.65 minutes, with a median of 13, and standard deviation of 22.68. Not

Factor	Mean	Median	Standard Deviation
exploration	41.62	39.5	24.46
collaboration	9	0	18.22
engagement	26.31	18.5	19.87
tradeoff	37.73	37.5	27.3
transparency	53	45	35.48
expressiveness	38.23	40	23.52

Table 1: Factor ranking results.

including submissions made during the tutorial, a total of 1060 SVG files were generated.

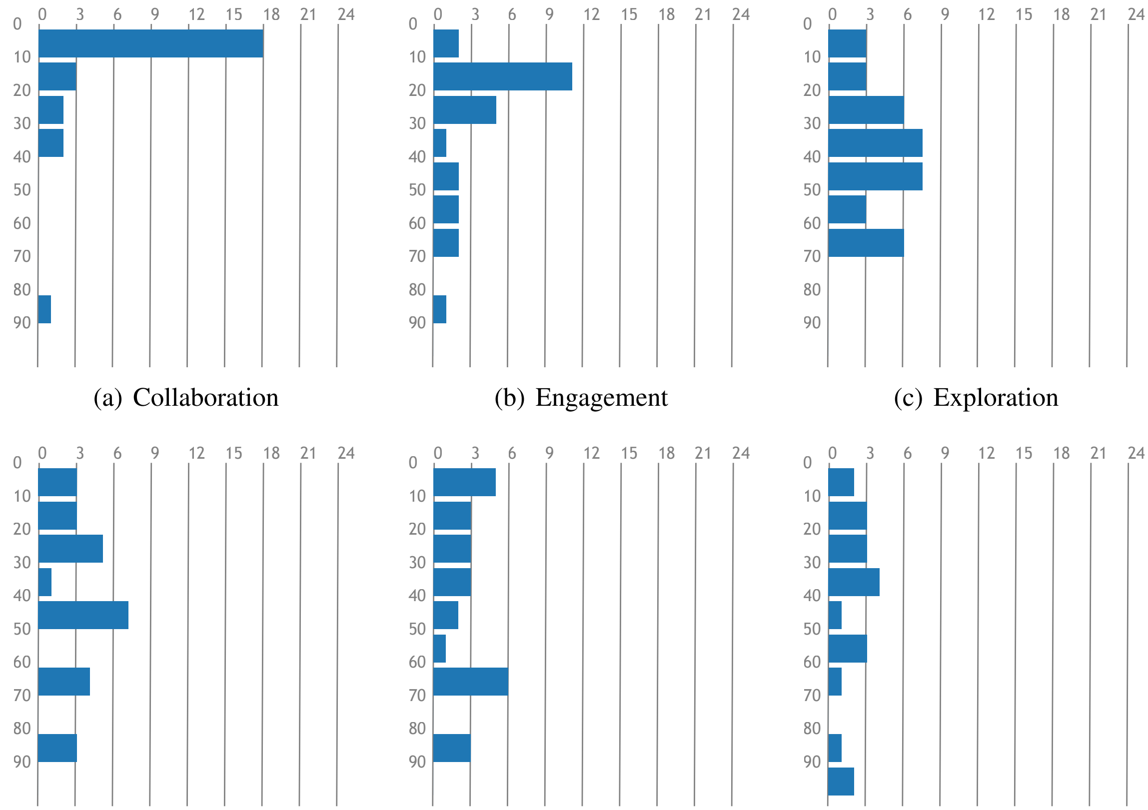
Since each participant worked with the same dataset and answered the same questions, our experiment adhered to a between-subjects design for the CSI evaluation. We divide our analysis into two cases to examine the relationship between experience, perceived creativity support, and productivity (i.e., the number of visualizations produced).

We excluded 4 participants who did not complete the CSI due to technical issues, and 3 participants who provided meaningless answers for the final questionnaire resulting in $n = 23$ for the final analysis.

Participants were then divided according to their reported experience with visualization tools. Since a 20-point scale was used for the visualization experience question, we divided participants evenly into a high ($experience >= 17, n = 12$) experience group and a low ($experience < 17, n = 11$) group.

7.0.3.1 Experience and Creativity Support

CSI scores are an index out of 100. They are calculated by combining ranked pairs with Likert scale results. For each factor, the Likert scale results are multiplied by the total factor count from the ranked pair portion. This value is then added to all the other factor results, and divided by 3 to arrive at an index out of 100. Table 1 and Figure 8 show the results.

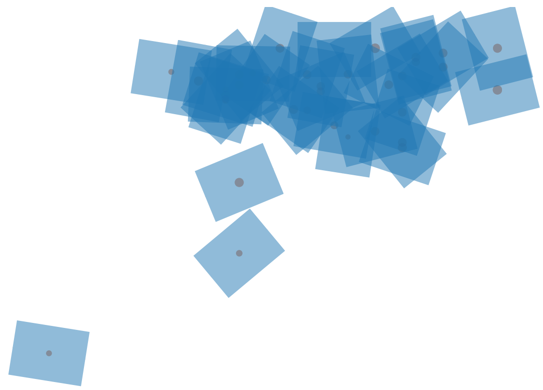


(a) Collaboration (b) Engagement (c) Exploration

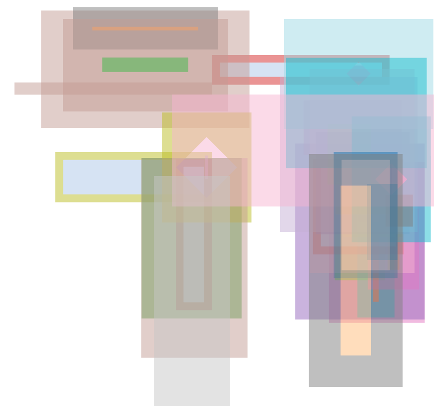
(d) Expressiveness (e) Effort/Reward (f) Tool Transparency

Figure 8: Histograms of the rankings for each factor (built using the visualization primitives prototype).

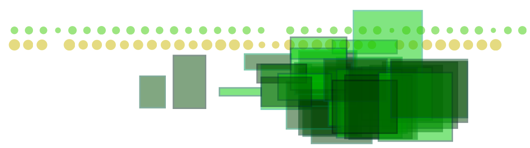
We compared CSI scores for the low-experience and high-experience groups using a t-test. This yields a significant effect for the CSI-score $t(21) = 2.0843; p = 0.0495$, with CSI-score in the high-experience group being higher than that of the low-experience group. The lowest mean CSI-score appeared in the low-experience group ($M = 54.27, SD = 22.50$), meaning participants in the high-experience group felt that Visualization Primitives supported their creativity in visualization design more ($M = 73.75, SD = 22.28$). (The paper presenting the CSI offers a sample ranking of 77.3 [8]) These results are consistent with our hypothesis that Visualization Primitives can significantly impact creativity in visualization design for participants who have sufficient experience with visualizations.



(a)

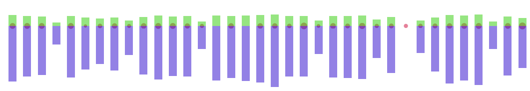


(b)

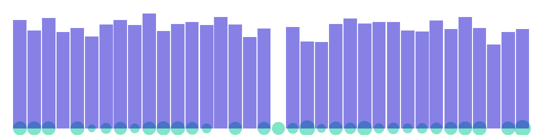


(c)

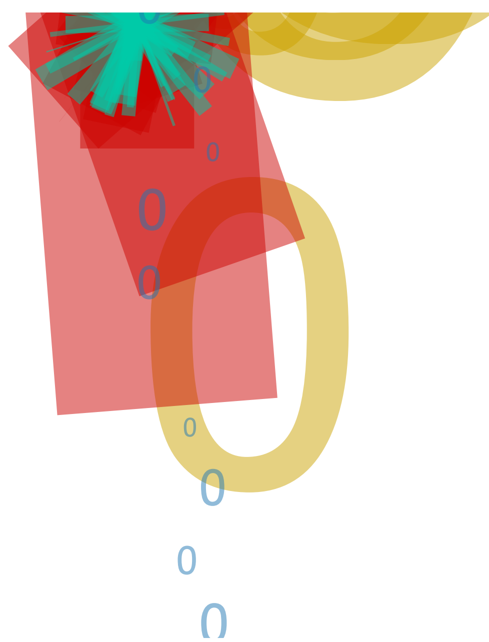
(d)



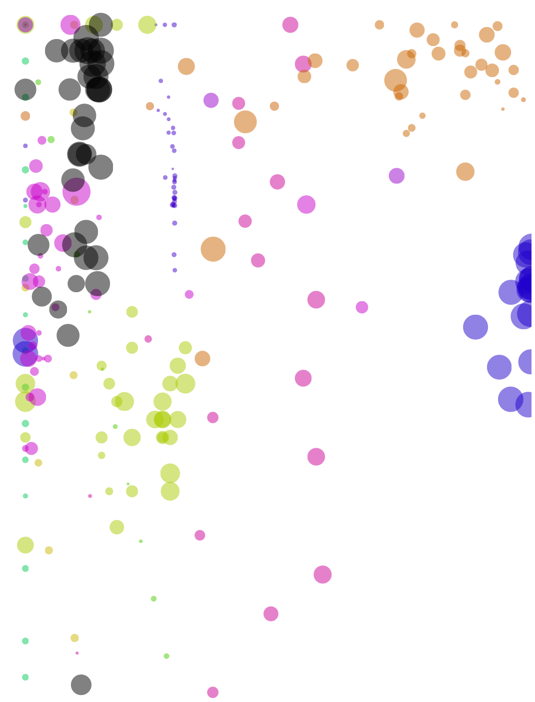
(e)



(f)



(g) By the same participant as 9(h).



(h) By the same participant as 9(g).

Figure 9: A selection of example visualizations submitted by study participants.

7.0.3.2 Experience and Productivity

We also compared experience with the total number of SVG graphics submitted. This did not yield a significant effect for the the total number of SVGs submitted, although participants in the high-experience group submitted more SVGs on average ($M = 21.42, SD = 19.38$) than those in the low-experience group ($M = 17.00, SD = 10.18$). These results do not support our hypothesis that users with more experience in visualization would submit a higher number of visualizations.

7.0.3.3 General CSI Results

In addition to discussing the total CSI scores, it is also useful to look at which categories the participants ranked most important to them while using Visualization Primitives. These results are shown in Figure 8. Tool transparency and exploration ranked most important, while collaboration and engagement ranked at the bottom of users' priorities.

7.0.3.4 Example Visualizations

The vast majority of users generated simple visualizations with a single primitive. Since the tutorial left users with a circle primitive, scatterplots became one of the most common visualizations that was created. Many users created visuals with only one or two primitives, and often there was no data driven spatial relationship between those primitives. Connections between primitives were rare, possibly because that ability was not explicitly shown to users during the tutorial, only described in text. The connections that were created were often built off of the bubble chart created during the tutorial (Figures 9(e), 9(f)). Often the creations fell somewhere between art and data visualization (Figures 9(a), 9(c), 9(d)), and some resulted in purely artistic work (Figures 9(b), 9(g), 9(h)).

CHAPTER 8: DISCUSSION

The results of the experiment indicate that visualization primitives support creativity, especially with users who have had adequate experience with visualization tools.

User interface issues were a limiting factor in users' exploration of the design space. Specifically, having to mouse over properties to see what they were bound to made it difficult to build and maintain a mental model of what was being shown. While our goal was to reduce clutter from connecting lines, the alternative may have created more issues than it solved.

The unfamiliarity of the interface definitely contributed to many users frustrations. Many comments expressed users' frustration with not knowing what the parts of the interface did. This could be addressed by further iterations of the tutorial.

The internal geometric representations of the primitives also may have contributed to confusion. Drawing elements to the screen is often not possible until multiple properties have been specified. This could be alleviated with default values, however the solution is far from elegant, and can produce unexpected behavior when assigning properties.

Not all the results that participants created and submitted were strictly visualizations in the sense of being useful. We see some playful examples that are more artistic than useful, which is in line with our goals. We did not ask people to create anything in particular so as not to constrain their exploration.

The considerable time spent on the site shows that they were engaged and interested in exploration, despite the user interface flaws This suggests that tapping into users' creativity is a promising way of getting them interested in visualization.

CHAPTER 9: FUTURE WORK

mappings connections between primitives children updates rethink creativity support

9.0.1 Set of Primitives

list of visualizations that cannot be created adjust set of primitives

CHAPTER 10: CONCLUSION

Our model presents a novel way to approach visualization design, and to support creativity. Visualization primitives in conjunction with visual properties create an environment optimized for rapidly prototyping new visualization techniques. The environment supports instant visual feedback and helps to develop an efficient flow for visualization design.

The theoretical contributions of this model are based on a cognitive argument that is closely aligned with visualization theory. Our study shows considerable interest in and potential for the creation of new visualizations using visualization primitives.

The most compelling argument for this approach is the control and flexibility given to non-technical users who want to create novel visualizations. We hope that some of the principles it presents can be integrated into existing and future visualization and visual analytics tools.

CHAPTER 11: APPENDIX

REFERENCES

[1] E. Anderson. A Semigraphical Method for the Analysis of Complex Problems. *Proceedings of the National Academy of Sciences of the United States of America*, 43(10):923–927, 1957.

[2] F. Anscombe. Graphs in Statistical Analysis. *The American Statistician*, 27(1):17–21, 1973.

[3] J. Bertin. *Semiology of graphics*. University of Wisconsin Press, 1983.

[4] Bestiario. Quadrigram. http://www.quadrigram.com/.

[5] M. Bostock and J. Heer. Protovis: A Graphical Toolkit for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–1128, 2009.

[6] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[7] S. Card and J. Mackinlay. The Structure of the Information Visualization Design Space. *Proceedings Information Visualization*, page 92, 1997.

[8] E. A. Carroll and C. Latulipe. The creativity support index. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 4009–4014. ACM, 2009.

[9] W. Chao and T. Munzner. Poster: Rapid Pen-Centric Authoring of Improvisational Visualizations with NapkinVis. *IEEE Transactions on Visualization and Computer Graphics*, 16(6), 2010.

[10] L. Grammel, M. Tory, and M. Storey. How information visualization novices construct visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):943–952, 2010.

[11] J. Heer, S. K. Card, and J. A. Landay. Prefuse: A Toolkit for Interactive Information Visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM.

[12] W. Horn, C. Popow, and L. Unterasinger. Metaphor graphics to visualize ICU data over time. *Intelligent Data Analysis in Medicine*, 1998.

[13] D. Keefe, D. Acevedo, J. Miles, F. Drury, S. Swartz, and D. H. Laidlaw. Scientific sketching for collaborative VR visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):835–847, July-August 2008.

[14] B. Lee, P. Isenberg, N. H. Riche, and S. Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2689–2698, 2012.

[15] L. Lins, D. Koop, J. Freire, and C. T. Silva. Defog: A system for data-backed visual composition. Technical Report UUSCI-2011-003, SCI Institute, University of Utah, 2011.

[16] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5(2):110–141, 1986.

[17] T. Möller. Rethinking Visualization: A High-Level Taxonomy. In *IEEE Symposium on Information Visualization*, pages 151–158, 2004.

[18] OECD. Better Life Index. http://www.oecdbetterlifeindex.org/.

[19] S. Ortiz. Use Cases of Impure, an Information Interface. In *VisWeek 2010 Discovery Exhibition*, pages 2–4, 2010.

[20] S. Pinker. A Theory of Graph Comprehension. In R. O. Freedle, editor, *Artificial Intelligence and the Future of Testing*, pages 73–126. Lawrence Erlbaum Associates, 1990.

[21] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: creating customized visualizations of complex data. *Computer*, 27(7):57–64, 1994.

[22] S. F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. Interactive Graphic Design Using Automatic Presentation Knowledge. In *Proceedings CHI*, pages 112–117. ACM Press, 1994.

[23] E. S. Spelke. Principles of Object Perception. *Cognitive Science*, 14(1):29–56, 1990.

[24] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[25] A. Taivalsaari. Classes vs. Prototypes: Some Philosophical and Historical Observations. *Journal of Object-Oriented Programming (JOOP)*, 10(7):44–50, 1997.

[26] J. Talbot, S. Lin, and P. Hanrahan. An Extension of Wilkinsons Algorithm for Positioning Tick Labels on Axes. *Computer*, 16(6):1036–1043, 2010.

[27] B. Victor. Drawing Dynamic Visualizations. http://worrydream.com/#!/Showreel2012.

[28] J. Walny, S. Carpendale, N. Henry Riche, G. Venolia, and P. Fawcett. Visual thinking in action: Visualizations as used on whiteboards. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2508–2517, 2011.

[29] H. Wickham. A Layered Grammar of Graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, Mar. 2010.

[30] S. Wilcox and W. Marion. *Users manual for TMY3 data sets*. National Renewable Energy Laboratory Golden, CO, 2008.

[31] L. Wilkinson and G. Wills. *The Grammar of Graphics*. Springer, Chicago, IL, 2nd edition, 2005.

[32] C. Ziemkiewicz and R. Kosara. Laws of Attraction: From Perceived Forces to Conceptual Similarity. *Transactions on Visualization and Computer Graphics (Proceedings InfoVis)*, 16(6):1009–1016, 2010.