

Scented Widgets: Improving Navigation Cues with Embedded Visualizations

Wesley Willett, Jeffrey Heer, and Maneesh Agrawala

Abstract—This paper presents *scented widgets*, graphical user interface controls enhanced with embedded visualizations that facilitate navigation in information spaces. We describe design guidelines for adding visual cues to common user interface widgets such as radio buttons, sliders, and combo boxes and contribute a general software framework for applying scented widgets within applications with minimal modifications to existing source code. We provide a number of example applications and describe a controlled experiment which finds that users exploring unfamiliar data make up to twice as many unique discoveries using widgets imbued with social navigation data. However, these differences equalize as familiarity with the data increases.

Index Terms—Information visualization, user interface toolkits, information foraging, social navigation, social data analysis.

1 INTRODUCTION

The success of an interactive visualization depends not only on the visual encodings, but also on the mechanisms for navigating the visualized information space. These navigational mechanisms can take many forms, including panning and zooming, text queries, and dynamic query widgets. However, effective navigation relies on more than input techniques alone; *appropriate visual navigation cues can aid users by guiding and refining their exploration.*

Both psychological and sociological considerations suggest approaches for improving navigation cues. Pirolli and Card’s information foraging theory [17] models the cost structure of human information gathering analogously to that of animals foraging for food. One result of this theory is the concept of *information scent*—a user’s “(imperfect) perception of the value, cost, or access path of information sources obtained from proximal cues” [17]. Improving information scent through better proximal cues lowers the cost structure of information foraging and improves information access.

While effective information scent cues may be based upon the underlying information content (e.g., when the text in a web hyperlink describes the content of the linked document, it serves as a scent), others may involve various forms of metadata, including usage patterns. In the physical world, we often navigate in response to the activity of others. When a crowd forms we may join in to see what the source of interest is. Alternatively, we may intentionally avoid crowds or well-worn thoroughfares, taking “the road less travelled” to uncover lesser-known places of interest. In the context of information spaces, such *social navigation* can direct our attention to hot spots of interest or to under-explored regions.

Our current interest in visual navigation cues is motivated by our experience building and deploying asynchronous collaborative visualization systems, in which groups of users perform visual data analysis by authoring comments and annotations within the visualizations [12, 20]. Usage studies of the sense.us collaborative visualization system [12] show that users fluidly switch between data-centric analysis and social navigation. After exhausting a line of inquiry, participants mine listings of comments left by other users to find new views of potential interest and to understand which areas have been explored. However, without explicit social navigation cues, users must continuously switch between the visualization and a separate list of comments.

In this paper we show that social activity cues can improve such

- Wesley Willett, Jeffrey Heer, and Maneesh Agrawala are with the Computer Science Division at the University of California at Berkeley, E-Mail: {willett, jheer, maneesh}@cs.berkeley.edu.

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.

For information on obtaining reprints of this article, please send e-mail to: twg@computer.org.

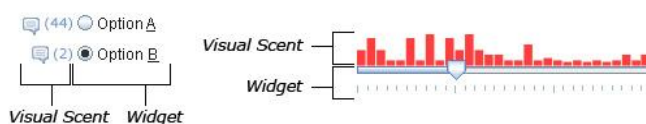


Figure 1. Widgets with visual information scent cues. Left: Radio buttons with comment counts. **Right:** Histogram slider with data totals.

social data analysis by enabling social navigation within the analytic environment of the visualization. We introduce *scented widgets*; enhanced user interface widgets with embedded visualizations that provide information scent cues for navigating information spaces (see Figure 1 for examples). We propose design guidelines for adding embedded visualizations to common user interface controls such as radio buttons, sliders, and combo boxes. We then present a Java-based toolkit-level software framework, developed according to these guidelines, that allows scented widgets to be added to user interfaces and bound to backing data sources. This framework allows visual navigation aids to be added to existing applications with minimal modifications to application source code. We also provide results from an initial evaluation of scented widgets in a social data analysis application. The results show that using scented widgets to provide social navigation cues help users make up to twice as many unique discoveries in unfamiliar datasets, but that these benefits equalize as users become more familiar with the data.

2 RELATED WORK

Numerous navigation mechanisms have been proposed to improve human-information interaction. In such interfaces, users may navigate along both *spatial* and *semantic* data dimensions. Examples of spatial navigation include maps and virtual worlds; examples of semantic navigation include web hyperlinks and dynamic query filters [1]. Navigation cues may be derived from the information content being explored (e.g., data distribution or landmarks) or from metadata, such as accumulated usage patterns. This last scenario is an example of *social navigation* [9], in which aggregated activity patterns are presented to promote awareness of other users’ actions within the information space. All such navigation cues provide proximal information that helps users stay oriented and gauge the relevance of distal information content.

One class of navigation aids seeks to facilitate browsing in geometric spaces, such as zoomable 2D canvases. Overview displays are one common approach, while other approaches embed navigation cues directly in focal display regions. For example, Halo [2] and City Lights [22] use marks near the periphery of a display to provide information about the relative position of off-screen elements.

Semantic navigation examples provide cues based on the information content itself. In visualization, histogram sliders [8] and

other data-driven variants [10] facilitate navigation to data regions of interest by summarizing the data distribution queried by the slider. On web pages, hyperlink text usually offers navigation cues about the content of the link target. This is the reason that human web surfers and modern web search indices rely on link text. Olston and Chi’s ScentTrails system [16] facilitates search and browsing of web sites by scoring documents in response to a text query and then enlarging hyperlink text to indicate paths to highly ranked documents. ScentTrails outperforms both searching and browsing alone in information-seeking tasks.

Another strategy is to provide information scent cues based on metadata. For example, social navigation is often based on displaying aggregated activity patterns. Blogs and discussion forums regularly include the number of posted comments in the link text of hyperlinks to discussions, while the del.icio.us social bookmarking service encodes the number of users who share a web bookmark in graduated red backgrounds for link text. Hill et al [14] explore the use of social navigation cues in a document editor, placing usage histograms within the scroll bar to indicate the prevalence of reading and editing activity throughout the document. Similarly, Björk and Redström [5] use color marks to indicate edits and search results along all edges of document frames. In the domain of collaborative visualization, Wattenberg and Kriss [20] gray-out visited regions of a visualization to provide “anti-social navigation” cues to promote analysis of unexplored regions.

Our work generalizes techniques such as histogram sliders and Hill’s read and edit wear, providing design considerations and a toolkit-level framework for embedding navigation cues in a variety of interface widgets. We contribute a general framework providing both data- and metadata-driven visual cues for navigating semantic dimensions in an information space.

Though not focused on navigation cues, a few additional projects share commonalities with scented widgets. Baudisch et al’s Phosphor [3] design provides real-time collaboration cues by using afterglow effects to highlight widget usage. Hill and Gutwin’s Multi-User Awareness UI [13] provides toolkit-level widget support for synchronous collaboration, such that users can see in real-time which interface widgets collaborators are using. Our scented widgets framework also provides a toolkit-level augmented widget suite, but one targeted at visual navigation cues rather than synchronous activity awareness

3 DESIGN CONSIDERATIONS

In designing a framework for encoding scent within widgets we consider; (1) the types of information metrics that can serve as navigation cues in scented widgets, (2) the matching of these encodings with the navigation models of the set of standard widgets, (3) the kinds of visual encodings used to convey this data, and (4) the modification of the standard widgets to accommodate scenting.

3.1 Information Scent Metrics

The first step in providing navigation cues is selecting the data source from which the cues will be derived. While the appropriate

data source usually depends on the specifics of the application, several kinds of data and metadata can be useful aids for navigation. One approach is to derive metrics directly from the information content. For example, a simple metric for interactive visualization is the number of visible data elements in each application state. This metric provides a sense of the density of data across the information space. More complicated metrics can be computed from the data itself, and may involve input from the user. Users might type in queries, as in ScentTrails [16], and be given scenting cues that indicate relevance scores. Alternatively, advanced users might use an expression language to enter in their own calculations over a visualized data set.

Social activity metrics are another potential data source, providing cues for social navigation. Interactive visualization applications such as sense.us [12] capture a number of social activity metrics that are typically invisible to users, but which could serve as valuable navigation cues. For example, displaying the number of visits to a view, comments on a view, or edits of a view, could guide users towards the relevant or most interesting views. Similarly, indicating the author of a comment or an edit could help users navigate to useful views. Temporal data regarding changes in any of these measures (e.g. recency or frequency information) are also candidates for display, as is location-based metadata. Our approach is premised on the notion that surfacing these sorts of activity metrics facilitates navigation.

3.2 Navigation and the Display of Visual Scent

Scent cues are specifically designed to aid navigation. Therefore scent cues should only be applied to interface elements that provide a way to navigate (i.e. change views) within the application. Moreover, widgets that represent a single navigation choice, such as buttons, should display only one scent value, while widgets such as combo boxes and sliders that offer multiple navigation choices should include scent cues corresponding to each potential choice.

3.3 Visual Encodings

Scented widgets embed a visualization of information scent metrics within a standard interface widget such as a slider, button, or combo box. Standard widgets are usually designed to fit within a small screen-space and a goal of our scented widgets designs is to add information to these widgets without adversely impacting user interface design.

We begin by considering a basic language of visual encodings for data. These include visual variables such as position, size, angle, color, and shape [4, 6, 15]. As noted by Cleveland [7] and Mackinlay [15], some encodings are more suitable than others for displaying different types of information. For example, position encodings are more accurate than length encodings for quantitative data, which in turn are more accurate than area encodings. For nominal data, color encodings are better than position.

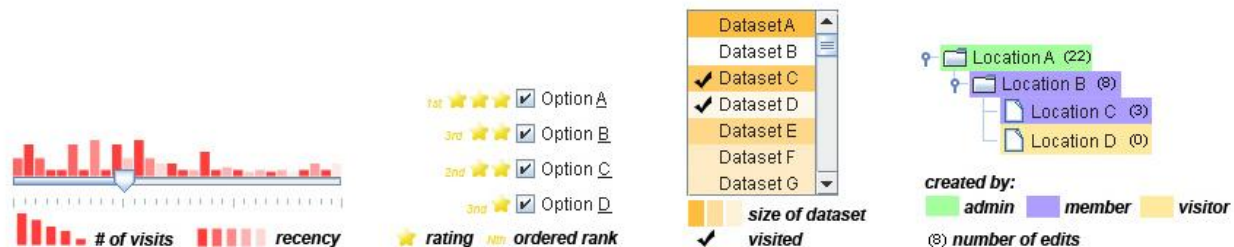


Figure 2. Examples of several scent encodings. From left to right: 1. A slider with visit totals encoded as a bar chart with recency encoded as opacity. 2. Checkboxes with star rankings encoded using icons and rank values displayed as text. 3. A list box with dataset sizes encoded using opacity and a visited/not visited value encoded using an icon. 4. A tree with author categories encoded using hue and edit totals encoded as text.

We can leverage these encodings in two distinct ways to convey information on or within a widget. One approach is to directly alter the attributes of the widgets that correspond to a given encoding. For example, a button's color could be based on the number of times the application state it leads to has been manipulated by users. Because widget sizes, shapes, and layouts are typically fixed, only a few of the visual variables (hue, saturation, lightness, and texture) can be applied directly to the widgets without disrupting the layout and impeding usability. However, visual variables such as position and length are typically more effective for displaying quantitative data. Therefore, as a second option, small visualizations that support these encodings can be embedded into the widgets. Examples include bar charts over a slider (e.g., Figure 1, [8]) and small, word-sized line charts (similar to Tufte's *sparklines* [19]) integrated with widget text.

3.4 Modifying Widgets

Based on these observations, we have selected seven different scent encodings to support within our framework. Direct encodings include the hue, saturation, and lightness properties of the widget. We also include four types of embedded visualizations: inset text, shape/icon, bar chart, and line chart. The examples in Figure 2 show several of these encodings applied to standard Swing widgets, while Table 1 describes each supported encoding type. We avoid encoding scent onto a widget's existing text labels, as label formatting is often modified by the application to convey highlighting, selection, keyboard shortcut combinations and other information.

3.5 Design Guidelines and Feature Requirements

Through inspection of the design space of widgets and study of related work [15, 18], we have developed a set of guidelines for the design of scented widgets.

Scent Encoding Guidelines

Modes of scenting should be chosen that maximize comparability and consistency across the interface. More specifically:

All widgets visualizing the same scent data should use matching visual encodings. Rationale: Encoding the same data differently across widgets complicates visual comparison.

Modes of encoding should reinforce semantic relationships between the widget scent and encodings in the application. Rationale: Conflict between the scent and the other parts of the application will lessen the effectiveness of both. For example, avoid encoding scent using color if the application already uses color to display unrelated information.

Visualizations showing the same scent data should be scaled identically (e.g. linearly, logarithmically, etc.) across all widgets. Rationale: Scaling the same type of data differently across widgets undermines accurate visual comparison.

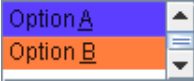
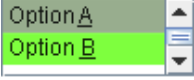

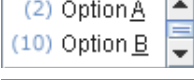
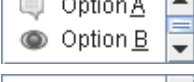
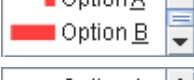
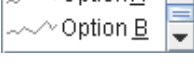
Modes of encoding should respect existing interface conventions. Rationale: User interface conventions tend to be well established and accepted by users, so scenting cues should not conflict with them. For example, a scent encoding should not repurpose text or icons commonly used elsewhere in the interface to encode unrelated data.

Encodings which make some elements markedly more salient than others, such as opacity, should be used with discretion. Rationale: If a widget is more salient than those around it, it is more likely to be used for navigation than its neighbors. Depending on the application, such enhancement may or may not be a desirable result.

Layout Guidelines

Interfaces should be laid out so that scented widgets are sufficiently proximal to allow comparisons between them. Rationale: Proximity aids judgments of position-based encodings and visual scent is most easily compared when graphic marks are adjacent.

Table 1. Scent encodings supported by scented widgets

Name	Description	Example
Hue	Varies the hue of the widget (or of a visualization embedded in it)	
Saturation	Varies the saturation of the widget (or of a visualization embedded in it)	
Opacity	Varies the opacity of the widget (or of a visualization embedded in it)	
Text	Inserts one or more small text figures into the widget	
Icon	Inserts one or more small icons into the widget.	
Bar Chart	Inserts one or more small bar chart visualizations into the widget	
Line Chart	Inserts one or more small line chart visualizations into the widget	

Scented widgets should be grouped, sized, aligned, and oriented similarly in order to provide common axes on which to compare scent. Rationale: Without common axes it is difficult to accurately compare marks across scented widgets, even if they show the same type of data.

Composition Guidelines

The overall number and type of scented widgets in a given interface should be small enough to allow easy comparison and visual tracking of changes. Rationale: The inclusion of too many scented widgets (and thus too many scent indicators) is likely to pollute the view, increasing cognitive load and making use more difficult.

Widgets should include identifiers (icons, tooltips, text, or a legend) that indicate what the scent cues correspond to. Rationale: It may be difficult for new users to discern what the cues indicate.

Many of these guidelines are addressed by our implementation. We deal with concerns about cross-widget consistency by grouping similarly-scented widgets and encoding them according to a shared configuration. While the distribution and layout of widgets in a user interface is clearly within the purview of developers, sizing, alignment and scaling can be fixed consistently across these groups.

4 IMPLEMENTATION

Using the preceding design analysis as a guide, our scented widgets framework provides toolkit-level support with which developers can quickly add visual scent cues to existing applications without writing a substantial amount of new code. The framework is implemented using Java Swing and takes advantage of the platform's Pluggable Look and Feel functionality, which allows the appearance of a wide range of standard interface widgets to be changed at runtime. In this section we discuss the design decisions made in our implementation, with the goal of providing guidance for developers building their own scented widget systems.

4.1 Rendering and Interaction

When implementing scented widgets, rendering and interacting with individual widgets is a primary concern. Ideally, the components for rendering visual scent cues should be implemented in a modular

fashion, such that application developers can reuse them across disparate widget types. In addition, the widgets should retain a familiar look and feel.

A number of implementation paths are possible. One might implement custom widgets from scratch, but this approach involves re-implementing basic rendering and interaction mechanisms and could result in an unfamiliar look and feel. Another strategy is to subclass existing widgets, overriding rendering and input handling techniques as needed. This approach is more efficient, requiring only targeted changes to widget behavior, but can still prove problematic. For example, restrictive access permission to members of the widget parent class may make it difficult to access parts of the widget state. Furthermore, both approaches require that developers explicitly use custom widget types in applications. Retro-fitting an existing application to use scented widgets then requires updating every widget definition in the application.

To avoid these limitations, we use Java's Pluggable Look and Feel layer to create a custom collection of scented widgets that can be installed without changing existing UI code. We extend Swing's default "Metal" Look and Feel and adjust the internal layouts of the Swing widgets to accommodate the embedded scent visualizations. Scented widgets also intercept user interface events as needed (e.g., allowing a mouse hover over an embedded visualization to trigger a custom tooltip for that graphic). Finally, we provide configurable renderers that are responsible for drawing the embedded visualizations. We use these *scent renderer* objects across the full widget set, promoting code reuse and ensuring consistent scent appearance in each widget type. Table 1 illustrates the encodings currently supported by our renderer.

4.2 Scent Configuration and Widget Groups

To map a backing data set onto visual scent cues, developers must provide a *visual specification*. For a group of related widgets, the visual specification indicates which data values to visualize and how to visualize them. Visual specifications define the names and data types of the variables to display in each scented widget and provide specific details about how the scent should be displayed. Specifications also maintain default values for encodings that are not determined by a variable. For example, a developer encoding a variable as a bar chart might specify default hue, saturation, or lightness values for the bars or add custom legend text or graphics.

In many cases, multiple widgets will show data from the same source, and the visualizations should be consistent across this group. Moreover, manipulation of a widget can alter the application state and require updates to the scenting of all related widgets. Our framework models these dependencies in a *widget group* abstraction

that monitors all widgets that should be updated in response to one another. Upon creation, developers associate a widget group with a visual specification and a backing data source. When a widget is added to the group, our framework automatically configures the widget to use the group's specification, ensuring consistent scent cues. The widget group then analyzes the widget to determine the set of potential values it can take. For example, a button can only be pressed, while a slider can take any number of values. The framework uses this set of potential values to determine the possible application states reachable at any given time. Next, the widget group adds listeners to the widget, allowing updates to both the widget's selection state and underlying data model to be processed by the framework.

4.3 Data Management

To track the current state of the application, every widget group models *state* as a set of name-value pairs for each widget in the group. When a widget value is changed (e.g. moving a slider, selecting a radio button, etc.), the widget updates its state pair. In some cases changing the value of a widget can affect the way other widgets in the application work. Thus, every time a widget changes state, the widget group requests new scent data for all the other widgets in the group to update their scent values.

To populate scented widgets with data, developers must implement the *data source* interface, which provides *scent data* in response to queries. Scent queries consist of the current state, the visual specification, and a reference to the widget being updated. Scent data, which may be numbers, strings, or arbitrary Java objects, are returned as sets of arrays for each variable defined in the visual specification. These arrays contain scent values for each state reachable using the widget under consideration. For quantitative and ordinal data, scent data objects can also provide a range over which the data will be scaled before rendering. Scaling may be linear or logarithmic, as configured in the visual specification.

Given the vast number of potential scent metrics, we expect that developers will build their own data source implementations that handle scent query requests in a domain-specific manner. However, our framework provides some tools that can help developers create custom data sources. For example, a caching layer caches query results and supports customizable replacement policies. Additionally, an SQL database helper aids developers in writing the code necessary to retrieve scent data from relational databases. The helper provides support for translating state objects and visual specification variables into SQL statements. A series of callbacks allow developers to customize the mapping between specified variable names and database column names and to generate custom database



Figure 3. Widgets from the usage example, before and after scenting.

```

01 //Create the VisualSpecification and define the scent encoding
02 VisualSpecification myVspec = new VisualSpecification();
03 myVspec.addVariable("numVisits", ScentConstants.QUANTITATIVE, ScentConstants.BARCHART, SwingConstants.VERTICAL);
04
05 //Get a ScentRegistry reference
06 ScentRegistry sr = ScentRegistry.getInstance();
07
08 //Create a WidgetGroup using the VisualSpecification and a data source
09 // defined by the developer which implements DataSource
10 sr.initWidgetGroup("myWidgetGroup", myVspec, new CachedDataSource(new VisitDataSource()));
11
12 //Create and register widgets, providing a name for the widget and
13 // the name of the WidgetGroup to which it should belong
14 JSlider myJSlider = new JSlider(1,20);
15 JList myJList = new JList(new Object[] {"Option A", "Option B", "Option C"});
16 sr.register("myWidgetGroup", "sliderValue", myJSlider);
17 sr.register("myWidgetGroup", "listValue", myJList);

```

Figure 4. Sample code for the usage example of the Scented Widgets framework.

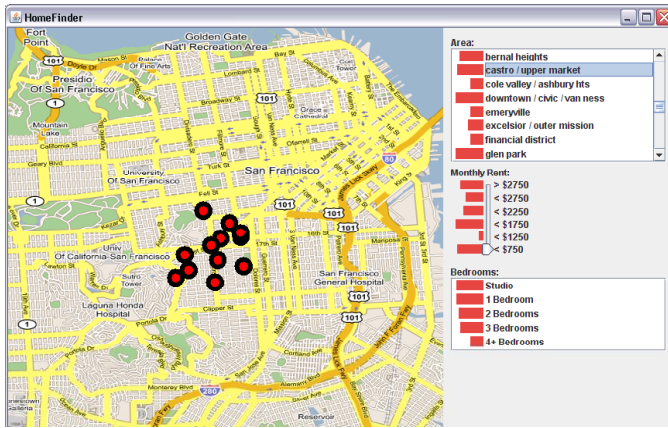


Figure 5. HomeFinder with histogram widgets. A scatter plot and scented query widgets show available apartments from craigslist.org.

keys from widget values. The helper then handles all data transfer, packaging the results of database queries into scent data instances.

4.4 Usage Example

The scented widgets API design is intended to allow developers to incorporate information scent cues into the widgets in their existing applications without substantial code revision. In the example given in Figures 3 and 4, we demonstrate how our framework can be used to provide scenting on a pair of interface widgets.

First we create a `VisualSpecification` and assign a scenting variable to it (lines 2-3). The system uses the assigned variable name to query the `DataSource`. The `QUANTITATIVE` and `BARCHART` arguments specify the data type of the variable and the visual encoding. Since we do not provide any other configuration details, the system relies on default settings for the other parameters. In this case, the system scales the quantitative scent values it receives from the `DataSource` and encodes them using a default color scheme.

Next we access the global `ScentRegistry` (line 6) to create a `WidgetGroup` (line 10). The widgets in this group will be scented using the encodings given in our `VisualSpecification`, with data values drawn from a `VisitDataSource` object. The `VisitDataSource` is a custom database wrapper that implements the `DataSource` interface to provide visit data about each of the widget states. Finally, we create a standard Java Swing slider and list box (lines 14-15) and, using a single line of code for each one, we register them with the `WidgetGroup` (lines 16-17). Thus, the system will query scent data from the `DataSource` and supply it to the widgets, which in turn will render themselves using the scent-enabled custom Look and Feel. The system refreshes the scent cues on each member of a widget group whenever a change is made to another member.

5 APPLICATIONS

As a preliminary evaluation of our framework, we have built three prototype applications that demonstrate diverse use cases for adding visual scent cues to traditional widgets.

5.1 HomeFinder with Histogram Sliders

The first application is a re-implementation of the HomeFinder [21], a geographic scatter plot visualization of available housing that uses dynamic query widgets to filter the view. Figure 5 shows our version of the application visualizing San Francisco apartment listings automatically harvested from craigslist.org RSS feeds. Scented widgets are used to show the number of available apartments across rental prices, neighborhoods, and number of bedrooms, providing an example of a data-driven scent metric. The `prefuse` toolkit [10] was used to provide the scatter plot and generate the query widgets, which were then registered as scented widgets. Scent data was

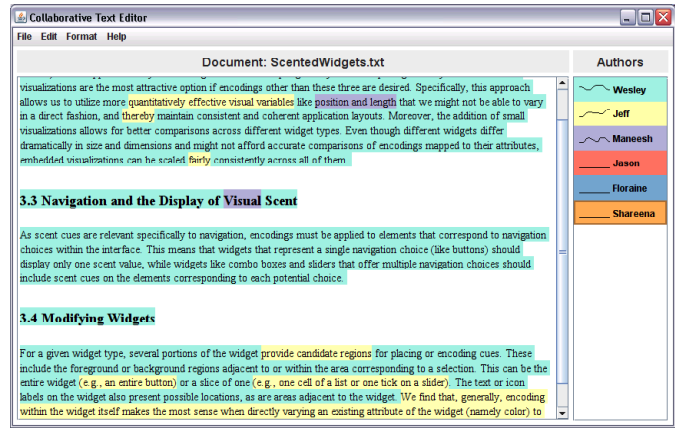


Figure 6. Collaborative text editor. A scented list widget identifies authors by color and displays a chart of editing activity over time.

provided by a custom data source that summarizes data in the underlying `prefuse` data table. The widget's visual specification was created with just one variable, the number of available houses, and was configured to use linearly-scaled bar charts.

5.2 Collaborative Authoring with Activity Indicators

The next application is a collaborative text editor, in which multiple authors access a document to simultaneously edit it. An example of our prototype is shown in Figure 6. Each author is assigned a unique color to identify the text segments they have edited. A scented list widget shows all authors who have viewed the document and a line chart of authors' daily edits. The combined interface allows authors to assess both textual editing patterns and the temporal activity of editors. To implement the prototype, we built a custom data source which models editing activity over time. A listener registered with the text editor aggregates editing events and posts them to a server. The visual specification includes two visual variables, one for hue and one for the line chart.

5.3 Social Data Analysis with Social Navigation

The third application uses scented widgets to add social navigation cues for collaborative data analysis. Figure 7 shows an interactive stacked area chart of the United States labor force from 1850-2000, broken down by occupation and gender. This is a reimplementation of a visualization used in the `sense.us` collaborative visualization environment [12]. Dynamic query widgets on the left allow users to navigate to specific occupations and toggle normalization of the data (i.e., view relative percentages or total worker count). As users explore the data, the system records their visitation patterns to an external database. Scented widgets then visualize these visitation patterns, indicating both highly visited and neglected views.

We implemented this application using the `prefuse` visualization toolkit, which provides the animated stacked area chart visualization. Our SQL data source helper was used to access a database of visitation patterns maintained by the application. The visual specification involves a single variable—the number of visits to each view—and specifies a bar chart encoding for the data. We used log scaling because the visitation data exhibited a power law distribution. We have also built a variant of this application that shows the number of comments made on each view.

6 EVALUATION

While prior work has explored various forms of data-driven scent cues [2, 5, 8, 10, 16, 22], less research attention has focused on visualizing social navigation cues [14, 20]. Therefore, we conducted a controlled experiment in which we asked subjects to perform information foraging tasks using the social data analysis application in Figure 7. We hypothesized that subjects would be more likely to revisit highly visited views using scented widgets, would make more

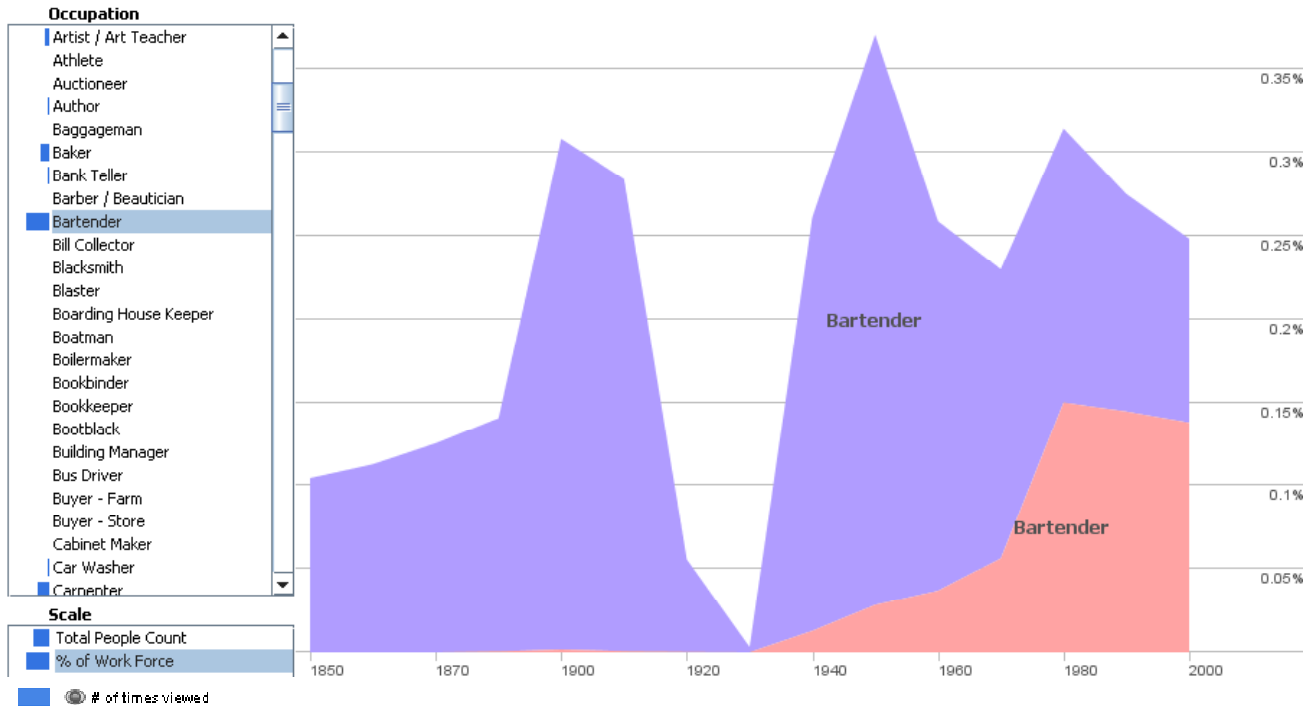


Figure 7. Social data analysis application with social navigation scent cues. A stacked time-series visualization shows the U.S. labor force, broken down by gender, from 1850-2000. The current view shows the percentage of the labor force that worked as Bartenders, with a sharp drop during Prohibition. Scented Widgets are used in the dynamic query widgets to show visitation rates in all views reachable from the current view.

unique discoveries using scented widgets, and would express a preference for scented widgets over traditional widgets. The study included twenty-eight participants (12 female, 16 male), all of whom were either graduate or undergraduate students, and were recruited through campus mailing lists. Participant ages ranged from 19 to 32 ($M = 25.3$, $SD = 3.8$).

6.1 Experiment Design

We asked subjects to find evidence either for or against specific hypotheses in a collaborative visualization of the United States labor force. We gave them an introductory tutorial to the system, and then asked them to complete three tasks. For each task, we presented subjects with one of the three following task hypotheses:

- T1:** Technology is costing jobs by making occupations obsolete.
- T2:** In the last half-century, women have joined the work force, but stereotypically male jobs remain almost entirely male.
- T3:** The number and variety of jobs directly related to the nation's food supply has diminished greatly since the 1800s.

For each task, we gave subjects 15 minutes to explore the data set and collect evidence relevant to the task hypothesis. The task hypotheses were intended to be of similar depth and diversity. We instructed subjects to make at least seven observations that provided evidence either for or against the current task hypothesis. At least two of the observations had to be unique findings on views not yet commented upon. Subjects were asked to note their observations by leaving new comments on the corresponding views.

For each task, we presented subjects with one of three scenting conditions. The conditions consisted of *no scent*, in which we used standard dynamic query widgets, *comment scent*, in which bar charts indicated the number of comments made on a view, and *visit scent*, in which bar charts indicated the number of prior visits to a view. To populate the interface with scent, we collected anonymized activity metrics from a study of the sense.us system [12] and supplemented them with a small amount of manual seeding to balance the metrics across conditions. Subjects in the previous sense.us study used a

similar visualization to freely explore the data. Our seed data consisted of a total of 1096 visits and 172 comments distributed across 154 views. Both visits ($R^2 = 0.96$) and comments ($R^2 = 0.90$) exhibited a power law distribution, and so we scaled them logarithmically for display in the scented widgets.

The study employed a 3 (Task) x 3 (Scent) between-subjects design. Task and scent pairings and presentation order were counter-balanced using a Latin Square. All tests were carried out in a laboratory environment using standard desktop PCs connected to a web server hosting the visualization and usage data. After completing the tasks, subjects filled out a survey that asked them to rate the scenting conditions on perceived utility and user experience.

6.2 Results: Revisitation

Our first hypothesis was that social navigation cues would increase the likelihood that users would visit views that others had visited previously. To test this hypothesis, we created three vectors, each representing the number of visits to each view in each scenting condition. We removed the starting overview from consideration, because users saw this view regardless of scenting condition. We then compared these visitation vectors to the visitation vector for the underlying activity measure used to seed the scented widgets. Using Pearson's product-moment statistic, we found correlations of $r(493) = 0.200$ for *visit scent*, $r(493) = 0.217$ for *comment scent*, and $r(493) = 0.181$ for *no scent* ($p < 0.01$ in all cases). These results suggest that users in the *visit* and *comment scent* conditions were more likely to visit the same views that were visited in the seed data than users in the *no scent* condition. However, we note that the correlations are not very strong. We believe that the semantics of the tasks also affect visitation patterns and likely had an effect on these correlations.

6.3 Results: Unique Discoveries

Next, we analyzed the data to check if scented widgets help users make unique discoveries. Our hypotheses were that scented conditions would have a higher occurrence of unique discoveries and that performance would improve over subsequent trials, regardless of

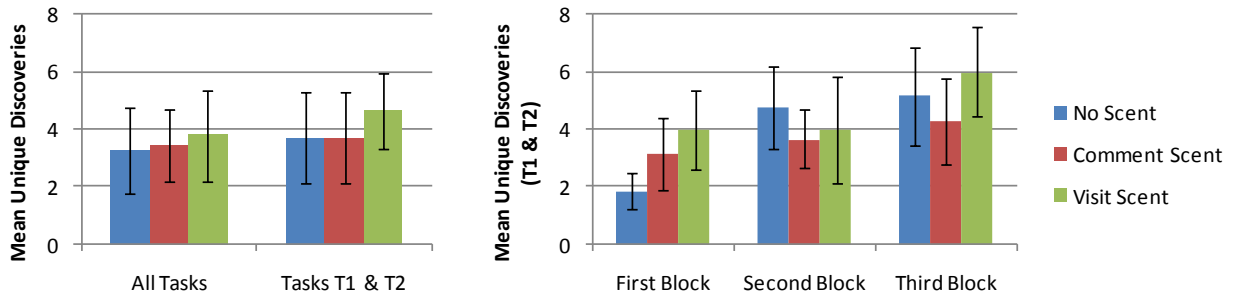


Figure 8. Experiment Results. Left: Mean unique discoveries for all tasks and just tasks T1 and T2. Right: Mean unique discoveries for tasks T1 and T2, divided into blocks by order of presentation. The differences in the first block are statistically significant.

the scented condition, due to learning effects. To compute a metric of unique findings we first collected all comments on visualization states that previously had no comments. We manually walked through these comments, decrementing the tally for comments that clearly had no bearing on the task hypothesis (e.g., jokes, unrelated questions, etc.). The result was a count of unique discoveries made in each task trial, across a total of 83 samples (due to a software glitch, one subject skipped a trial).

As shown in Figure 8, scented provided limited benefits over all tasks. The data are not normally distributed and so we used non-parametric tests (the Kruskal-Wallis H and Mann-Whitney U statistics) for statistical analysis. Based on these tests, the differences in unique discoveries between scented conditions did not reach significance ($H(2) = 1.245, p = 0.537$).

However, there was a significant main effect for task hypothesis ($H(2) = 11.154, p = 0.004$). Pairwise comparisons using Mann-Whitney tests found that unique discovery counts for task hypotheses T1 ($M = 4.2, SD = 2.4$) and T2 ($M = 4.3, SD = 2.4$) were not significantly different ($p = 0.456$), but that both were significantly different ($p = 0.008$ and $p = 0.002$, respectively) from T3 ($M = 2.6, SD = 1.3$). Examining the data, we found that a lower number of views were relevant to T3 and thus there was a limit on the number of possible unique findings. Subjects commented on only 25 unique views in T3, compared to 101 in T1 and 111 in T2.

We then analyzed the data according to the order in which the tasks were performed and found a significant main effect for task ordering ($H(2) = 6.341, p = 0.042$), indicating learning effects. The number of unique discoveries increases monotonically with practice, with significant differences between the first ($M = 3.0, SD = 1.7$) and subsequent blocks ($M = 3.6, SD = 2.1$ and $M = 4.4, SD = 2.6$). We then looked at the effects of scent within each block. Based on our earlier task analysis, we omitted the trials in T3. In the first block of trials, *visit scent* ($M = 4.1, SD = 1.6$) averaged 2.2 times more unique findings than *no scent* ($M = 1.9, SD = 0.4$) and *comment scent* ($M = 3.6, SD = 2.2$) averaged 1.7 times more. These differences were significant ($H(2) = 6.613, p = 0.037$). Pairwise comparisons found that *visit scent* resulted in significantly more unique findings than *no scent* ($p = 0.029$). The difference between *comment scent* and *no scent* failed to reach significance ($p = 0.053$), as did the difference between the two scented conditions ($p = 0.281$). Analyses for the second and third blocks of tasks found no significant effects for scent ($H(2) = 0.45, p = 0.799$ and $H(2) = 1.338, p = 0.512$).

6.4 Results: User Preferences

We analyzed survey responses and found that users significantly preferred both scented conditions to the non-scented condition across the board (Table 2): for finding undiscovered views, for finding discovered views, for finding interesting views more quickly, and in terms of enjoyment. We conducted a one-way ANOVA for each of these questions; each found a significant effect ($F(2,78) \geq 7.402, p < 0.002$ in all cases). In each case, we performed post-hoc comparisons using Fisher’s LSD test and found significant differences at the 0.05 level between the scented and non-scented conditions, but found no

Table 2: User Survey Results. All ratings are on a 5 point scale.

Survey Ratings	Visits		Comments		No Scent	
	M	SD	M	SD	M	SD
Finding undiscovered views	4.1	0.9	4.2	0.9	1.7	1.0
Finding discovered views	4.1	1.1	4.2	1.0	1.9	1.3
Finding interesting views	3.5	1.0	3.6	1.0	2.6	1.1
How enjoyable	4.1	0.7	4.1	0.7	3.3	1.2

significant difference between the two scented conditions. Furthermore, users did not find either scented condition to be cluttered or disruptive ($M = 1.6/5, SD = 1.0$ for both), and rated both about equally helpful overall ($M = 3.7/5, SD = 0.9$ for both). Users were evenly split between the scented conditions as to which condition was their favorite (14 comment, 12 visit, 1 no scent, 1 abstention), and the no scented condition was consistently ranked as the least favorite (24 no scent, 2 comment, 1 visit, 1 abstention).

The few complaints about scented widgets were largely related to users wanting the widgets to display different kinds of information. Three subjects expressed interest in toggling between multiple types of scented information, and several more made offhand remarks to this effect. One subject also voiced discomfort with the inability to turn off scent indicators, stating that she preferred to explore without being influenced by the browsing paths of previous users.

6.5 Discussion

The results suggest that subjects found scent useful for navigating the data when it was new to them, but as they learned the data, they relied on scent less. As their familiarity with the data increased, subjects may have transferred from social to semantic navigation of the data. Some caution is warranted in this claim, however, as we found advantages for scent after removing T3 from consideration. On the other hand, we only asked subjects to find a minimum of two unique discoveries, and so our results may be conservative. If users were asked to maximize unique discoveries, the differences between scented conditions might become stronger. As it stands, the results suggest that scented increases unique discoveries in unfamiliar data even when unique discoveries are not the primary concern.

The reduced impact of social navigation cues over time seems plausible given the limited complexity of the data set — it is not complicated, nor particularly large. The finding also has a nice intuitive analogue; in many tasks social navigation is unnecessary after one becomes familiar with one’s environment. A resulting hypothesis is that social navigation cues assist unfamiliar users in becoming oriented. Another hypothesis is that social navigation cues become increasingly useful for larger data sets as more time is needed to become familiar with the data. We leave further investigation of these hypotheses to future work.

It is also possible that earlier exposure to scent cues was partly responsible for the decreasing reliance on social cues we observed. All subjects that encountered the no scent condition in later blocks

had already been exposed to at least one of the sets of scenting data. More careful study is needed to assess if exposure to scent affects subsequent behaviour in other conditions.

At first glance, the results seem to suggest that visit scent may be preferable to comment scent. Though visit and comment scent fare equally well in user preference ratings, visit scent results in more unique discoveries than comment scent. However, the differences between the two are not statistically significant. Still, there are reasons to suspect benefits for visit scent. One hypothesis is that uninteresting views may be visited but are unlikely to accrue comments, so visitation metrics provide cues absent in comment scent. Another hypothesis is that, because commented views are visited more than uncommented ones, high visitation rates may be a good indicator of commentary. Indeed, analyzing the recorded activity metrics finds the expected correlation between visitation and commenting ($r(154) = .603, p < 0.01$). Further study is needed to determine which social navigation cues are to be preferred. In response to both this uncertainty and user requests, we recommend supporting user controls over the display of visual scent cues.

Finally, it is worth reiterating that the activity metrics used in the study were primarily drawn from general, unstructured exploration. We were interested in determining if making such activity traces visible impacts analysis, as one can collect this data easily and unobtrusively. However, one could also collect activity metrics in a more structured fashion. If visitation and commenting data can be associated with users' tasks or hypothesis, scented widgets could display scent data specific to the current task. The potential benefits for this form of scenting seemed clear enough not to require experimentation, but may be worth investigating to check if the same learning effects apply. In any case, task-specific scenting requires design mechanisms that allow task metadata to be associated with usage data in a lightweight fashion.

7 FUTURE WORK

Several limitations in the current system stand to be addressed in future work. One issue is widgets supporting multiple selections. In a multiple selection list box, a user can select one item from a list and then use a modifier key (typically *shift* or *ctrl*) to select additional items. As selecting a new item in the list *in addition* to the currently selected one leads to a different state than selecting *only* the new item, the number of potential states grows combinatorially. Modelling these states is straightforward, and we could use lazy querying of scent data to alleviate resource concerns, but there remain unresolved design issues. To handle multiple selections, scent can be updated not only when a widget value is changed, but also when a modifier key is depressed. Making scent displays modal in this fashion solves some design issues, but requires further study.

We have found that the most time-intensive part of applying scented widgets is implementing a data source. Further support for data management would reduce implementation time. Our SQL data source helper (Section 4.3) is one example, as it greatly speeds development when using a backing database. Further work is needed to better understand other data sources of interest and determine if we can provide toolkit support. For example, support for accessing data in visualization toolkits such as *prefuse* [11] could accelerate the creation of data-driven scented widgets.

8 CONCLUSION

In this paper we have introduced *scented widgets*, user interface components enhanced with embedded visualizations to aid information foraging. We proposed guidelines for incorporating small embedded visualizations and other visual cues into standard user interface designs. We then presented a toolkit-level framework for adding visual scenting cues to widgets in the Java Swing user interface toolkit. With a backing data source in place, our framework allows developers to quickly add visual navigation cues to existing applications with minimal changes to source code, typically with only a few additional lines of code.

We have evaluated our framework by building a set of demonstration applications and running a formal experiment in which scented widgets provided social navigation cues. Experimental results found that scenting led to more unique discoveries when users were unfamiliar with the data, but that these benefits equalized over time, suggesting a transfer from social to semantic navigation. Subjects significantly preferred scented widgets to traditional widgets for the analysis tasks and did not find visual scent cues to be cluttered or distracting.

ACKNOWLEDGEMENTS

We thank Martin Wattenberg for initial discussions that led to scented widgets and for sharing *sense.us* usage data. We also thank all the experimental subjects for their participation. This work was supported in part by fellowships from IBM and Microsoft as well as the Alfred P. Sloan Foundation and the Okawa Foundation.

REFERENCES

- [1] C. Ahlberg, C. Williamson, B. Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *Proc. ACM CHI 1992*: 619-626. Monterey, CA. May 1992.
- [2] P. Baudisch, R. Rosenholtz. Halo: a technique for visualizing off-screen objects. In *Proc. ACM CHI 2003*: 481-488. Fort Lauderdale, FL. Apr. 2003.
- [3] P. Baudisch, et al. Phosphor: explaining transitions in the user interface using afterglow effects. In *Proc. ACM UIST 2006*: 169-178. Oct. 2006.
- [4] J. Bertin, *Semiology of Graphics: Diagrams, Networks, Maps*. Madison, Wisconsin: The University of Wisconsin Press, 1967/1983.
- [5] S. Björk, J. Redström. Window Frames as Areas for Information Visualization. In *Proc. of the Second Nordic Conference on Human-Computer Interaction*: 247-250. Aarhus, Denmark. 2002.
- [6] S.K. Card, J.D. Mackinlay, B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufman. 1999.
- [7] W.S. Cleveland, R. McGill. Graphical Perception and Graphical Methods for Analyzing Scientific Data. *Science*, 229: 828-833. 1985.
- [8] M. Derthick, J. Harrison, A. Moore, S.F. Roth. Efficient multi-object dynamic query histograms. In *Proc. IEEE InfoVis 1999*: 84. Oct. 1999.
- [9] P. Dourish, Chalmers, M. Running Out of Space: Models of Information Navigation. In *Proc. Human Computer Interaction (HCI'94)*. 1994.
- [10] S. Eick. Data Visualization Sliders. In *Proc. ACM UIST 1994*: 119-120. Marina del Rey, California. Nov. 1994.
- [11] J. Heer, S.K. Card, J.A. Landay. *prefuse: A Toolkit for Interactive Information Visualization*. In *Proc. ACM CHI 2005*: 421-430. April 2005.
- [12] J. Heer, F. Viégas, M. Wattenberg. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. In *Proc. ACM CHI 2007*: 1029-1038. San Jose, CA. Apr. 2007.
- [13] J. Hill, C. Gutwin. Awareness support in a groupware widget toolkit. In *Proc. ACM SIGGROUP 2003*: 258-267. Sanibel Island, FL. Nov. 2003.
- [14] W. Hill, J.D. Hollan, D. Wroblewski, T. McCandless. Edit wear and read wear. In *Proc. ACM CHI 1992*: 3-9. Monterey, CA. May 1992.
- [15] J. Mackinlay. Automating the design of graphical presentation of relational information. *ACM Transactions on Graphics* 5(2): 110-141. 1986.
- [16] C. Olston, E.H. Chi. ScentTrails: Integrating browsing and searching on the Web. *ACM TOCHI*, 10(3): 177-197. Sep. 2003.
- [17] P. Pirolli, S.K. Card. Information Foraging. *Psychological Review*. 106(4): 643-675. 1999.
- [18] H. Senay, E. Ignatius. Rules and principles of scientific data visualization. Tech. Report GWU-IIST-90-13, The George Washington University. 1990.
- [19] E. Tufte. *Beautiful Evidence*. Graphics Press. 2006.
- [20] M. Wattenberg, J. Kriss. Designing for Social Data Analysis. *IEEE Trans. on Visualization and Computer Graphics*, 12(4): 549-557. 2006.
- [21] C. Williamson, B. Shneiderman. The dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system In *Proc. ACM SIGIR 1992*: 338-346. Copenhagen, Denmark. Jun. 1992.
- [22] P.T. Zellweger, J.D. Mackinlay, L. Good, M. Stefik, P. Baudisch. City lights: contextual views in minimal space. In *Extended Abstracts of ACM CHI 2003*: 838-839. Fort Lauderdale, FL. Apr. 2003.