Constant Information Density in Zoomable Interfaces

Allison Woodruff, James Landay, Michael Stonebraker

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720 USA
E-mail: {woodruff,landay,mike}@cs.berkeley.edu

ABSTRACT

We introduce a system that helps users construct interactive visualizations with constant information density. This work is an extension of the DataSplash database visualization environment. DataSplash is a direct manipulation system in which users can construct and navigate visualizations. Objects' appearances change as users zoom closer to or further away from the visualization. Users specify graphically the point at which these changes occur.

Our experience with DataSplash indicates that users find it difficult to construct visualizations that display an appropriate amount of detail. In this paper, we introduce an extension to DataSplash based on the Principle of Constant Information Density. This extension gives users feedback about the density of visualizations as they create them. We also introduce an extension that suggests improvements to existing visualizations.

We have performed an informal study of user navigation in applications with and without constant information density. We suggest that designers take density into account when designing applications to avoid biasing user navigation in unexpected ways.

KEYWORDS: Clutter, information density, interactive graphics, information navigation, visual interfaces, visualization, zoomable interfaces

INTRODUCTION

Multiple studies have shown that clutter in visual representations of data can have negative effects ranging from decreased user performance to diminished visual appeal. For example, Phillips and Noyes demonstrated that reducing visual clutter improved map reading performance [15]. Similarly, Springer showed that directory assistance operators located targets more quickly on screens with less information [16]. For a review of a number of other studies, see [20].

A number of visualization systems have been proposed that address the clutter problem by allowing users to view detail selectively. Early work includes fisheye views and the Spatial Data Management System (SDMS) [6,9]. More recent paradigms include the Pad zoomable interface, Magic Lenses, and DataSplash [14,3,1].

In this paper, we describe an extension to DataSplash that gives users feedback about the information density of applications as they are constructing them. We also introduce an extension that suggests improvements to existing visualizations. In this way, we help users create interactive applications that display the appropriate amount of detail at all times. We call our system VIDA (Visual Information Density Adjuster).

Previously, we introduced a direct-manipulation interface for constructing zoomable database visualizations [1]. This interface has been implemented on top of the POSTGRES object-relational database management system [17] and released as the DataSplash software.[1] In DataSplash, objects appear in a two-dimensional canvas. Users view the canvas as if with a camera that moves in three-dimensional space but always points straight down at the canvas. Users can pan across the canvas (changing the *x,y* location of the camera). Users can also zoom in and out above the canvas (changing the *z* location, or *elevation*, of the camera). Because a given set of objects looks different when seen from different elevations, a visualization that is appealing at one elevation is likely to be unappealing at another. Therefore, DataSplash objects change representation as users zoom closer to them. For example, when a user zooms closer to a circle representing a city, the name of the city may appear next to the circle. DataSplash provides a unique mechanism, the layer manager [23], which allows users to visually program the way objects behave during zooming. The resulting program is called an *application*. Screenshots and further detail appear in the next section.

Although users generally respond positively to the layer manager, we observe that they have difficulty constructing applications that display an appropriate level of detail at all elevations. To understand this, consider the simplest possible application – one in which the representation of the objects never changes (scaling aside) as the user zooms in and out. The *display* seen by the user is a fixed-size

[1] This software can be obtained at http://datasplash.cs.berkeley.edu/.

viewport onto an underlying, or native, coordinate space defined by the *x,y* values of the objects. The objects never change their native *x,y* position, so object density in the native space obviously never changes. However, any change in elevation implies a change in the area of the native space visible in the display, which implies (in general) that the display contains a different number of objects. This in turn implies a change in display object density. As a result, the same visualization can be appealing at one elevation and cluttered at a higher elevation. Unfortunately, users have difficulty extrapolating from the view at one elevation to the view at another; multiple object representations can make density prediction even harder. Therefore, poor visualization quality may result if users do not visit many elevations (checking for appropriate detail) whenever they modify applications. This is a tedious, highly iterative process. Furthermore, users of current systems must visually and subjectively judge "appropriate" density.

A guiding principle that addresses this issue can be derived from the *Principle of Constant Information Density*, drawn from the cartographic literature [5,18]. This principle states that the number of objects per display unit should be constant. A more general formulation posits that the amount of information (as defined by metrics discussed below) should remain constant as the user pans and zooms. To maintain constant information density, either (1) objects should be shown at greater detail when the user is closer to them, or (2) more objects should appear as the user zooms into the canvas, or (3) both.

We define a *well-formed* application as one that conforms to the Principle of Constant Information Density. By definition, as the user pans and zooms in a well-formed application, the number of objects visible in the display should remain constant. We present a system that interactively guides users in the construction of well-formed applications.

Indirectly-related work has been done in a number of areas. Previous work has examined appropriate amounts of information density for specific character displays, user interface screens, or images (the equivalent of a fixed elevation in our system). Useful summaries appear in [8,20]. Other work has considered the layout of objects at multiple granularities [10]. However, the layout problem detailed in this work has different objectives, requiring that no objects overlap and that the minimal amount of space be wasted. Further, researchers in the area of map generalization have studied automated generation of maps of given scales, although with limited success [4]. The Multi-scale Tree takes a different approach [5]. Given a set of maps produced (manually or with a computer-assisted tool) at different scales, it automatically produces different views of the data as the user zooms. These views have a constant number of active pixels. However, the system does not support interactive construction of visualizations, gives end users and developers no direct feedback about the density of the different levels, and gives end users and developers no direct control over the final presentation.

We believe our system is the first environment that interactively guides users in the construction of applications with constant information density. It has the added advantages of being a direct manipulation interface and of producing general-purpose applications, rather than being limited to a specific domain such as cartography.

We have conducted an informal study of user response to interactive visualizations with varying densities. Our results, though highly preliminary, suggest that users avoid higher density elevations in preference for lower density elevations and that users pan more frequently in displays that have lower density. We propose that application designers use our DataSplash extensions to ensure constant information density and thereby minimize unexpected user navigation patterns.

In the next section we describe the DataSplash database visualization environment. We then discuss how we have modified this environment to provide visual feedback about the density of applications. In the following section we propose a semi-automated mechanism for the modification of layers. Next, we describe our pilot study. Finally, we discuss future work and conclude.

THE DATASPLASH ENVIRONMENT

In this section, we describe the existing DataSplash environment. In DataSplash, all objects in a canvas are organized into *layers*. Each object is a member of exactly one layer. Each layer is associated with exactly one database table. Each row in the table is assigned an *x,y* location in the canvas, *i.e*., the rows are scattered across the canvas, giving an effect similar to a scatter plot. The *x,y* locations are derived from data values in the rows. For example, if the user has a table of United States cities with latitude and longitude columns, *x* and *y* can be assigned to the longitude and latitude values of each city.

At any point, the user can create an object in DataSplash's paint program interface and duplicate that object for every row in the database table. As a result of this duplication operation, a copy of the object appears at the *x,y* location of every row in the table. The effect is like splashing paint across the canvas, coating every scattered row. The user may also associate display properties of objects with columns in the table, *e.g*., height, width, color, and rotation of each splash object can be derived from values in the columns of its row. Continuing our example of a visualization of United States cities, the user may specify that a circle is to be drawn at the *x,y* location of each city. The user may further specify that the radius of each circle be proportional to the population of that city to create a display such as that seen in the left side of Figure 2.

Users can pan and zoom above the resulting two-dimensional canvas. When they zoom, they change their elevation above the canvas. Elevation is expressed as a

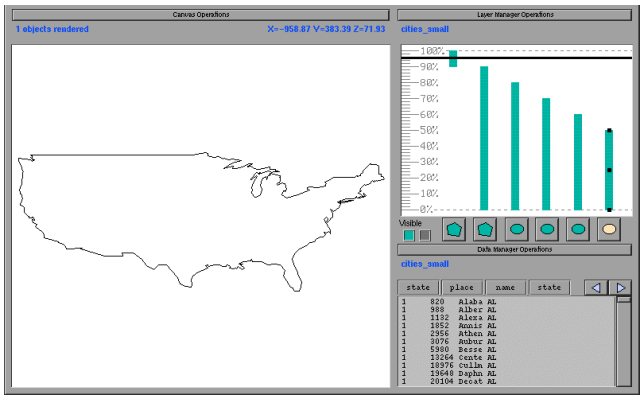


Figure 1: United States cities application seen from a high elevation.

The layer manager appears in the white rectangle in the upper-right. The horizontal line (the elevation bar) indicates that the user is at a high elevation. The only layer active at this elevation is the United States outline layer. This layer is rendered in the display on the left.

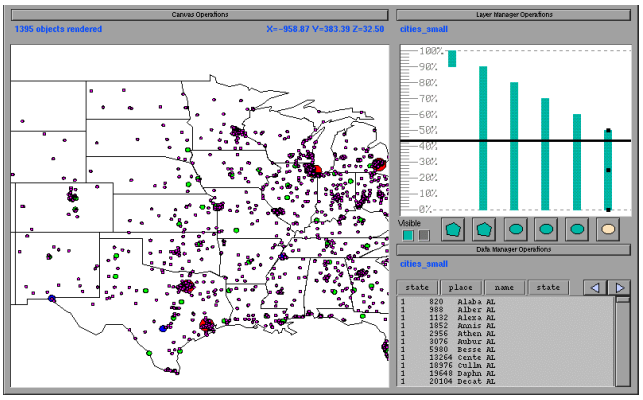


Figure 2: United States cities application seen from a low elevation.

As can be seen from the position of the horizontal elevation bar in the layer manager above, the user has zoomed to a lower elevation. At this elevation, the state outlines and all city circles layers are active. Therefore, the objects associated with these layers are visible on the left.

percentage of a user-specified maximum elevation. DataSplash allows users to control the range of elevations at which each layer is rendered. To this end, each layer appears as a vertical bar in a layer manager. The top of the layer bar represents the highest elevation at which objects in the layer are rendered. Similarly, the bottom of the layer bar represents the lowest elevation at which objects in the layer are rendered. The user's current elevation is shown with a horizontal *elevation bar*. Any layer bar that is crossed by the horizontal elevation bar is considered to be *active* and objects in the corresponding layer are rendered. An icon of the type of object displayed by each layer appears in the button below its layer bar.

Users can graphically resize layer bars in the layer manager. In addition to resizing layer bars, users can also shift them up and down and add or delete new layer bars. These operations are performed in the same way as in traditional paint programs, *e.g.*, to resize a layer bar, the user drags a resize handle. DataSplash has a number of additional features, but we do not discuss them here for the sake of brevity.

Figures 1 and 2 show the display and layer manager of a sample application. The application contains six layers. The first layer contains an outline of the United States. The second layer contains outlines of each the United States. Note that these two representations are mutually exclusive, *i.e.*, their elevation ranges are disjoint, so only one will be visible at a given time.

The third through sixth layers are based on city data. The data has been segmented according to population size; the third layer contains cities with the highest population, the fourth layer contains slightly smaller cities, *etc*. Each city in each entry is drawn in a circle and cities with higher populations are drawn as larger circles.

In Figure 1, the user is at a high elevation. Only the United States outline is rendered at this elevation. In Figure 2, the user is at a lower elevation. At this elevation, the state outlines and all city circles are drawn.

As a final note on DataSplash functionality, DataSplash provides portals. Portals are windows that open onto other canvases. DataSplash users can automatically generate a portal for every row in a database table. For example, the user can easily specify that each city in a visualization of the United States should have a portal that goes to a map of that city. A portal history mechanism allows users to go backwards and forwards between canvases.

DENSITY FEEDBACK

Users of the original DataSplash layer manager find it difficult to construct visualizations that have appropriate detail at all elevations. In this section, we describe how users can express their preferences for application information density, how the system lets them know when these preferences are not being met, and how the user can correct such conditions. We then briefly discuss alternative schemes for density measurement.

We begin by considering a distribution of data that is uniformly distributed in the *x* and *y* dimensions. At the end of this section, we discuss skewed distributions.

Measuring Information Density

We have designed a software framework in which we can explore generalizations of the Principle of Constant Information Density. Since we are unlikely to anticipate the needs of all applications, we express all of our notions of information density in terms of extension and configuration interfaces. We provide two such interfaces, one to measure density and the other to bound it.

Information density metrics are expressed using density functions. Expert users may define new density functions to supplement those already included in the system. (These functions are currently compile-time extensions.) Density functions return the associated density metric value for a given layer at a given elevation.

The system maintains maximum and minimum bounds on the cumulative density (*i.e.*, the density aggregated across all visible layers). These bounds, which can be modified by the expert user at run-time, define a range of acceptable densities; therefore, rather than being literally constant, the application's information density at each elevation is expected to fall within this range. Defining acceptable density in terms of a single constant value would require a change to the displayed information for each change in elevation. Note that the system does not enforce the density bounds. Instead, it provides users with feedback using the mechanisms described below.

Providing Visual Density Feedback

We have modified two of the display objects contained in the layer manager so that their visual properties give the user an indication of the application's information density. Specifically, we have changed the shape of the layer bars and the color of the layer manager trim to provide such feedback.

First, the width of each layer bar now reflects the density of the corresponding layer at the given elevation. The original DataSplash layer manager does not associate the layer bar width with any property of the layer. We have extended the layer manager so that the width of a layer bar at a given elevation is exactly proportional to the layer's density at that same elevation. The scale is defined in terms of a maximum layer bar width, such that a single layer bar of maximum width would have (by itself) 100% of the maximum cumulative density. (This implies that the cumulative layer bar widths at a given elevation in a well-formed application are no greater than this maximum width.)

Exceptional conditions can occur because the system does not enforce the cumulative density bounds. We crop layer bars at the maximum width to allow the bars to have fixed horizontal spacing. We also enforce a minimum width to prevent layer bars from becoming invisible.

Second, the layer manager now relates the cumulative density value at each elevation to the density bounds. Notice the tick marks along the left side of the layer manager display in Figure 3. There are three possible conditions for a given elevation: it may lie within the density bounds, it may fall below the minimum density bound, or it may exceed the maximum density bound. Each tick mark is assigned one of three colors to indicate which condition pertains at a given elevation.

Figure 3 shows a visualization of selected companies from Fortune Magazine's Fortune 500 and Global 500 lists.

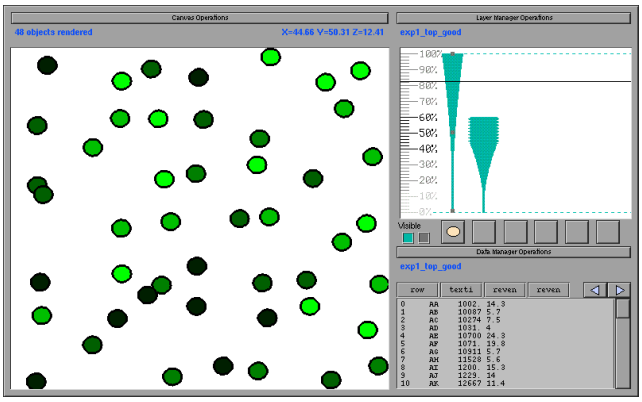


Figure 3: A visualization of selected companies from the Fortune 500 and Global 500 lists.

The width of each layer bar at a given elevation now represents the density of the layer at that elevation. The minimum and maximum density bounds are set to 10 and 100 objects, respectively. The colors of the tick marks on the left side of the layer manager indicate the density values at given elevations. Elevations 40%-60% are too dense, elevations 14%-38% and 62%-100% have appropriate density, and elevations 0%-12% are too sparse.

These companies are displayed as circles in an interactive scatterplot. The *x* axis represents the percent profit (profit dollars divided by revenue dollars) of the company during a given year. The *y* axis represents the number of employees of the company. The color of the circle represents the profit of the company in dollars.

The density function in the current implementation measures density by counting the number of objects visible in the display at a given elevation. This metric is essentially that of Töpfer's original Radix Law [18], from which the Principle of Constant Information Density is derived; as we have mentioned before, many other density metrics are possible.[2]

Figure 3 highlights some of the interesting properties of the object density metric. These properties illustrate the point we raised in the introduction: density metrics are difficult to extrapolate. First, note that the sides of the layer bars in Figure 3 have a rotated parabolic shape. To understand this, assume for the moment that the number of objects per unit area in the native data space remains the same. The area of the native space visible in the display increases quadratically as the elevation increases. Consequently, the number of objects visible in the display, and therefore the

[2] As an aside, this metric can be represented as follows in a Space-Scale Diagram [7]: each object can be considered as a point. Each point defines a ray that passes through the scales at which the object is visible. As the viewport is moved throughout the diagram, the number of rays that penetrate the viewport should remain constant.

object density, increases quadratically as well. Second, observe that the rate of change in width is more pronounced for the layer bar on the right. Because the right-hand layer bar contains more objects, its density increases more quickly. Our extensions graphically illustrate these properties, eliminating the need for users to compute them mentally.

User Interaction with the New Layer Manager

Based on the feedback provided by the extensions described above, users can modify applications as they are constructing them. As the layer manager is currently implemented, there are two primary ways users can change the density of their applications. (In the next section, we propose an extension to the layer manager with which users can prompt the system to create layers with specified densities.) First, users can modify the layer manager, *i.e.*, change the elevations at which layers are active. Second, they can change the contents of layers.

In the DataSplash environment, users may graphically modify the layer manager in two ways. They may adjust the top or bottom elevation of a layer bar. When this happens, the shape is extended (according to the width calculation function) as the user makes the adjustment. Users may also graphically drag the entire layer bar up and down to shift the elevation range at which the layer is visible. When this happens, the shape of the bar changes as the user drags it. Additionally, as the user modifies the bar in either of the ways just described, the colors of the tick marks change to reflect the modification. Intuitively, the user moves the bar around, trying to maximize the number of green tick marks (by default our interface uses green to indicate appropriate density).

Users can modify the contents of layers in two ways. First, they may use the paint program interface to modify the contents of a layer. For example, to modify the number of objects, they may add or delete objects. If other density metrics (*e.g.*, the number of vertices) are considered, a variety of other operations, such as changing the shapes or colors of objects, affect the density values as well.

The second way users may modify the contents of a layer is by using the visual select and join mechanisms described in [13]. These operations affect the number of rows in the table associated with the layer, thereby affecting the number of objects rendered. When the user modifies the contents of the layer using either the paint program interface or the visual select and join mechanisms, the layer bars and tick marks are automatically updated to reflect the change.

As an additional note, the extensions to the layer manager not only provide the user with feedback about specific applications, but teach the user about the properties of density functions in general. For example, a user who has not thought explicitly about the relationship between zooming and the number of visible objects receives an intuitive introduction to the concept by using our extensions to the DataSplash environment.

Density Metrics

Our system currently supports two density metrics, number of objects and number of vertices. There are a number of other metrics that could be used, *e.g.*, Tufte's data density [19]. For a thorough review see [12]. Because the focus of our work is on maintaining constant information density for a given metric rather than on determining good density metrics, we have not yet implemented any additional metrics. However, the interface is independent of the density metric and we have designed the system such that expert users may register their own density functions.

It is our belief that the interface will be particularly useful in teaching application developers about the properties of different density metrics under zooming conditions. For example, the ink metric (the number of live pixels) is not elevation-sensitive. To see this, imagine that the canvas contains a chessboard and that black pixels are live. Since half the pixels are white and half are black, 50% of the pixels are live. Now imagine zooming closer to the chessboard. The view changes considerably, but the pixel distribution remains the same. Therefore, ink is probably not an appropriate density metric for zoomable applications.

Non-uniform Data

Recall that the width bars and the cumulative density values are based on the *average* density of each layer. For this reason, we expected them to be most useful for uniform data sets. While this prediction turned out to be correct, we have been pleasantly surprised by their successful application to non-uniform distributions.

Figure 4 shows the application shown in Figures 1 and 2 augmented with width bars and tick marks colored according to cumulative density. We used the feedback mechanisms to improve the visualization, yielding Figure 5. While this improved visualization contains some regions that are too dense and others that are too sparse, it is still a significant improvement over the original visualization. We have also augmented VIDA with additional mechanisms to handle non-uniform distributions. These are discussed in [21].

SEMI-AUTOMATED ADJUSTMENT OF LAYER DENSITY

In the previous section, we described how users can manually modify their visualizations to have appropriate density. We have designed and are in the process of implementing a mechanism with which users can semi-automatically adjust the density of a given layer. We first describe the way the system creates layers of a desired density. We then describe the interface to this mechanism.

Calculating Layers of a Desired Density

In this section, we describe *modification functions* that can be applied to a layer to modify its density. These functions operate on one of two components of the layer, the data

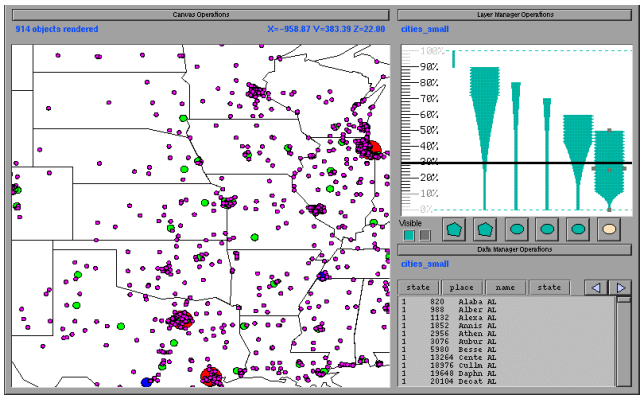


Figure 4: A cluttered application.

The width bars and tick marks suggest that this application is cluttered not only at the current elevation, but at other elevations as well. The four layer bars to the right reference cities of different populations. The shapes of the bars indicate that there are few cities in the higher layer bars (the ones associated with larger cities). They illustrate that there are many cities in the two rightmost layers (the ones associated with smaller cities).

rows in the database table and the graphical representation of the data. Functions come in pairs, one that decreases density and one that increases density.

To modify the data, DataSplash can create views of the table. Basic views include simple restriction or aggregation queries. Because more complex views may be desirable, DataSplash can also consider views created by expert users as potential modifications to the database table. Modification of the graphical representation is straightforward.

Table 1 presents modification functions that decrease visual density. The table uses the following visualization of cities in the United States as an example. We suppose the user has created a layer based on a table of cities that includes fields for latitude, longitude, and population of each city. The graphical representation of the user-created layer is a gray circle placed at the longitude, latitude location of each city. The circle is assigned a size based on the population of the city. This "original" representation appears in the first row of the table. The visible area is a zoomed-in view of Baltimore and Washington, D.C., in the United States.

The remainder of Table 1 contains seven modification functions. For each modification function, the table presents an example of a specific modification, an example of a density metric affected by that modification (in many cases multiple metrics are affected; for brevity we only identify one per modification function), and the visualization resulting from the application of that modification to the original visualization.

The first three modifications (select, aggregate, and reclassify) apply to the data. When the given select operation is applied, only the largest cities remain visible.

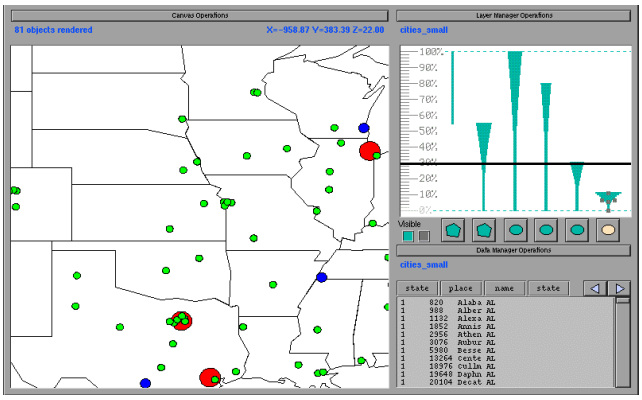


Figure 5: An improved version of Figure 4 (seen from the same elevation).

The user has adjusted the tops and the bottoms of the layer bars based on interactive visual feedback.

When the given aggregate operation is applied, the system aggregates cities by states. Chesapeake Bay can be seen in the resulting visualization. Both select and aggregate reduce the number of visible objects. The given reclassify operation classifies cities into two groups according to their population; the resulting visualization has fewer sizes than the original.

The remaining four operations (change shape, change size, remove attribute association, and change color) are modifications to the graphical representation of each object. When the shapes are changed from circles to triangles, or when the size of the circles is changed, the amount of ink is decreased. When the association between population and circle size is removed, the data density is decreased. Finally, changing the color may affect the total number of colors in the visualization (in this example, this effect only occurs when other layers are considered as well).[3]

Observe that not every modification function decreases/increases density for a given metric. For example, the size of an object can be decreased (to reduce the amount of ink) or increased (to increase the amount of ink). However, this operation may have little or no effect on the number of objects being displayed. Also observe that in some cases a modification function may in fact decrease density according to one metric while increasing it according to another metric.

[3] The graphical representation manipulation functions represent a fairly comprehensive set in the context of our system. Note Bertin's observation that there are eight variables that provide information in two-dimensional graphics (*x* and *y* position, size, value, texture, color, orientation, and shape) [2]. DataSplash does not modify value, texture, or orientation as the former two do not pertain in our system, and the latter is not clearly desirable.

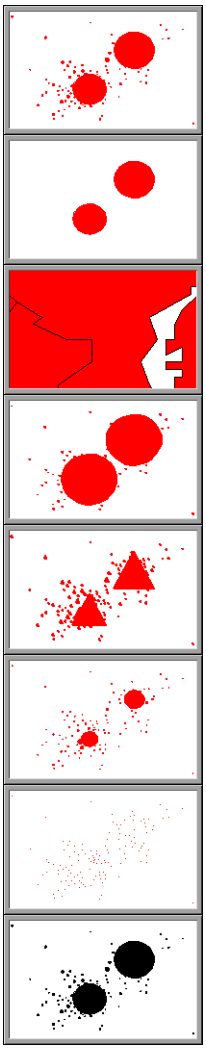
Original visualization	
Select *Restrict to cities with population > n* Decreases number of objects	
Aggregate *Aggregate cities by state* Decreases number of objects	
Reclassify *Assign to population brackets* Decreases number of sizes	
Change shape *Change circles to triangles* Decreases amount of ink	
Change size *Scale circle radius* Decreases amount of ink	
Remove attribute association *Disassociate size from population* Decreases data density	
Change color *Change color from gray to black* Decreases number of colors	

Table 1: Modification functions to decrease density.

The original visualization in the top row of the table shows cities in the United States. The following rows show how that visualization changes in response to the application of various modification functions. Density metrics affected by these modifications are also presented in the table; while a specific density metric is listed as an example for each modification function, others may pertain as well.

As mentioned above, expert users may register new density functions. When they do so, they may also identify modification functions that will affect the corresponding metric. If they perform this task, DataSplash will be able to suggest modifications based on the new density metric.

Interface

Recall that we have extended the layer manager so that the width of a layer bar represents its density. To invoke the semi-automated adjustment of a layer, we propose that users graphically adjust the width of that layer. In response, DataSplash could apply modification functions to generate a number of transformations of that layer that conform to the density correlated with the width specified by the user. DataSplash could present these options to the user in a series of portals. The user could enter each portal and explore it. When they found one they preferred, they could select it as the new version of the layer.

PILOT STUDY

We performed a small, informal study of user navigation behavior in applications with and without constant information density. To our knowledge there have been no similar studies. Therefore, our purpose in this study was to gain intuition about navigation patterns and to identify interesting directions for more formal study. Note that our intent was not to examine density metrics and appropriate values for such metrics, but rather to examine user response to density variance according to a given metric (number of objects).

Visualizations of Fortune 500 and Global 500 data were used in the study. The *x* and *y* axes represented profit % and number of employees, respectively. The data was sampled to create data sets with varying numbers of objects. Based on these data sets, we created a number of visualizations that had varying densities at different elevations.

Users viewed these visualizations in a Java applet that provided simple pan and zoom functionality. The applet was embedded in a World-Wide Web page and recorded data about each of the panning and zooming gestures made by the participants.

Seventy-nine participants were recruited through technical mailing lists and news groups. Participants were told to locate the company they thought had the highest revenue growth (revenue growth is expressed as percent change from the previous year).

In this limited experiment, our results suggest that zooming may be influenced by information density. When selecting an elevation, users appeared to display a preference for layers that had lower visual density. Users also appeared to pan more frequently in layers that had lower visual density. (As a side note, nearly all pan operations were performed at either the highest or the lowest possible elevations.) Further details are presented in [22].

There are a number of obvious limitations to this pilot study. Because we did not control the speed of the applet, the applet was probably less responsive when displaying layers with higher visual density, particularly in the case of computers on which Java runs slowly. As a result, users may have avoided more dense layers because of the poor responsiveness of these layers rather than because of a visual preference. Based on user comments, *e.g.*, one of the users of a visualization that had high density in all layers said the applet was “nicely responsive,” we hope this effect did not significantly affect user behavior. However,

a more formal study should minimally ensure that all layers within a visualization are equally responsive.

Further, while the World-Wide Web was a good way to reach a broad spectrum of users, the participants and the testing conditions were not controlled. Finally, several users stated that they found the task confusing. Such confusion could easily have influenced the results.

Nonetheless, further study seems warranted. If such research does bear out our observations, designers of zoomable applications should take density measurements into account when designing applications. If they do not, users may be influenced by the information density and behave in ways not intended by the designer.

FUTURE WORK

We are actively pursuing a number of issues related to this work, as detailed in this section.

Automated Adjustment of the Layer Manager

In some cases, a poorly-formed application can be converted to a well-formed application simply by adjusting the elevation ranges of existing layers. Calculating the solution is, however, NP-Complete [22]. We are examining pseudo-polynomial algorithms that will automatically adjust the elevation ranges of layers to create a well-formed application.

Automatically-generated Transformations

Previously, we described a system that suggests modifications to individual layers. We have also designed an extension that suggests improvements to entire applications [22]. Users can indicate that they wish the system to present them with a number of well-formed transformations of their application. This extension intentionally generates disparate transformations so that the user has a broad range of options from which to choose. These options are presented in a single *transformation canvas*. The user can browse this canvas using the system-provided navigation facilities, *i.e.*, the user can pan, zoom, go through portals, and use the portal history mechanism. In this way, the user can explore the options provided by the system. An example based on a visualization of the cities in the United States appears in Figure 6. As an additional feature, the system could try to satisfy multiple density metrics simultaneously. While related work has considered automating the design of presentations, it has not explicitly used density as a guiding criterion [11].

Movement Optimization

As users pan and zoom, they may require less detail on the screen than when they are not moving. The user interface should allow the user to differentiate between appropriate detail for movement versus still conditions. For example, the user may require fewer objects while panning and zooming than when viewing a still image. This information can be used to reduce rendering time.

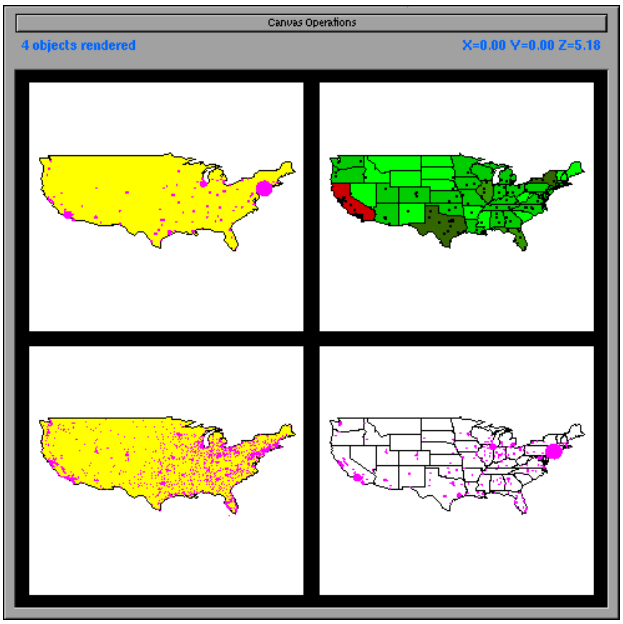


Figure 6: Automatically generated transformations of a user-created visualization.

The user has requested transformations of a visualization of cities in the United States. Each automatically generated transformation appears in a portal. Note that as the user pans and zooms, the contents of the portals will change dynamically, e.g., new layers may become visible. The user may therefore view the transformations simply by navigating in the transformation canvas. Alternatively, they may enter one or more of the portals to explore it more fully. The portals contain disparate transformations. The option in the upper-right is particularly interesting. Circles in the original application were associated with population. In this transformation, the association between circle radius and population has been removed. However, an association between state color and population has been added.

User Studies and Density Metrics

Further studies of user response to applications with constant information density are plainly warranted. Additionally, although we are currently focusing on preserving constant information density for a given metric rather than comparing density metrics and studying appropriate values for such metrics, such studies would plainly be useful. A formal taxonomy of density metrics would also be of significant interest.

CONCLUSIONS

We have introduced the notion of well-formed applications, ones that display an appropriate amount of information (as defined by user-parameterized constraints) at any given elevation. The notion of well-formedness applies to other visualization systems that support multiple representations, *e.g.*, Pad [14]. We have introduced a system, VIDA, that helps users construct well-formed applications in the DataSplash database visualization environment, both by

providing visual feedback on application density and by suggesting modifications to user-constructed applications. We have also conducted a pilot study that suggests that information density affects user navigation and should therefore be taken into account during the construction of visualizations.

ACKNOWLEDGMENTS

We are grateful to Paul Aoki for many helpful discussions and suggestions. We thank Joe Hellerstein and Ray Larson for helpful comments on earlier versions of this paper. We would also like to acknowledge the DataSplash implementors, Michael Chu, Mark Lin, Chris Olston, and Mybrid Spalding. We thank Vuk Ercegovac for developing the Java browser for the pilot study. This research was sponsored by NSF under grants IRI-9400773 and IRI-9411334.

REFERENCES

1. Aiken, A., *et al*., "Tioga-2: A Direct Manipulation Database Visualization Environment," *Proc. 12th Int'l Conf. on Data Engineering*, New Orleans, Louisiana, Feb. 1996, pp. 208-217.
2. Bertin, J., *Semiology of Graphics*, translated by Berg, J., The University of Wisconsin Press, Madison, WI, 1983.
3. Bier, E., *et al*., "Toolglass and Magic Lenses: The See-through Interface," *Proc. ACM SIGGRAPH 1993*, Anaheim, Calif., Aug. 1993, pp. 73-80.
4. Buttenfield, B.P. and McMaster, R.B. (eds.), *Map Generalization: Making Rules for Knowledge Representation*, Longman, London, 1991.
5. Frank, A., and Timpf, S., "Multiple Representations for Cartographic Objects in a Multi-scale Tree - An Intelligent Graphical Zoom," *Computers & Graphics*, Nov.-Dec. 1994, 18(6):823-829.
6. Furnas, G.W., "The FISHEYE View: A New Look at Structured Files," Bell Laboratories Tech. Report, Murray Hill, New Jersey, 1981.
7. Furnas, G.W. and Bederson, B.B., "Space-scale Diagrams: Understanding Multiscale Interfaces," *Proc. ACM SIGCHI 1995*, Denver, Colorado, May 1995, pp. 234-241.
8. Galitz, W.O., *User-interface Screen Design*, QED Pub. Group, Boston, 1993.
9. Herot, C.F., "Spatial Management of Data," *ACM Trans. Database Sys.*, Dec. 1980, 5(4):493-513.
10. Ioannidis, Y., *et al*., "User-Oriented Visual Layout at Multiple Granularities," *Proc. 3rd Int'l Workshop on Advanced Visual Interfaces*, Gubbio, Italy, May 1996, pp. 184-193.
11. Mackinlay, J., "Automating the Design of Graphical Presentations of Relational Information," *ACM Trans. Graphics*, Apr. 1986, 5(2):110-141.
12. Nickerson, J.V., *Visual Programming*. Ph.D. Dissertation. New York Univ., New York, 1994.
13. Olston, C. and Stonebraker, M., "VIQING: Visual Interactive QueryING," *submitted for publication*, Sept. 1997.
14. Perlin, K., and Fox, D., "Pad: An Alternative Approach to the Computer Interface," *Proc. ACM SIGGRAPH 1993*, Anaheim, Calif., Aug. 1993, pp. 57-64.
15. Phillips, R.J. and Noyes, L., "An Investigation of Visual Clutter in the Topographic Base of a Geological Map," *Cartographic J.*, Dec. 1982, 19(2):122-131.
16. Springer, C., "Retrieval of Information from Complex Alphanumeric Displays: Screen Formatting Variables' Effects on Target Identification Time," *Cognitive Engineering in the Design of Human-Computer Interaction and Expert Sys.* (Proc. 2nd Int'l Conf. on Human-Computer Interaction, Honolulu, Hawaii, Aug. 1987, Vol. II), pp. 375-382, G. Salvendy (ed.), Elsevier, Amsterdam, 1987.
17. Stonebraker, M. and Kemnitz, G., "The POSTGRES Next-generation Database Management System," *Communications of the ACM*, Oct. 1991, 4(10):78-92.
18. Töpfer, F., and Pillewizer, W., "The Principles of Selection, A Means of Cartographic Generalization," *Cartographic J.*, 1966, 3(1):10-16.
19. Tufte, E.R., *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Conn., 1983.
20. Tullis, T.S., "Screen Design," *Handbook of Human-Computer Interaction*, M. Helander (ed.), Elsevier Science Publishers B.V., Amsterdam, 1988.
21. Woodruff, A., Landay, J., and Stonebraker, M., "End-user interaction with clutter reduction techniques," Tech. Report CSD-98-979, Univ. of California, Berkeley, Calif., Jan. 1998.
22. Woodruff, A., and Stonebraker, M., "VIDA: Visual Information Density Adjuster," Tech. Report CSD-97-968, Univ. of California, Berkeley, Calif., Sept. 1997.
23. Woodruff, A., *et al*., "Navigation and Coordination Primitives for Multidimensional Browsers", *Visual Database Systems 3: Visual Information Management* (Proc. 3rd IFIP 2.6 Working Conference on Visual Database Systems, Lausanne, Switzerland, March 1995), pp. 360-371, S. Spaccapietra and R. Jain (Eds.), Chapman & Hall, 1995.