A Framework for Focus+Context Visualization

Staffan Björk, Lars Erik Holmquist and Johan Redström
PLAY: Applied research on art and technology
The Viktoria Institute, Box 620, SE-405 30 Gothenburg, SWEDEN
{bjork,leh,johan}@viktoria.informatics.gu.se
http://www.viktoria.informatics.se/play/

ABSTRACT

Focus+context visualization techniques aim to give users integrated visual access to both details and overview of a data set. This paper gives a systematic account of such visualization techniques. We introduce the notion that there are different levels of information visualization, with focus+context being a second-level visualization, and illustrate this with examples. We then provide a formal framework for describing and constructing focus+context visualization and relate this to the examples. A description of a software framework based on the principles of the theoretical framework follows, and we give some examples of how different focus+context visualization applications have been constructed using this framework. Finally, we discuss the implications of the formal framework and outline some future work in this area.

Keywords

Focus+context visualization, information visualization, fisheye views, formal methods, theory

INTRODUCTION

Information visualization is widely acknowledged as a powerful way of helping users make sense of complicated data, and a great number of methods for visualizing and working with various types of information have been presented. However, all information visualization techniques will have to comply to one inherent limitation: they will need to limit themselves to the available area of a computer screen. A common solution to this problem is to provide some kind of movable view-port to the data, which can be controlled through the manipulation of scrollbars or other means. Zooming interfaces have also been introduced to let users control the amount of data shown, e.g. [3]. Sometimes, however, it might be important to give users access to both overview and detailed information at the same time; such techniques include [21], with separate areas for overview and detail-on-demand information.

Here, we will concentrate on a certain family of techniques, that attempt to integrate both detail and overview on the same display area in an effort to not divide the user's attention. Some terms which have been used for such techniques include *fisheye views*, *distortion-based presentations* and *detail-in-context visualizations*. In the following we will use the term *focus+context visualizations*, which is wide enough to encompass all the properties we will be discussing.

RELATED WORK

Although the origin of focus+context visualization can be traced back to non-interactive distortion-based techniques for visualization of map data [14], the first computer-based interactive method was introduced with the *FISHEYE View* [8], more known as the *Generalized Fisheye View* [9]. This original fisheye notion was in fact a general interaction framework for information filtering according to the user's current point of interest in the material, rather than a specific visualization technique, and was shown to be applicable to various types of data, notably structured programs and tree structures. (Some confusion has been the result of several other techniques using the term "fisheye", and currently fisheye visualization is often more closely associated with distortion-based techniques that give the graphical impression of the fisheye-lens of a camera.) In connection with the Generalized Fisheye View, important concepts such as the *Degree of Interest* (DoI) function and the *Level of Detail* (LoD) were introduced.

Another early interactive example of focus+context visualization was the *Bi-Focal Display* [29], where a graphical focus+context display was applied to a calendar display, introducing distortion in the horizontal dimension. A somewhat similar technique, the *Perspective Wall* [20], used a 3D perspective to achieve the same effect. The *Document Lens* [24] developed the concept further by combining a perspective view with a magnifying-glass effect to give combined detail and overview presentation of a document. Other techniques that use various forms of distortion to display two-dimensional images or maps include the *Graphical Fisheye View* [25] and *Rubbersheet View* [26], and forays have been made into extending such techniques to three dimensions [7]. *Flip zooming* [11] was developed to visualize sequentially ordered material, and it has been used for visualizing documents [12] and hierarchically ordered image collections [13]. Techniques developed specifically for visualizing graphs and hierarchies include *Hyperbolic Trees* [18], the *Continuous Zoom* [2], and *Cone Trees* [23].

Among papers seeking to classify or formalize focus+context techniques, [19] is probably the most widely cited. It gives an overview of the various techniques and provides a unifying theory in the form of a rubber-sheet analogy. [10] introduced *Space-scale diagrams* as a framework for analysis of multi-scale (or zooming) interfaces, and showed that such diagrams could also be used for describing focus+context techniques. So-called *Non-Linear Magnification Fields* [16] have been introduced as an abstract representation of distortion-based magnification techniques, and these have since been more generally applied to the problem of detail-in-context visualization [17]. [28] introduced several dimensions of transformation, *X, Y, Z,* and *W,* where the W-transformation corresponded directly to the Generalized Fisheye View.

THE FOCUS+CONTEXT VISUALIZATION PROCESS

Levels of Representation

When describing information visualization, it is often sufficient to describe the underlying data, how the data is represented and what manipulation or interaction this representation will allow [6, 30]. Manipulation can be either manipulating the data itself, or, if the visualization is interactive, manipulating the way in which the data is presented. Focus+context visualizations can also be described in this way. However, we argue that it is useful to describe a focus+context visualization as a *second-level visualization*, i.e. a visualization of a visualization.

To clarify this, consider the rubbersheet metaphor as described in [19]. Here, a focus+context visualization is compared to a sheet of rubber that has an image of some sort printed on it, e.g. a map or document. The rubbersheet is tied up in a rigid frame, representing the fixed size of the screen. Magnification of a certain area can then be achieved by stretching part of the sheet, and due the limited space available within the frame, other areas will shrink correspondingly. According to our distinction, we would say that manipulating a second-level visualization corresponds to manipulating the rubbersheet itself. Manipulating the first-level visualization, however, would correspond to some manipulation of what information is actually printed on the sheet.

This distinction is important, since in many cases it might be interesting to be able to perform manipulations at *both* levels of visualization. Separating the levels in this way will make the different types of interactivity clearer, and will also make it easier to account for how we can combine different focus+context visualizations with different types of information visualization techniques. In the following, some examples will be given to illustrate this.

Example 1: Structured high-level computer program

Here, the data consists of a sequence of code that represents a computer program. One way to visualize and interact with a program would be to show it as a succession of lines, indented according to their place in the program structure, in which the user can scroll up and down. The program might also be represented as uniformly sized pages of text, which the user can switch between (this would reflect the way the program would look when printed on a laser printer and might be useful when making changes according to comments written on a print-out). We might also isolate the various components of the program, such as functions and data structures, and show these as nodes in a hierarchically ordered tree; this would represent the inherent hierarchical structure of the program.

On any of these visual representations, we can then apply a focus+context visualization technique. In the case of lines of indented text we might choose to use the Generalized Fisheye View [9]. If we have text separated into uniformly-sized pages, we might use the Document Lens [24] or the Zoom Browser [12]. If we choose to have the program represented as a set of hierarchically ordered objects and functions, we might want to use the Hyperbolic Tree Browser [18] or Cone Trees [23].

Considering the interaction that might be possible in the system, users should of course be able to manipulate the data itself by making changes in the code; these changes will directly affect the data, and will be reflected in the first-level visualization as changes in the text, indentation, hierarchical structure, etc. But users can also manipulate the focus+context visualization by means of changing the focus, increasing or decreasing the degree of magnification, etc. These changes are occurring in the second-level visualization, and will not change the actual data, only the way it is shown to the user.

Example 2: Geographical elevation data

When creating a geographical model of a certain area, the data can be described as a number of data triplets, with the two first values representing coordinates in the plane, and the third component representing the altitude. A common way to represent this type of data is to create a graphical map in two dimensions, where gray-scales or colors indicate the altitude. In some cases, however, it might be useful to use a table of the underlying numerical values, perhaps for working with the data in a spreadsheet application. Alternatively, we might create a fully 3-dimensional representation of the data, which could be rotated and viewed from different angles.

A 2-dimensional map is the most common representation used for this kind of data in focus+context visualization, as it is suited to for many distortion-based techniques, such as the Rubbersheet View [26] and the Graphical Fisheye View [25]. A very different, but still valid, type of focus+context view can be given of the tabular data with a technique such as the Table Lens [22]. In the case of a fully three-dimensional representation there may be a natural focus+context effect in the use of perspective: the parts that are close to the point of view will be more into focus than parts further away. However, for a more generalized focus+context view of 3-dimensional data, methods such as those presented in [7] might be used.

Considering the interactivity, if the map data is only to be viewed as-is, users might only interact with the information at the focus+context, i.e. second, level of visualization, by changing the focus and magnification, etc. However, if the user is going to change the data in some way, say do some manual corrections to the survey values, this interaction will take place at the first level, and be directly reflected in the table, map or other underlying visual presentation.

A FORMAL DESCRIPTION

We will now describe the focus+context visualization process in a more formal manner.

Visualizations

Any information visualization starts with a set of data, i.e. the information to visualize. A visual representation of this data set – or some set of data derived or constructed from this set – can be constructed based on the values or inherent structures of this data. Let us define this information visualization as:

IV ([D], V, I)

Here, **IV** is some form of information visualization in which **[D]** is the set of underlying data, **V** is how the data is presented visually, and **I** the interactivity or manipulation possible in the information visualization.

We must here distinguish between two different ways of manipulating **IV**. If **I** affects **[D]**, we can use **IV** according to **I** to manipulate the underlying data set **[D]**. This would for instance correspond to making changes to the data in a spreadsheet or a word processor. A different mode of manipulation is when **V** is affected by **I**, i.e. when a user can manipulate **IV** in order to change the way **[D]** is presented. An example of this is the case with visual information searching through dynamic queries [27], where the user can customize the visualization to show certain aspects of the data, without making any changes to the underlying data set.

Second-level Visualizations

If we instead of using **[D]** in the formula above insert some information visualization **IV**, or rather, a structure of visualizations, **[IV]**, we will have a second-level visualization, **IV'**:

IV' ([IV], V', I')

Here **IV'** is the new second-level visualization, **[IV]** is the underlying set of information visualizations, **V'** is the second-level visual component, and **I'** is the interaction or manipulation possible in this visualization. This formula will now enable us to import any information visualization set **[IV]**, with its constraints **V** and **I** for how the structure can be visualized and changed, and apply any suitable new visualization and interaction method to this representation. Of course, in the same way as certain representations are only suited to certain types of data, **[IV]** may have to meet some constraints in order to fit into a certain second-level visualization **IV'**.

Focus+Context Visualization

We will now describe focus+context visualization as an instance of a second-level visualization **IV'**. It will take any set of information visualizations **[IV]** as its input, given that **[IV]** is compatible with the focus+context visualization technique in question. We apply a visual presentation component **V'** and some interaction **I'** that reflects the focus+context method chosen. As we incorporate some underlying information visualization **[IV]** rather than some data set **[D]**, we can focus on the aspects of **V** and **I** that are unique to focus+context techniques.

Interaction

The most notable aspect of interaction in focus+context visualization is the ability to select a focus and have the presentation changed accordingly. A convention introduced in [9] is to call the point (or rather, object) in focus **'.'** (dot). Now, we can ask how other objects in the underlying visualization **[IV]** are related to **'.'**: given a **'.'** ∈ **[IV],** how important is another object **x** ∈ **[IV]**? According to the same convention, this can be termed the *Degree if Interest*, **DoI.** In order to answer this, we have to describe the relation between **'.'** and **x,** or rather, the "distance" between **'.'** and **x**. The distance will depend on how closely the two objects are related to each other, but also of the individual properties of **x**. In [9] the function *Level of Detail* was used to establish a measure of this distance. The level of detail of an object **x** reflect where in a hierarchical structure it belongs; objects belonging to higher levels (i.e. more abstract) are said to have a lower level of detail, and hence they are more important when providing a general context. Let us use:

W (. , x)

Where **W** is the weighted distance between **'.'** and **x**, or in other words the importance of **x** given **'.'** (where **'.'** and **x** ∈ **[IV]**).

However, there are other ways of controlling how closely related two objects are as well. We might for instance let the user link objects to each other, ensuring that whenever one of them is in focus, the other one will be brought forward as well. We might also allow for other ways of weighting the objects besides using their position in a hierarchy, making it possible for individual objects to have an independent "importance factor" associated with them. Furthermore, we might want to use a tool similar to the focal length on a camera, controlling how big the difference between the focus and context should be. At one extreme the use of such a tool would imply that nothing but **'.'** is seen, and at the other that there is no difference between **'.'** and the rest, i.e. a maximal and a minimal difference between **'.'** and the rest of **[IV].**

Visualization

Given that we know which object is in focus, and how important the other objects in **[IV]** are in relation to it, we can create a visual presentation. As the available resources are limited, some constraints have to be met. This makes it useful to introduce a threshold function, **T**. **T** depends on the size of the screen, **s,** its resolution, **r,** and the computational resources, **c,** available (at least if real-time interactivity should be possible). Hence we have:

T (s, r, c)

The threshold function **T** gives a value of how close an object will have to be to **'.'** in order to be visualized. In order to determine whether a certain object **x** should be visualized or not, the weighted distance **W (. , x)** is compared with **T**:

$$\mathbf{W\ (\ .\ ,\ x) > T}$$

However, in some focus+context techniques objects are never excluded, meaning that **T** is not used to determine whether **x** should be visualized or not (or, alternatively, that **W (. , x) > T** for every **'.'** and $\mathbf{x} \in$ **[IV]**).

W (. , x) can also be used in order to determine which, if any, transformations of **x**'s underlying visual presentation **IV** (which is presented according to **V** in the underlying representation) should be made, e.g. distortion or scaling. For example, **x** can be given an amount of space on the screen proportional to its distance to focus as defined by **W(. , x)** in which case **V'** can be a simple scaling of the image produced by **V**. **W** can also be used to determine where to display **x** in relation to **'.'**, representing **W** with actual distance between objects on the screen.

Besides functions depending on **'.'** and **W(. , x)**, transformations of the underlying representations and rules for screen layout can also be applied. For instance, structural aspects of **[IV]** can be used to determine where on the screen a certain object should be placed. If the objects in **[IV]** are ordered sequentially, say, as the pages in a book, we might want them to be ordered in the same way on the screen, whereas if **[IV]** is presented hierarchically, we would want the focus+context presentation to reflect this accordingly.

APPLYING THE FRAMEWORK

Having defined the formal framework, we can now use it to describe some of the examples presented earlier.

Considering the first example, the structured computer program, we have one set of data that is the code being edited, which we can term **[C]**. We can then choose to have some interactive representations of it: a line-based representation, or one based on discrete uniformly-sized pages of text, or one based on a hierarchically ordered set of components. Let us call them $\mathbf{CV_L}$ (line-based code visualization), $\mathbf{CV_P}$ (page-based), and $\mathbf{CV_H}$ (hierarchical), respectively. Examining the components **I** and **V** of each representation, we see that the visual component **V** in the first case is a long sequence of lines of code, in the second it is a number of sequentially ordered pages of equal size, and in the third **V** is a number of differently sized chunks of code each representing a logical unit of some sort, presented in a tree structure. Similarly, in the first case **I** allows us to move up and down in the sequence of lines; in the second, it will allow us to switch back and forth between discrete pages of code; and in the third, it allows us to navigate the hierarchical structure of the program. If we term these components $\mathbf{V_L}$ (line), $\mathbf{V_P}$ (page) and $\mathbf{V_H}$ (hierarchy), and $\mathbf{I_L}$, $\mathbf{I_P}$, and $\mathbf{I_H}$, respectively, we have the following formulas:

$$\mathbf{CV_L = IV\ ([C], V_L, I_L)} \quad \text{(line-based visualization)}$$

$$\mathbf{CV_P = IV\ ([C], V_P, I_P)} \quad \text{(page-based)}$$

$$\mathbf{CV_H = IV\ ([C], V_H, I_H)} \quad \text{(hierarchical)}$$

We can now insert these representations into a focus+context visualization. Common for all of these will be that the **I** component will allow the user to move the focal point, **'.'**, in some way. In the Generalized Fisheye View, this will be through focusing on a single line; in the Document Lens and The Zoom Browser we can focus on a single page; and the in the Hyperbolic Tree and Cone Tree, we can move a certain point in the hierarchy into focus. These interactions, which we can term $\mathbf{I_L'}$ (line-based interaction), $\mathbf{I_P'}$ (page-based) $\mathbf{I_H'}$ (hierarchical), respectively, correspond directly to the interactive components of the first-level representation.

The visual component **V'** in the various cases has these properties: In the Generalized Fisheye View, only certain lines of code will be shown according to their degree-of-interest, with most detail being shown nearest to the focus; this we will term $\mathbf{V_L'_{DoI}}$ (line-based degree-of-interest view). In the Document Lens the pages surrounding the focus will be distorted according to the combined perspective and optical metaphor used, but will keep their relative position. This we can call $\mathbf{V_P'_F}$ (page-based focus+context view with fixed position). With the Zoom Browser, all surrounding pages will be shrunk to the same size, and rearranged sequentially according to the browser's left-to-right, top-to-bottom convention; this we call $\mathbf{V_P'_S}$ (page-based view with sequential position). Finally, in the Hyperbolic Tree Browser and Cone Trees, the act of focusing on one component will affect how the other components are shown according to their place in the hierarchy, so that components farther away in the hierarchy will be less visible, with close objects more visible. This we will call $\mathbf{V_H'_H}$ (hierarchical view based on hyperbolic geometry) and $\mathbf{V_H'_{3D}}$ (hierarchical view based on 3D-perspective), respectively.

We can now describe any of the focus+context applications in this example in a formal way. For instance, the Generalized Fisheye view (let us call it **GF)** becomes:

$$\mathbf{GF = IV'\ ([CV_L], V_L'_{DoI}, I_L')}$$

In the same way, the Hyperbolic Tree (**HT)** used on our hierarchically ordered program becomes:

$$\mathbf{HT = IV'\ ([CV_H], V_H'_H, I_H')}$$

Using Cone Trees (**CT)** on the hierarchical ordering gives us a similar formula:

$$\mathbf{CT = IV'\ ([CV_H], V_H'_{3D}, I_H')}$$

The other focus+context examples can be constructed according to the same principles.

We can also do some novel combinations. Say that we want to apply the Hyperbolic Tree view to a set of uniformly-sized sequential pages. Since the only structure we have access to is the discrete pages in sequential order, $\mathbf{I_P}$, we will have to base the interaction on this, but the visualization can still be done using hyperbolic geometry. Let us call this new Hyperbolic Tree variant **HT**:

$$\mathbf{HT_P = IV' ([CV_P], V_H'_H, I_P')}$$

Since the visualization is designed to reflect a hierarchical structure, **HT'** might not be of much practical use, but the important point is that such novel applications can be constructed in this framework.

Similarly, returning to the map example, we may term the underlying geographical data **[G]**. If we choose to represent it as a static 2-dimensional map, **M,** we may have a visual component $\mathbf{V_{M2D}}$ (2-dimensional map) but no interaction component (resulting in **I** being empty). We can then apply, say, a Rubbersheet View to this map, with the visual component being that of rubbersheet deformation, $\mathbf{V_R}$, and the interactive component being that of rubbersheet interaction, $\mathbf{I_R}$. The Rubbersheet View (**RV**) visualization of a static map would then be:

$$\mathbf{RV = ([M], V_R, I_R)}$$

Where $\mathbf{M = ([G], V_M, I)}$, and **I** is empty. However, we might want to have an interactive rather than a static map as first-level representation of **[G]**. For instance, if we want to have a zoomable map, being able to zoom in on certain parts for further visualization in the Rubbersheet view, we may have $\mathbf{M_Z = ([G], V_M, I_Z)}$, if $\mathbf{I_Z}$ is the zooming interaction and $\mathbf{M_Z}$ is the resulting zooming representation of the map. This can then be inserted in the Rubbersheet view, resulting in a new variant:

$$\mathbf{RV_Z = ([M_Z], V_R, I_R)}$$

An interesting scenario would be to add some more complex interaction to the first-level representation, say a set of dynamic query sliders [27] to facilitate advanced visual data retrieval. We would then insert the interaction $\mathbf{I_{DQ}}$ for the dynamic query searching, getting the resulting dynamic query-based map visualization $\mathbf{M_{DQ}}$. By applying a Rubbersheet view we would then get a focus+context application which included dynamic query searching of the map data:

$$\mathbf{RV_{DQ} = ([M_{DQ}], V_R, I_D)}$$

This might in fact be quite a useful application, since it will combine an advanced visual query method with the detail and overview supported by the Rubbersheet. Thus, the formal system has been shown to handle both existing focus+context applications, and novel combinations of first- and second-level visualizations.

A SOFTWARE PACKAGE SUPPORTING THE MODEL

As we have seen, it is possible to generate different focus+context visualizations given the same underlying representation, or to apply the same focus+context visualization to a number of different representations, by varying the parameters described in the theoretical framework. This property of the formal description makes it suitable for implementation as a general software platform. We have constructed such a software package, to support the creation of focus+context visualizations of information visualizations consisting of sequentially ordered discrete visual objects. The reason for this choice of underlying visualization is that the package grew out of our work with flip zooming [11, 12], which was developed specifically for this type of visualizations. However, the implementation of a general software package has allowed us to implement some quite novel variations of the original flip zoom concepts.

A Discrete Focus + Context Software Package

The package was constructed using the Java Abstract Window Toolkit [1]. It is based on two types of Java classes: *f+c (focus+context) components* and *f+c containers*, corresponding to **IV** and **IV'** respectively. An f+c component is based on a standard Java window component, with the added functionality needed to interface with a focus+context visualization. In terms of the formal description presented above, components must provide ways to facilitate event handling related to the interaction **I'** given by a higher-level visualization **IV'**. The **V** and **I** portions of the components provide the painting of the component on the screen, and the handling of input from keyboard and mouse, for instance to facilitate manipulation of the underlying data set **[D]**.

The f+c components are stored within f+c containers, in the same way as **[IV]** is used in **IV'**. An f+c container is a Java subclass of the f+c component class, meaning that it inherits the properties of the component and must facilitate the same functionality. An advantage of this is that it is possible to insert an f+c container into another f+c container, making higher-order visualizations possible. Further functionality is needed in order to support the focus+context visualization; most notably, the containers interaction portion **I'** has to allow for sequential transversal and the random access of focus objects.

The visualization **V'** consists of two parts: The *f+c layout manager* and the *f+c visualizer.* The layout manager, which handles how the components are placed on the screen area, can be implemented according to a number of different strategies, giving rise to a number of different presentation styles. It determines the size and position of the components and provides methods for how to change the layout when setting, changing and losing focus, or when objects in **[IV]** are inserted or removed during execution. The actual drawing of the components is done by the f+c visualizer, which has access to the different visualization functions **V** in the underlying visualizations in **[IV]**.

Examples of different implementations

We have used the software framework to implement a number of sample applications. In the following, we will briefly describe some of these, focusing on how **IV** and **IV'** are related to each other. (More details on the applications can be found in the references.)

The Hierarchical Image Browser

The *Hierarchical Image Browser* [13] was designed to explore the possibilities of using hierarchies to present large image sets in a structured way (see **Figure 1**). The hierarchies might for instance reflect the way art is exhibited in a museum, i.e. being placed in different rooms, sections and floors according to the types of paintings. The images in the set **[IV]** were ordered into containers **IV'** according to their placement in the hierarchy. Further, these containers were ordered in higher level containers **IV''**, **IV''',** etc., according to the hierarchical structure. This application shows how the general software framework allowed us to insert focus+context visualizations into higher-level focus+context visualizations, thus reflecting the general nature of the theoretical framework.

The Digital Variants Browser

Developed as an aid to literature researchers, the *Digital Variants* application [4] presented several versions of one text to facilitate comparative studies (see **Figure 2**). The application accommodated a number of document variants **IV,** each of which was presented in a focus+context display **IV'**. This set of focus+context visualizations **[IV']** was then visualized in a third-level focus+context visualization **IV''** of slightly different sort, namely one that allowed for two simultaneous foci, facilitating the comparison of two texts. This application shows how we could use the software framework to create second- and third-level focus+context visualizations with slightly different interactive and visual properties.

The WEST Browser

The WEST browser, a *WE*b browser for *S*mall *T*erminals [5], was developed for use on small mobile devices, such as Personal Digital Assistants (see **Figure 3**). Due to the limitations in display area (160 x 160 pixels) and computational power, both the space factor **s** and the computational factor **c**, put constraints on the visualization. To solve these problems, webpages were pre-processed in a number of steps to create a suitable structure **[IV]**. First, a web page was stripped of banners and divided into a number of small chunks, *cards*, each which would fit into the allowed screen space. The cards were then ordered in a hierarchical structure with no more than seven children to any node. All images in the original web page were scaled to the appropriate size and saved in the representation **[IV]**. Further, each of the cards was analyzed in order to find links and keywords. These were used as complementary structures of the webpage in **[IV]**. Thus, the pre-processing delivered three sets of **[IV]**: one based on the graphical look of the cards, one based on the extracted keywords and one based on the links.

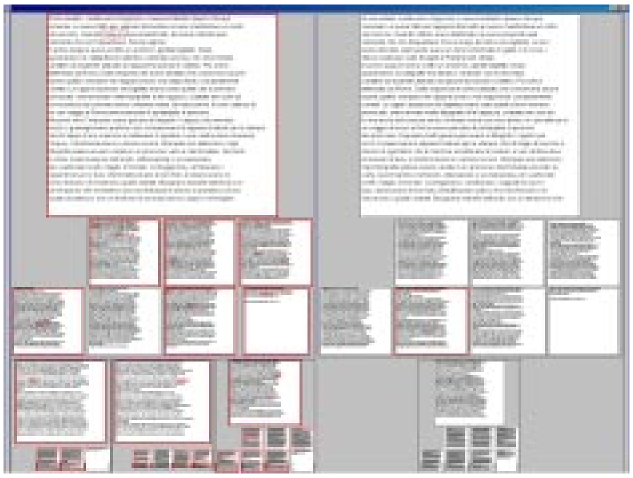

Figure 2. The Digital Variants Browser; a total of six documents are shown, two are in focus

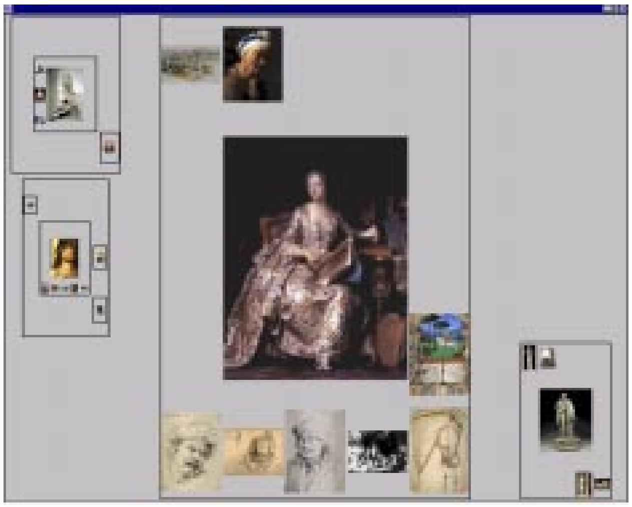

Figure 1. The Hierarchical Image Browser

The interface **I'** of the WEST browser facilitated navigation between the different levels of cards representing one webpage, but also the traditional functionality **I** associated with a web browser, such as the ability to follow links and use a history list. The user could also switch between three views: normal webpage, keyword view and link view, thus visualizing different components of **[IV]** in the same higher-order visualization **IV'**. This application shows how the

Figure 3. The WEST Browser allows for several different views of the same web-page source

framework allowed us to construct a complex interactive visualization of several different underlying visualizations.

DISCUSSION AND FUTURE WORK

In this paper, we first presented arguments for separating focus+context visualizations into first- and second-level visualizations, supported by some intuitive examples. We then presented a formal framework for describing properties of such aggregated visualizations and the relations between them. This enabled us to describe our initial examples in a formal way, thus validating the formal framework. We showed that the framework allowed us to construct some novel combinations of first- and second-level information visualizations. We also described some work with a general software package based on the formal framework, including example applications that uses hierarchies of focus+context visualizations and multiple underlying visualizations.

We can now see that according to our formal description, any **IV** that fulfils the constraints posed by **IV'** can be incorporated into **[IV]**. This means that we can incorporate any information visualization **IV** into any higher-level visualization **IV'**. This opens a lot of interesting possibilities: there is for instance nothing to stop us from applying several focus+context visualizations **IV, IV', IV'',** etc. to each other. As we saw with the hierarchical image browser and the Digital Variants browser, this can in fact be a very useful technique for combining different types of views or building a hierarchical visualization**.**

In the software package, we also have the possibility of using different types of applications within a f+c container as long as they fulfil the specified criteria for being a f+c component. One example of such an application is the *Focus+Context Desktop* (see **Figure 4**), which incorporates any application displayed in a Java window, including web browsers, web-cameras, file directory browsers and telnet clients, into a common workspace based on focus+context visualization (similar systems include [3, 15]). Future work should include evaluating such systems, as well as further experiments with nested focus+context visualizations, and applications that have heterogeneous types of underlying visualizations

The framework given in this article is not limited to focus+context visualizations, and it should be possible to use it to describe and construct many other types of interesting higher-level visualizations. Similarly, it should be possible to construct a software framework that supports other types of visualizations apart from focus+context techniques. (As we have seen, the Java language is quite suitable for the construction of such software.) However, we need to better understand the properties of the visualization components, (**V, V',** etc.) and the interaction components (**I, I',** etc). In particular, if we could isolate the necessary properties required for a certain higher-level visual component **V'** and interactive component **I'** to be compatible with the lower-order **V** and **I**, we will be able to state more clearly whether a certain combination of visualizations is likely to be practically useful or not. For instance, in the example section, we gave only an intuitive motivation for why Hyperbolic Trees might not be well suited to visualizing sequential data; if such relations could be expressed more formally, the usefulness of the framework should be increased quite significantly.

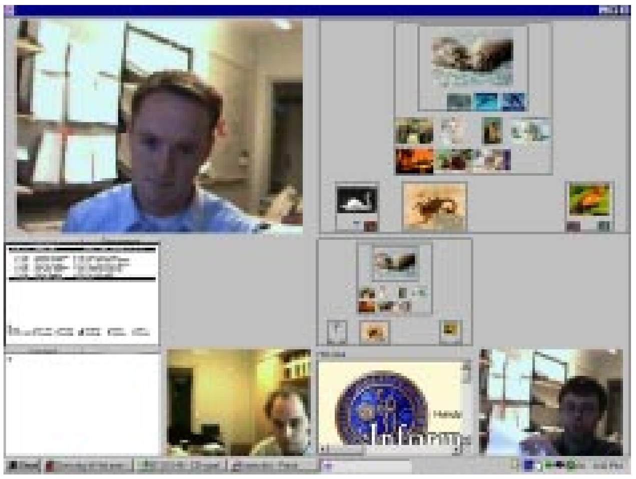

Figure 4. The Focus + Context Desktop, incorporating several different applications

If extended in such a way, the framework might allow us to better explore the properties of novel visualizations even before they are implemented. It might provide answers to questions such as: What focus+context visualizations are best suited to a specific underlying visualization? How can different visualizations be combined in a focus+context visualization? How does the interactivity of a underlying visualization affect a focus+context visualization and vice versa? Our hope is that by making the distinction between different levels of visualization explicit, and by introducing a formal system that supports this notion, new possibilities within the design space of both focus+context techniques and information visualization in general will become available.

ACKNOWLEDGEMENTS

The work presented here was part of the project *Effective Display Strategies for Small Screens*, funded by SITI, the Swedish Institute of Information Technology. We are grateful to our project partners at Ericsson Microwave, Telia Research and the Swedish Institute of Computing Science, and to our colleagues at the Viktoria Institute.

REFERENCES

1. Arnold, K. and Gosling, J. *The Java™ Programming Language,* Second Edition, Addison-Wesley, 1998.

2. Bartram, L., Ho, A., Dill, J., Henigman, F., The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Spaces, in

Proceedings of ACM UIST '95, pp. 207-215, ACM Press, 1995.

3. Bederson, B., Hollan, J., Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of ACM UIST '94,* ACM Press, 1994.

4. Björk, S. and Holmquist, L.E.: The Digital Variants Browser: An explorative tool for literature studies. *Proceedings of Computers, Literature and Philology,* Edinburgh, UK, 1998. (To appear)

5. Björk, S. and Redström, J. An Alternative to Scrollbars on Small Screens. *Extended Abstracts of CHI '99,* ACM Press, 1999. (To appear)

6. Card, S.K., Mackinlay, J.D and Shneiderman, B. Information Visualization. In Card, S.K., Mackinlay, J.D and Shneiderman, B. (eds.) *Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers, San Francisco, California, pp. 1-34, 1999.

7. Carpendale, M.S.T., Coperthwaite, D.J. and Fracchia, F.D. Extending Distortion Viewing from 2D to 3D. *IEEE Computer Graphics and Applications,* July August, 1997.

8. Furnas, G.W. The FISHEYE View: A New Look at Structured Files. Bellcore Technical Report, 1981. Reprinted in: Card, S.K., Mackinlay, J.D and Shneiderman, B. *Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers, Inc., San Francisco, California, 1999.

9. Furnas, G.W. Generalized Fisheye Views, *Proceedings of CHI '86,* pp. 16-23, ACM Press, 1986.

10. Furnas, G.W and Bederson, B.B. Space-Scale Diagrams: Understanding Multiscale Interfaces. *Proceedings of CHI '95,* ACM Press, 1995.

11. Holmquist, L.E. Focus+Context Visualization with Flip Zooming and the Zoom Browser. *Extended Abstracts of CHI '97,* ACM Press, 1997.

12. Holmquist, L.E. The Zoom Browser: Showing Simultaneous Detail and Overview in Large Documents. *Human IT,* vol. 2 no. 3, ITH, Borås, Sweden, 1998.

13. Holmquist, L.E. and Björk, S. A Hierarchical Focus + Context Method for Image Browsing. *SIGGRAPH '98 Sketches and Applications,* ACM Press, 1998.

14. Kadmon, N., and Shlomi, E. A polyfocal projection for statistical surfaces. *Cartograph,* J. 15, 1, 36-40, 1978.

15. Kandogan, E., Shneiderman, B., Elastic Windows: Evaluation of Multi-Window Operations. *Proceedings of CHI '97,* pp. 250-257, ACM Press, 1997.

16. Keahey, T. and Robertson, E.L. Non-Linear Magnification Fields. *Proceedings of IEEE Visualization '97, Information Visualization Symposium,* IEEE Press, 1997.

17. Keahey, T. The Generalized Detail-In-Context Problem. *Proceedings of IEEE Visualization '98, Information Visualization Symposium,* IEEE Press, 1998.

18. Lamping, J., Rao, R., Pirolli, P., A focus+context technique based on hyperbolic geometry for viewing large hierarchies. *Proceedings of CHI '95,* ACM Press, 1995.

19. Leung, Y.K, Apperley, M.D, A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction,* vol. 1 no 2, pp. 126-160, 1994.

20. Mackinlay, J. D., Robertson, G. G., Card, S. K, The Perspective Wall: Detail and Context Smoothly Integrated. *Proceedings of CHI '91,* pp. 173-179, ACM Press, 1991.

21. Plaisant, C., Milash, B., Rose, A. Widoff, S., and Shneiderman, B. Lifelines: Visualizing Personal Histories. *Proceedings of CHI '96,* ACM Press, 1996.

22. Rao, R. and Card, S.K. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. *Proceedings of CHI '94,* ACM Press, 1994.

23. Robertson, G.G., Mackinlay, J.D. and Card, S.K. Cone Trees: Animated 3D Visualizations of Hierarchical Information. *Proceedings of CHI '91,* ACM Press, 1991.

24. Robertson, G.G., Mackinlay, J.D., The Document Lens. *Proceedings of UIST '93,* pp. 101-108, ACM Press, 1993.

25. Sarkar, M. and Brown, M.H. Graphical Fisheye Views. *Communications of the ACM,* vol. 37 no. 12, pp. 73-84, 1994.

26. Sarkar M., Snibbe, S.S., Tversky, O.J. and Reiss, S.P., Stretching the Rubber Sheet: A Metaphor for Viewing Large Layouts on Small Screens. *Proceedings of ACM UIST '93,* pp. 81-91, ACM Press, 1993.

27. Shneiderman, B. Dynamic Queries for Visual Information Seeking. *IEEE Software,* 11(6), 70-77, 1994.

28. Spence, R. A taxonomy of graphical presentation *INTERACT '93 and CHI '93 conference companion,* pp. 113-114, ACM Press, 1993.

29. Spence, R., Apperley, M., Data base navigation: an office environment for the professional. *Behavior and Information Technology,* vol. 1 no. 1, pp. 43-54, 1982.

30. Tweedie, L. Characterizing Interactive Externalizations. *Proceedings of CHI '97,* ACM Press, 1997.