

Flip Zooming: Focus+Context Visualization of Linearly Ordered Discrete Visual Structures

Lars Erik Holmquist

Abstract. The focus+context visualization technique *flip zooming* was developed to present data sets that can be represented as collections of linearly ordered visual elements, such as the pages of a document or a collection of images. The technique works by laying out the elements 2-dimensionally in a left-to-right, top-to-bottom fashion that reflects the linear ordering of the elements. The user move an element to the focus by clicking on it, or by moving the focus forwards or backwards to an adjacent element in the sequence. The chosen element then zooms up to a readable size, while the other elements shrink accordingly. Since the linear ordering is preserved, users have access to both a detailed view of one element and an overview of the remaining elements presented in the correct sequence. Flip zooming has been implemented in a number of prototypes, including a text-only web browser, an image browser, and a browser for hierarchically ordered image collections. During the course of the implementations, user experience motivated a move from a space-preserving layout strategy (i.e. filling the display with as much information as possible) to a place-preserving one (i.e. trying to maintain the positions of visual elements as far as possible). Currently, the most promising application area for flip zooming is for use in devices with small displays, e.g. hand-held computers.

1 Introduction

Many of the information sets that we encounter in daily life consist of a number of discrete elements which can be viewed individually, but which make more sense when placed within the context of a certain linear ordering. For instance, the pages of a book or a document can be read individually; but most of the time, one wants to read them in the correct sequence. In a calendar, each day might be viewed individually, for instance to check the current day's appointments; but many times it makes more sense to see each day's entries in the context of the preced-

ing and following days. For a presentation, each slide accompanies a certain part of the speech; but in the flow of the presentation each slide will build on the previous and lead into the next. In each of these cases, we are not only interested in the switching from one item to another; we also want to know how far into a book we are and how much is left; if this week is more crowded with appointments than the next, and if there are any major holidays coming up that we should be aware of; if a boring speech will soon be over so that we can go home; and so on.

When we are using these objects, there are many physical clues that help us answer such questions. Books have a certain thickness, and by inspection we know approximately how far we have read and how much is left. Similarly, our paper calendars are easy to open at approximately the right place, and can be flipped through quickly to find a free spot for a meeting. And when listening to a boring presenter, we can see simply by the thickness of the stack of slides he or she is handling whether the pain will be over soon or whether it is time to think of a plausible excuse and make a quick exit. Electronic media rarely have these properties, and perhaps this is one of the reasons that many people prefer real books, paper calendars and transparencies over using computers. In particular, a computer screen is always of a limited size, which means that it can be very hard to get an overview of a material that is too large to be presented all at once on the screen.

However, some of the inherent limitations of computer screens might be overcome with novel display strategies. In our work, we have been exploring how to efficiently show large amounts of information on a limited display area, giving users visual access to both detail (“focus”) and overview (“context”). This is often referred to as *focus+context visualization*. The *flip zooming* focus+context visualization technique was initially developed with the intention of displaying documents, but has proven useful to display other data, such as image collections. In the following, we give an account of related visualization techniques, followed by a discussion of different types of visual representations. The flip zooming technique is then described, followed by an account of how it has been implemented in a number of prototypes. Finally, conclusions based on our experience with the flip zooming prototypes are drawn, and future work is outlined.

2 Focus+Context Visualization

The problem of displaying large amounts of information on a limited display can be approached in a variety of ways. For instance, interactive techniques such as *dynamic queries* can be used to enable users to interactively cut down the amount of information according to some desired criteria [24]. Various intelligent filters, so-called *software agents*, have been proposed to automatically reduce the amount of information that reaches the user even before it is visualized [19], and so on.

However, if we do not want to cut down or filter out any information, we are faced with the problem of how to efficiently show a very large data set on a limited display area. Consider a large map, perhaps several meters across. If we shrink it to a size small enough to show it on a desktop screen, the user will be able to see the whole map at once, but the map will probably be too small for her to make out any detail. If we let the user see a portion of the map in actual size through a scrolling window, she can scroll to any section she wants and see that in sufficient detail, but will then have lost the important overview. A better solution might be to first present the user with an overview, and then let her zoom in on a desired portion, as for instance in the *Pad* and *Pad++* interfaces [1, 20]. However, when she views the entire map she still has no access to details, and when she zooms in to reveal details the overview will still be lost!

Therefore, it would be useful if we could present *both* an overview and a detailed view of a large material. One class of solutions to this problem are termed *overview+detail*. Here, the overview and the detailed view are not presented on the same area; instead, the user can either switch between them on the same display, much like zooming, or see them presented in different parts of the screen [7, p. 285]. In contrast, *focus+context* techniques aim to *integrate* the overview and detail in the same display area [7, pp. 307-309]. By not forcing the user to divide her attention between several different display areas, such techniques aim to provide a more effective access to visual information in a large data set. In the following, we give an brief outline of the development of focus+context visualization techniques; for a more complete view, Card et al. [7] provides a good starting point.

The first examples of focus+context visualization were non-interactive techniques for visualization of map data [14]. With the introduction of computers, it became possible to perform focus+context visualization

interactively, and an early computer-based method was the *FISHEYE View* [9], later presented in slightly different form as the *Generalized Fisheye View* [10]. Here, a framework for information filtering based on the users' current point of interest was presented, so that the user for instance could see the most important information (that which was "close to the focus") in great detail, while less important information ("farther away from the focus", according to the metaphor) was only presented in outline form. The Generalized Fisheye View was initially implemented to work on text-based displays; a contemporary visualization for graphical displays was the *Bi-Focal Display* [26]. Here, distortion in the horizontal dimension was introduced to give users access to both overview and detail in a wall-sized calendar display. It is worth pointing out that the Bi-Focal display was presented as part of a larger system, where wall-sized displays, physical icons, novel input devices, etc. aided users' navigation of a large information space.

Later, the *Perspective Wall* [18] used a 3D perspective to present temporal data, achieving a similar effect to the Bi-Focal Display, and the *Document Lens* [21] 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 data displays include the *Graphical Fisheye View* [22], which allowed users to zoom in on a node in a graph or network without disturbing the spatial relationships between nodes; and the *Rubbersheet View* [23], which enabled users to apply a 2-dimensional distortion to an image, analogous to the effect of stretching a rubber sheet mounted in a rigid frame. Such techniques have also been extended to 3-dimensional displays [8]. Techniques developed specifically for visualizing hierarchies includes the *Hyperbolic Tree Browser* [16], which mapped information organized in a tree structure onto a hyperbolic surface.

3 Visual Structures in Focus+Context Visualization

When constructing a focus+context visualization for a certain data set, it is of great importance to pay attention to the way that a data set is *represented visually*. The reason is that a focus+context visualization is a form of *view transformation*, i.e. it takes a visual structure and performs

a number of operations to transform the view interactively according to the user's actions [7, pp. 17, 31-32]. Another way of describing this is to say that a focus+context visualization can be seen to constitute a *second-level visualization*, or a “visualization of a visualization”; a formal account of this is given in [5]. In practice, this means that when a focus+context technique is applied to a data set, an underlying visual structure has already been selected for that data set, and that visual representation will influence whether the focus+context visualization is successful or not.

3.1 Continuous and discrete visual structures

Many focus+context techniques have been developed to visualize a 2-dimensional continuous visual presentation of some data. These visualizations simply treat the whole visual representation as a 2-dimensional image, and apply distortion to portions of the image regardless of what the image actually represents. We can thus apply techniques such as the Document Lens [21] or the Rubbersheet View [23] to *any* 2-dimensional picture, regardless of whether that picture represents a set of document pages, a map, a graph or something else entirely.

However, some focus+context techniques, for instance the Hyperbolic Tree Browser [16], can not readily be applied to continuous presentations; these techniques are specifically developed to visualize structures consisting of discrete visual elements of some sort. The difference is that these techniques transform each element in a visual structure separately and then adjust the overall presentation to accommodate these transformations, while retaining the important structural information. For instance, when zooming in on a node in a graph using the Graphical Fisheye View [22], there is no need to distort the individual nodes – these can retain their proportions. However, the placement and size of the nodes is adjusted as necessary to accommodate the increased size of the element in focus while still correctly reflecting the structure of the graph. Compare this to the same operation in the Rubbersheet View [23] – here, the surrounding information is distorted vertically and/or horizontally, since the transformation is continuous.

Thus, we can see two major types of representations where focus+context visualizations (and other view transformations for that matter) can be applied:

- *Continuous visual structures*, such as maps and images
- *Discrete visual structures*, such as trees and graphs

Continuous representations have been very popular in focus+context visualization. Leung and Apperley [17] gave an overview of such techniques, proposing a generalized “rubber-sheet” metaphor for describing them. Another general description of how to apply a focus+context view to a continuous 2-dimensional image can be found in the work on *Generalized Detail-in-Context Views* [15]. Here it is shown how a variety of continuous transformation can be applied to a 2-dimensional surface, replicating the effects of other continuous focus+context techniques. This method can also be shown to replicate “1-dimensional” techniques such as the Bi-Focal Display and the Perspective Wall. Thus, it seems that all the multitude of continuous focus+context techniques can be represented as a continuous function applied to a 2-dimensional surface.

Among discrete focus+context techniques, there are many notable visualizations of graphs (e.g. The Graphical Fisheye View applied to graphs) and hierarchies (e.g. The Hyperbolic Tree Browser). Although graph visualizations usually aim to preserve certain 2-dimensional relations, they can take greater liberties in moving the discrete elements, than can the continuous techniques. In visualizing trees, no relations need to be preserved except those which indicate the relations between parent and child nodes, thus allowing for even greater flexibility when designing the resulting focus+context layout. The strength of the Hyperbolic Tree Browser, for instance, is that when it maps a tree structure onto a hyperbolic surface the results may not look like any traditional tree, yet the tree structure is still immediately and intuitively recognizable. Mapping a graph or a continuous 2-dimensional image onto a hyperbolic surface in the same way would probably give a much less satisfying result. On the other hand, the Graphical Fisheye View visually preserves the relationships in a graph to much greater degree, but cannot present a tree nearly as effectively as the Hyperbolic Tree Browser.

3.2 Structuring of visual elements

Most of the time the information in a discrete visual representation is not simply a collection of elements (i.e. a set), but it is *structured* in

some way, which in turn will influence the structure of the visual representation. The way in which a visual representation is structured will effect a focus+context visualization in how it restricts the *traversal* of the elements, i.e. how a user can choose a new focus by moving the focus from the current element to one that is adjacent to it in the visual structure. For instance, when viewing a linear sequence of elements, say like the pages of a document, the user should be able to move back and forth in the document to preceding and following pages. On the other hand, when viewing a hierarchical structure, i.e. a tree, the user should be able to traverse the tree according to its structure of branches and leaves, and so on. In other words, the user should always be able to traverse the data set in a way which reflects its inherent structure.

It thus seems that we need to make a difference not only between discrete and continuous representations, but also in the various ways in which discrete representations are structured. Shneiderman [25] listed seven types of *data* structures: 1-dimensional; 2-dimensional; 3-dimensional; Temporal; Multi-dimensional; Tree; and Network. However, a data structure is not the same as a visual structure – in many cases there has to be a mapping from the data structure to a corresponding visual structure that can be presented on a 2-dimensional display. Apart from unstructured collections, i.e. sets, we can see three such visual structures:

- *Linear ordering*, where each element has a place in a linear (1-dimensional) sequence. Here, the user can only go to the following or the preceding element in the sequence.
- *Hierarchical ordering*, where each element has a certain place in a tree structure. This can be traversed by going up and down in the hierarchical structure.
- *Graphs*, where each element can be connected to one or more other elements to an arbitrary level of complexity. Here, the user has the most freedom since she can go to any node in any direction that there is a direct connection from the current node

Although the linearly ordered structures offer the most limited navigation, they are quite important, since they can be used to represent a wide variety of data types, and we will concentrate on this aspect for the remainder of the paper.

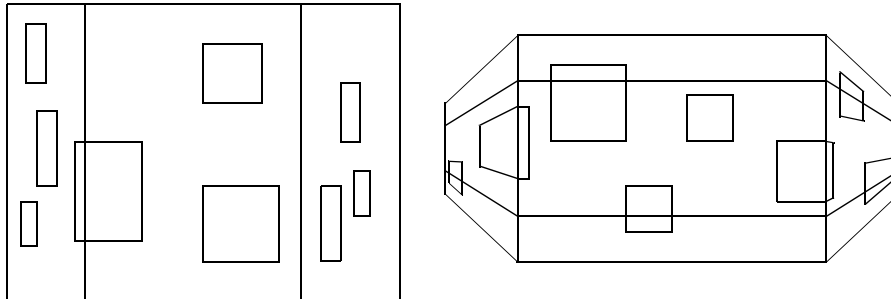


Fig. 1. Schematic diagrams of the view transformations taking place in The Bi-Focal Display (left) and The Perspective Wall (right).

3.3 Linearly ordered discrete visual structures

We can now see that there is something in common between the physical information representations that were discussed in the introduction – documents, calendars, presentation slides, etc. They consist of a number of separate visual elements, each which can be viewed on its own; but they also possess an inherent linear ordering, which must be preserved if the data set is to make sense as a whole. In other words, they are all *linearly ordered discrete visual structures*. This type of visual structure is quite common in the real world, but difficult to effectively represent on a computer screen. The most common way to do this is to use a scrollbar, which lets the user move a small window over an area that is larger than the display. However, traditional scrollbars have no means of providing an overview of the material, apart from telling the user approximately at which position the window is relative to the larger area.

Some focus+context methods have been developed for displaying an area that is wider than the available screen space; two of these are shown schematically in **Fig. 1**. The Bi-Focal display (left) [26] solved this by compressing the horizontal portion of the material left and right of the center, whereas the Perspective Wall (right) [18] used graphical distortion according to a 3-dimensional metaphor, by showing the context material as if it was receding into the background of the image on both sides of the focus. Although these methods are designed for continuous visual representations, they might also be modified to display dis-

crete visual structures. For instance, if displaying a set of document pages, we might allow only one page in focus at the time, taking up the full focus area, and have all the preceding and following pages displayed in the left and right context areas. Navigation could then be performed in discrete steps, by moving one page forwards or backwards.

But it should also be noted that although the methods mentioned above would preserve the linear ordering of a discrete visual structure well, they are not optimal in all respects. The Bi-Focal Display introduces a heavy horizontal compression on the context areas, while keeping the vertical dimensions unchanged; this “stretching”, while space-filling, gives context elements a very distorted look. The 3D-metaphor used in the Perspective Wall gives a more natural appearance to the context, since it resembles how far-away objects would look in the real world, but it is not very space efficient since a large portion of the vertical space goes unused. Also, although fitting to the perspective metaphor, the distortion introduced in the context display may make elements far from the focus hard to recognize.

4 The Flip Zooming Focus+Context Technique

With flip zooming, we initially set out to design a focus+context visualization technique for documents, in particular documents consisting of several discrete pages such as the pages of a book. However, having arrived at such a technique, we of course hoped to be able to generalize it to other types of data. We wanted our new method to share some fundamental properties with previous focus+context techniques, including:

- *Overview of the whole data set.* The entire data set should be presented simultaneously, to give the user an overview.
- *One selectable focus.* It should be possible to focus on one element, i.e. present it large enough to be readable.
- *Random access to any visual element.* In the visually presented data, the user should have instant access to the whole data set, so that any element can be moved to the focus (e.g. by point-and-click)
- *Space efficiency.* The available space should be used to display as much information as possible.
- *Preservation of linear ordering.* The inherent linear ordering of the elements should be preserved and presented in the resulting display.

Some additional goals that we wanted to achieve, that were not fully met by previous methods, were:

- *Low computational demands.* To facilitate real-time interaction, the methods should require as little calculations as possible.
- *No distortion of the text.* All continuous focus+context methods introduce some degree of distortion, which we found undesirable, especially for viewing text. We felt that a method which preserved the proportions of the visual elements would be preferable.
- *Linear traversal of visual elements.* Since the most common way to traverse a document (or, more generally, to a set of linearly ordered visual elements) is to go from one page to the next, the new method should support this by allowing the user to move “backwards” and “forwards” through the elements (e.g. by using some keyboard command)

4.1 Design process

With the above goals in mind, we started sketching a variety of different possibilities. The basic idea was to lay out a document as separate pages and then let the user zoom in on a particular page. Some obvious ways

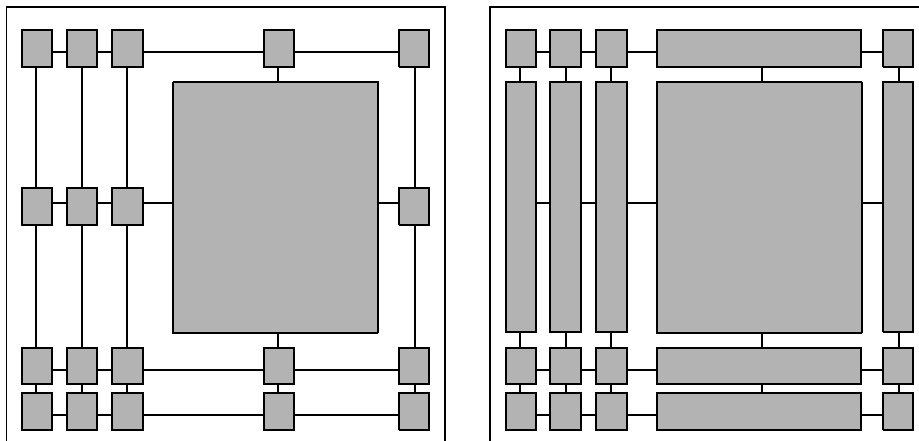


Fig. 2. Some concepts sketches; the gray boxes represent document pages. Left, proportions are preserved; right, space is used more efficiently by distorting the context elements.

of achieving this are sketched in **Fig. 2**. In the left figure, which is similar to the Graphical Fisheye View [22], when a user zooms in on a page, the other pages shrink accordingly to accommodate the current focus; their proportions are preserved. This has the advantage of not distorting the content of the pages, thus making them easier to recognize, but it does not maximize the use of screen space, since there are large areas which are not used to display information. In the right variant, which is similar to the Rubbersheet View [23], certain pages are expanded vertically or horizontally to the same degree as the focus. This has the advantage of using the available space more efficiently to display information, but it also introduces visual distortion to the content of the pages.

It seemed that there was no way to achieve a display which preserved the linear ordering while using the space efficiently and without introducing distortion to the individual elements. However, after more sketching, we realized that we were placing a too hard restriction on our display: To preserve linear, i.e. *1-dimensional* ordering, we were in fact preserving a *2-dimensional* ordering. We realized that if we found some other way to preserve the linear ordering, we might break the rigid frame that had constricted the previous variants, thus utilizing the full display area more effectively. The resulting technique, which allowed users to “flip through” a data set, and “zoom in” on the element which most interested them without losing the overview, was termed *flip zooming*.

4.2 Flip Zooming

The easiest way to explain how flip zooming works is to refer to **Fig. 3**. Here, it is schematically shown how a set of linearly ordered discrete visual elements are first laid out on a 2-dimensional surface (left). This could for instance be thumbnails (i.e. miniature graphical representations) of the pages in a document. The figure then shows how the same data collection looks when the user has chosen to focus on a certain element; the element is zoomed up to a readable size, while the surrounding elements are shrunk accordingly to accommodate the focus (right). Compare the view in this figure with the corresponding views in **Fig. 2**. It can be noted that flip zooming allows for the context elements to be of a much larger size (approximately four times the size in these examples), thus preserving more readability, while still not introducing any distortion to the individual elements.

However, the linear ordering of the document pages or other data must be preserved in some way. In focus+context methods which completely preserve 2-dimensional relationships, this is done automatically. In flip zooming, it is instead done through the convention of always laying out the elements in a *left-to-right, top-to-bottom* fashion. **Fig. 4** illustrates how this works. When the view is not zoomed, the ordering of the elements is quite straight-forward. When the user zooms in, the view changes, but the ordering is preserved according to the left-to-right, top-

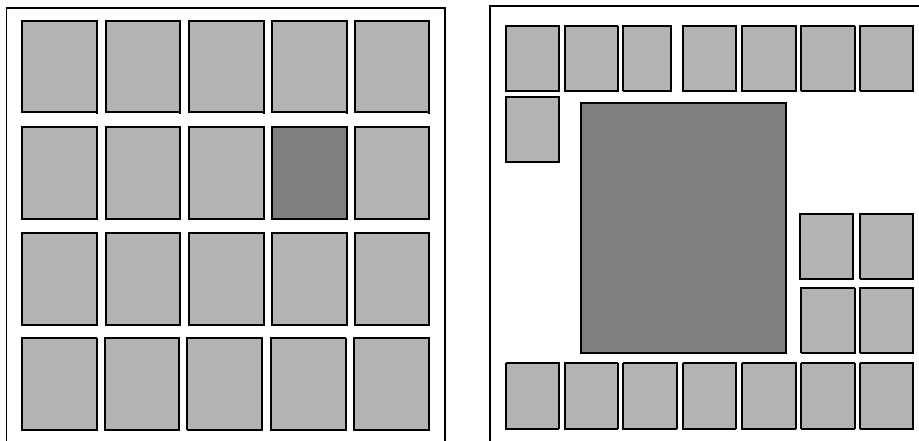


Fig. 3. Flip zooming view of 20 linearly ordered elements. Left, un-zoomed view; right, zoomed view.

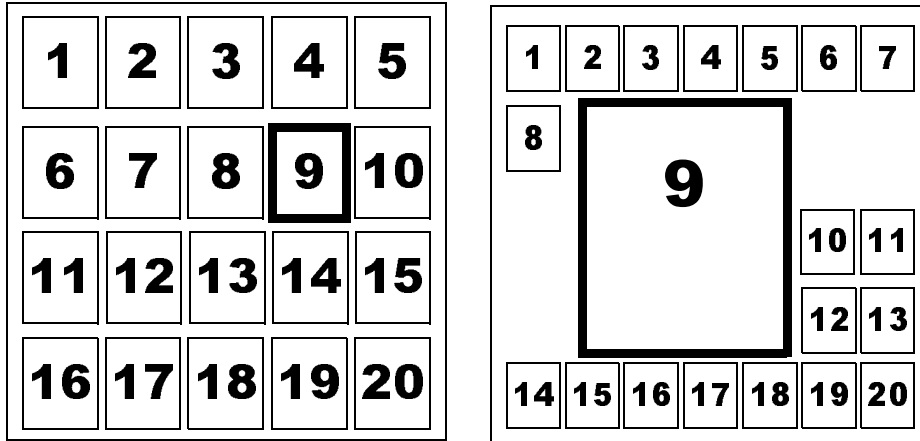


Fig. 4. A view of the left-to-right, top-to-bottom ordering.

to-bottom scheme. Every element which is *before* the focus in sequence, can be found *to the left of and / or above* the focus; every element that comes *after* the focus can be found *to the right of and / or below* the focus.

It should be noted that the exact results of this layout scheme is not strictly defined. In particular, there is quite a lot of freedom in deciding exactly how to construct a zoomed layout, both when it comes to the placement of the focus and the layout of the context elements. This means that it might be hard to tell where an element might appear when focused on, since this might not be the same as its position when not in focus. **Fig. 3** provides a good illustration of this, since the zoomed element in the right figure does not seem to have appeared very close to its previous position in the left figure (compare this to **Fig. 2**, where the element is more or less fixed in the same place when zooming).

We can now see how flip zooming fulfills the criteria we initially set up:

- *Overview, focus, and random access.* All these fundamental focus+context properties are fulfilled by flip zooming; the whole data set is presented visually, and any element can be moved into focus (i.e. zoomed up to a readable size) by clicking on it.
- *Space efficiency.* Because of the mapping to two dimensions, flip zooming takes advantage of the shape of most displays, by efficiently using the vertical as well as the horizontal screen space for context

information. Also, compared to methods which use two dimensions but preserve a strict 2-dimensional ordering, flip zooming uses space more efficiently with the result that the context elements can be made larger; compare **Fig. 2** (left image) with **Fig. 3** for an illustration of this.

- *Preservation of linear ordering.* With the consistent left-to-right, top-to-bottom ordering, flip zooming preserves the linear ordering of the elements, although it maps it to two dimensions and modifies it as necessary to make the most effective use of the display.
- *Low computational demands.* Flip zooming requires very little computational power. The visual elements will need to be scaled to the right size, but once this has been done, the only computation that needs to take place is that required for calculating the placement of the visual elements. Since the visual elements can be cached once they have been calculated, flip zooming systems can be very responsive even on machines with very limited processing power.
- *No distortion of individual elements.* Flip zooming does not impose any visual distortion on the context material. Instead, all elements keep their proportions and are merely reduced in size, which means that their visual properties are better preserved. Flip zooming does not introduce vertical or horizontal distortion to context elements; compare **Fig. 2** (left image) with **Fig. 3** for an illustration of this.
- *Linear traversal of visual elements.* A user can always move to the next or previous element in the sequence on the display. (Exactly how this is done is implementation dependent but it could be done for instance by using the “left” and “right” arrow keys, or by providing some GUI buttons.)

However, there are also some obvious disadvantages with the method:

- *The placement of focus and context elements is not fixed.* Whenever arranging a flip zooming layout with one element in focus, there may be several (equally correct) ways in which the elements can be placed. This might confuse users since it can be hard to predict the result of a certain action, e.g. where a new focus will appear when focusing on a certain item.
- *Zooming changes the row and column layout.* When the user zooms in on an element, it is necessary to change the number of rows and columns to accommodate the focus; this may make the transition between different views confusing.

- *Limit to how much data can be displayed.* At a certain point, when there are too many elements visible, the context elements will be too small to be useful (the theoretical limit is context elements that are just one pixel in size, but in practice this happens much earlier). To combat this, some strategy for displaying more complex structures, e.g. hierarchies, might be needed.

5 Flip zooming prototypes

To experiment with and evaluate the technique, we have implemented flip zooming in several applications. In the following we will describe three prototypes:

- *The Zoom Browser*, a text-only web browser
- *The Flip Zooming Image Browser*, an application for browsing images
- *The Hierarchical Image Browser*, an application that allowed browsing of more complex image collections

When describing the prototypes, we will take special note of these points:

- *Layout*, i.e. the various ways in which the prototypes arrange the individual visual elements.
- *Lessons from user experience*, i.e. what we learned from evaluating the prototypes.

5.1 The Zoom Browser

The first implementation of flip zooming was the *Zoom Browser* [11, 12]. It was a text-only browser for the World Wide Web and acted as a test-bed for exploring the feasibility of flip zooming. The Zoom Browser presented a view of one or more web pages by dividing the text into a number of separate chunks, and then presenting the chunks as pages of text on a flip zooming display. The Zoom Browser was developed using an early version of the language Java. This meant working under some limitations, most notably that the Java language at that time was quite slow; for this reason, it was fortunate that flip zooming has

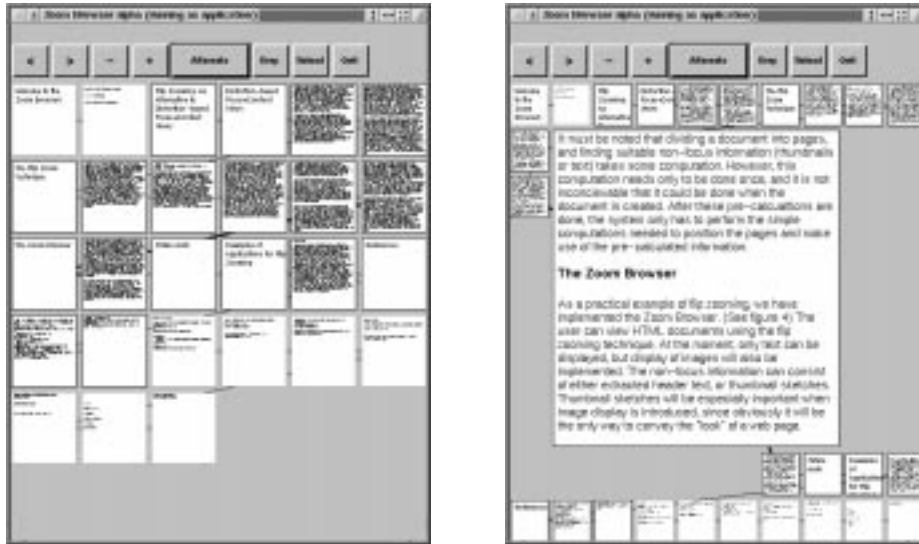


Fig. 5. The Zoom Browser; starting view (left) and zoomed view (right).

very low computational demands (as pointed out above). Also, the lack of any readily-available HTML parsing and rendering code meant that we had to limit the display of web pages to a text-only format. These disadvantages were however more than compensated for by the advantages of using Java, including easy implementation of web services and the possibility of making the prototype instantly available as a demo on the web.

Layout. The layout used in the Zoom Browser was designed to be *space-preserving* rather than *place-preserving*. To maximize the usage of screen space and leave as little area of the screen unused as possible, it was necessary to allow focus and context elements appear at different positions depending on what element was in focus. Furthermore, the layout allowed for some freedom in the placement of the focus element. This was used to place the current focus element in a position that could be assumed to be close to the user's current focus of attention. Depending on whether the user had clicked on a context element, or used a command shortcut to advance to the next element, the focus element was placed as close as possible to the selected context element, or the previous placement of the focus element, respectively. However, this

also meant that the same focus element could appear at different positions depending on which action had been performed to arrive at the focus.

Lessons from user experience. The Zoom Browser was not subjected to any formal user testing. However, we let several users experiment with the prototype and comment on its features, and received some valuable feedback, including:

- *The layout was sometimes confusing.* Our decision to make the layout scheme space-preserving rather than place-preserving unfortunately resulted in situations where it was hard for users to predict where a certain element would appear during interaction. On the other hand users also complained that the browser did not seem to be using the display space as effectively as possible (even though it can be shown that it actually did use the space as effectively as possible most of the time). This was an indication that there is an important balance to be struck between making a display that is too “crowded” and confusing, and between making a clear and understandable display, which might not use the available screen-space as effectively as possible.
- *Users liked the quick access to detail and overview that the Zoom Browser provided.* Several users immediately expressed sentiments like “I want this!” or “When can I have this for my word processor?”. These user reactions showed that for viewing text, flip zooming seemed to be a very compelling technique.

5.2 The Flip Zooming Image Browser

After the initial experiences with the Zoom Browser, we wanted to explore how we might use the flip zooming technique on other kinds of data sets. Browsing image collections seemed a natural progression from browsing text pages, since they too are comprised of individual visual elements. Many image collections do not have a strong inherent linear ordering, but can be accessed in any order, something that might make them a less obvious subject for flip zooming. On the other hand, the overview and easy access provided by flip zooming might make image browsing a suitable choice. Thus we decided to implement a *Flip Zooming Image Browser*.

We implemented the Flip Zooming Image Browser in Java, like previously the Zoom Browser. At this time, the Java language was still too limited to smoothly scale images in real-time; instead we decided to use a set of pre-scaled thumbnails of a fixed size that were stored along with the full-size images. This limited the flexibility of the application somewhat in that it was difficult to adapt the layout to different screen and window sizes. It was however evident that the advantages of using Java, such as the short development time and the possibility of making material available directly on the web, again outweighed the shortcomings. Fortunately, the continued development of the Java language meant that the problems could be overcome in the next prototype (below), making this original Image Browser a stepping stone that was most valuable in the user experience it provided.

Layout. The Image Browser, like the Zoom Browser, used a space-preserving rather than place-preserving layout strategy. However, it was slightly less effective than the Zoom Browser, since it could not scale images in real-time to make the most of the available display area.

Lessons from user experience. We performed a formative evaluation of the Flip Zooming Image Browser [3]. The study emphasized qualitative results and was an effort to find possible areas of improvement of the prototype as well as of flip zooming in general. In the study, ten users performed a variety of tasks while using the Flip Zooming Image



Fig. 6. The Image Browse running as a web-page applet; browsing images (left) and PowerPoint slides (right).

Browser to browse two collections of images: one consisting of pictures of different animals, and one consisting of a collection of PowerPoint slides with a mixture of text and graphics. The tasks were designed so that they required the use of different search strategies, such as image recall, text search, etc. The tasks were for instance: “find a slide with a certain word”, “find out if there are more pictures of one type of animal than another”, etc.

To provide a contrasting experience, the subjects also performed similar tasks using the Adobe Acrobat PDF Reader’s overview+detail display to browse PDF versions of the images, and a traditional web browser to browse web pages generated from the presentation. However, the applications were too different for them to be directly comparable (for instance, the Acrobat Writer did not provide thumbnails, only blank rectangles, making for a severely limited overview). After each session, users were subjected to semi-structured interviews, the results of which were collected in various categories (e.g. “Good overview”, etc.). Some points which were mentioned by at least 50% of the subjects were:

- *Overview.* 10 (i.e. all) users said that the Flip Zooming Image Browser provided a good overview of the material; in comparison, 3 said that the web page provided a good overview and only 1 that the Acrobat Reader provided a good overview. Thus, we could conclude that users perceived the flip zooming as a useful approach to giving an overview of a large material.
- *Clear or confusing display.* 5 users specifically pointed out that they liked the clear layout of the Adobe Acrobat Reader (one user called it “neat and tidy”). In contrast, no users said that the Image Browser had a clear display, but 5 users specifically pointed out that they felt the Image Browser’s display was confusing. This was likely a reflection of the trade-off between space- vs. place-preserving layout strategies, and we concluded that even though we managed to use the space effectively this was not particularly appreciated by users (no user mentioned it). Instead, we felt it might be fruitful to explore methods to make the display less confusing by keeping elements fixed in the same place, even if that meant a trade-off in screen-space usage.
- *No / too small thumbnails.* 5 users thought that the thumbnails displayed by the Image Browser were too small (despite our space-preserving strategy). This might be because the size of the thumbnails

was fixed, due to the technical limitations mentioned above. 9 users complained that the Acrobat Reader showed no thumbnails, only outlines, indicating (not surprisingly) that thumbnails are considered useful when browsing image collections, and should be included if possible.

Suggestions from users included to allow for a full zoom-in on any picture on the display, and to separate the thumbnails from the focus image in a manner similar to the Acrobat Reader's display. We also noticed that the overview provided by flip zooming allowed users to initially give approximate (and fairly, but not 100%, correct) answers to some tasks, which they then would rapidly refine through a more extensive search of the material. Thus we saw that flip zooming could be used by novice users to first get an overview or a general "feel" for a visual material, and then allowed for a more detailed exploration.

5.3 The Hierarchical Image Browser

Based on the experience gained from the Image Browser, we constructed a new prototype: The Hierarchical Image Browser [2, 13]. Thanks to the improvement of the Java language, it was now possible to scale images in real-time, allowing for a much more flexible display. Furthermore, we wanted to explore how we might visualize more complex collections of images, in particular those which were ordered according to certain categories; thus the introduction of hierarchies. By using the fact that Java allows for nesting of interface components, hierarchical displays was achieved quite easily. The flip zooming application was generalized to allow the display of not just images, but any kind of interface components; and since the flip zooming display in itself is an interface component, nesting followed automatically. The display displays another flip zooming display as if it were just another visual element, thus allowing for introduction of any number of nested flip zooming views.

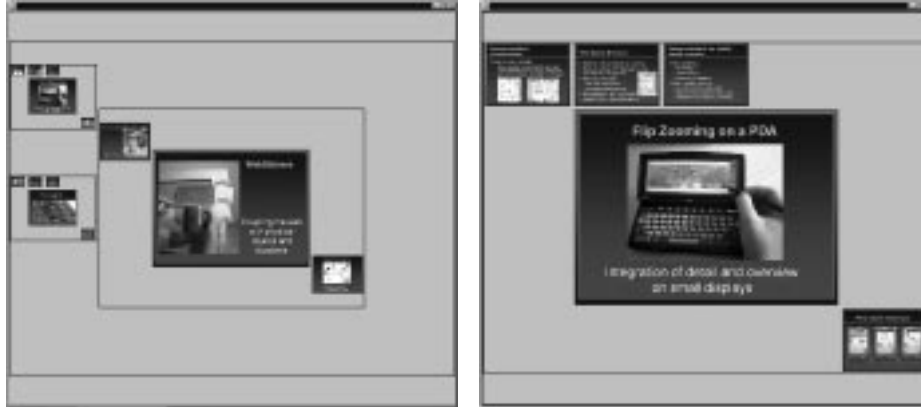


Fig. 7. The Hierarchical Image Browser; showing a full overview of a structured PowerPoint presentation (left) and zoomed in on one section (right). A full-screen view, where a single slide fills the whole display, is also available but not shown here.

In practice, the user would provide the application with a file structure of one or more directories containing images, where each directory in turn can contain sub-directories and so on. The application creates a main flip zooming display for the whole directory tree, then creates a new flip zooming display inside the main display for each individual directory, and recursively creates new nested displays for each subdirectory, and so on. Thus the entire directory tree is represented as flip zooming components, which in turn may contain images and/or further sub-components.

Layout. The layout of the Hierarchical Image browser was heavily influenced by the formative evaluation of the first Flip Zooming Image Browser. In this evaluation we found that users were confused by the fact that the focus image could appear on different places in a not very predictable manner. This was because of our effort to create a space-efficient display.

For this prototype, we decided to instead implement a place-preserving layout, and keep the focus element in a fixed location at the middle of the display. This would not place the image at the same location as the corresponding thumbnail when it was clicked on (as in the Zoom Browser), but it would make it appear consistently on the same place

every time, making for a more predictable interaction. These changes, however, meant that we had to make the layout much less space effective: to maintain the linear ordering, we had to leave some portions of the display area unused, depending on which image was in focus. But based on the evaluation of the previous prototype, we felt that this was a necessary step to take.

Additionally, the Hierarchical Image Browser allowed the user to control the amount of context displayed, by enabling the user to zoom in completely on one branch of the directory tree if needed. When doing so, only the images in the current directory were shown, not the outside context. Thus, users could perform a form of visual “pruning” of the hierarchy, so that context that was not relevant to the current task was not displayed.

Lessons from user experience. To evaluate the Hierarchical Image Browser, we decided to perform an “evaluation by real usage”, choosing some real expert users as subjects – namely, ourselves. We therefore made sure that the prototype was sufficiently robust to use it as a presentation tool, acting as a direct replacement to Microsoft PowerPoint. Presentation slides still had to be prepared in PowerPoint, but by saving them as JPG images, they could be viewed directly in our own prototype.

We used the Hierarchical Image Browser for about six months in a variety of situations, including formal talks at conferences (e.g. at SIGGRAPH '98) and at open walk-in demonstrations (e.g. at CSCW '98). At the demonstration, we ran a flip zooming display of a presentation on a large touchscreen in conjunction to the demonstration, to be able to quickly provide illustrations to certain points in the demonstration. We also used the prototype at external presentations of our work at other research labs, when giving presentations for local industrial partners, during lectures at the local university, and so on.

During this usage, we found the Hierarchical Image Browser to be a good help in presenting structured presentations, especially when several different topics were to be covered using different sets of slides. These slides were then simply collected in different directories, and presented one by one by zooming in on each slide set. Especially useful was the ability to be able to quickly jump to some slide (for instance when an audience member asked a question) by zooming out to get an

overview of the presentations, find the slide by visual inspection of the whole set, and then zoom in again to display the slide in full size.

6 Discussion

With flip zooming, we have introduced a focus+context visualization technique which preserves the linear ordering of a set of discrete visual elements, while making efficient use of a 2-dimensional display area. This is done by mapping the 1-dimensional ordering to two dimensions, thus using both vertical and horizontal space efficiently. However, this approach gives rise to some problems. Most previous focus+context techniques, in particular continuous methods but also those that deal with discrete visual representation, are strictly *place-preserving*, i.e. they keep the spatial relationships between visual elements intact when the user zooms in. Flip zooming does not require this; when the user zooms in on an element, it may change its position to a completely different part of the display, the context elements may also move, and the number of rows and columns may change. All this can of course serve to confuse users when interacting with a flip zooming system.

The reason for the changing display, however, was to make flip zooming *space-preserving*, i.e. to use the available display space as efficiently as possible. Many focus+context techniques do this, usually by the introduction of spatial distortion (vertical, horizontal, or resembling a 3D-dimensional effect). Flip zooming does not introduce spatial distortion of the individual elements, but this is done at a price: to keep the technique as space-preserving as possible, we needed to loosen the place-preserving restrictions, with the problems outlined above as a result.

That the unpredictable layout indeed was a serious problem was clearly born out by the user experience, and in the last prototype, we attempted to address this. We decided that it was worth trading some of the available display area for a more easily-understood layout. To accomplish this, we changed the layout strategy from being primarily space-preserving in the first two prototypes to a more place-preserving approach in the last. In particular, this meant keeping the position of the focus element at the centre of the display, rather than placing it at the position that would permit us to use the available space most efficiently.

The result was a layout that gave a predictable response during user interaction, albeit at the cost of being less information dense.

We believe that this result, where we saw the necessity to trade a certain amount of information density for a clear and predictable display, has implications for focus+context visualization in general. It is not enough to provide a totally space-efficient display if users are confused during interaction. At the same time, it is not always necessary to maintain a totally strict space-preserving layout (or, as in for instance the case of document visualization, to strictly preserve a 2-dimensional layout for a linearly ordered data set). Even when accounting for the introduction of a more place-preserving layout the flip zooming technique is still quite space efficient when compared to many other techniques, and through the course of the prototype development it has achieved a promising middle-ground between space-efficiency and user friendliness

7 Future work

With flip zooming, we have so far only started to examine the implications of the trade-offs of space-preserving vs. place-preserving layouts. We see an opportunity for much user-testing of both current and forthcoming prototypes, hopefully to be able to construct guidelines for when and to how high a degree such a trade-off should be considered. Also, real benchmarking of flip zooming versus other focus+context techniques is yet to be done, and there is a need for both quantitative and qualitative comparative studies in this area. In addition to this, we think that flip zooming can be generalized to other usage situations and data types than those described in this paper.

Currently, one of the most promising areas for flip zooming small displays, such as those found on hand-held computers. The fact that flip zooming has low computational demands makes it extra suitable for such applications. We have so far developed two prototypes that used flip zooming to display data on small displays: *WEST* [6], a web browser for small terminals, and *PowerView* [4], an integrated activity calendar which allowed users to retrieve personal data commonly stored on a PDA, such as contacts, meetings, to-do-items, e-mails, etc. In both these applications, the limited display area made it necessary to introduce various strategies to filter and hide data, such as hidden hierarchi-

cal structures in WEST and usage-sensitive context displays in PowerView. This is indicative of that focus+context visualizations such as flip zooming are in themselves not sufficient when presenting data on small displays; instead, there is need for combinations where various data selection techniques are combined with visualization in an intelligent manner.

Thus, it should be fruitful to explore of how focus+context visualization and usage context (in a wide sense) can be utilized on handheld computers. If a PDA is combined with a mobile phone, we will have access to information such as who the user is currently talking to on the phone. This information can be used to adapt the context display so that for instance when a certain person calls, only the information relevant to this specific caller is displayed on the PDA. Other types of contextual information might also be used to adapt the information display on a mobile device. The size of the focus and thumbnails might change according to the light conditions; the level of ambient noise might affect if audio cues might be used to support interaction, and so on. Thus, the concept of “context” in focus+context visualization can take on a whole new meaning, and flip zooming, being a light-weight, adaptable technique, should adapt well to this change.

8 Acknowledgements

Thanks to all my collaborators in the various projects on which this paper is based, in particular Staffan Björk and Johan Redström, who contributed many important developments, and Christopher Ahlberg, who was my supervisor in the Zoom Browser project. The Flip Zooming Image Browser was implemented by Roberto Busso. This work was funded by NUTEK through the IVES project; by KFB through the Internet Project; and by The Foundation for Strategic Research through the Interactive Institute. Additional funding was received from SITI through the Mobile Informatics research program.

9 References

1. Bederson, B., Hollan, J., Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of ACM UIST '94*, ACM Press, 1994.
2. Björk, S. Hierarchical Flip Zooming: Enabling Parallel Explorations of Hierarchical Visualizations. *Proceedings of Advanced Visual Interfaces 2000*, ACM Press, 2000.
3. Björk, S. and Holmquist, L.E. Formative Evaluation of a Focus+Context Visualization Technique. *Proceedings of HCI '98* (poster), The British HCI Society, Sheffield, UK, 1998.
4. Björk, S., Holmquist, L.E., Ljungstrand, P. and Redström, J. Power-View: Structured Access to Integrated Information on Small Screens. *Extended Abstracts of CHI 2000*, ACM Press, 2000.
5. Björk, S., Holmquist, L.E. and Redström, J. A Framework for Focus+Context Visualization. *Proceedings of IEEE Information Visualization '99*, IEEE Press, 1999.
6. Björk, S., Holmquist, L.E., Redström, J., Bretan, I., Danielsson, R., Karlgren, J. and Franzén, K. WEST: A Web Browser for Small Terminals. *Proceedings of ACM UIST '99*, ACM Press, 1999.
7. Card, S.K., Mackinlay, J.D and Shneiderman, B. (eds.) *Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, San Francisco, California, 1999.
8. 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.
9. 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.
10. Furnas, G.W. Generalized Fisheye Views. *Proceedings of CHI '86*, pp. 16-23, ACM Press, 1986.
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. Keahey, T. The Generalized Detail-In-Context Problem. *Proceedings of IEEE Visualization '98, Information Visualization Symposium*, IEEE Press, 1998.
16. 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.
17. 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.
18. 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.
19. Maes, P. Agents that Reduce Work and Information Overload. *Communications of the ACM*, Vol. 37, No.7, pp. 31-40, 146, ACM Press, July 1994.
20. Perlin, K. and Fox, D. Pad: An Alternative Approach to the Computer Interface. *Proceedings of SIGGRAPH '93*, ACM Press, 1993.
21. Robertson, G.G., Mackinlay, J.D., The Document Lens. *Proceedings of UIST '93*, pp. 101-108, ACM Press, 1993.
22. Sarkar, M. and Brown, M.H. Graphical Fisheye Views of Graphs. *Proceedings of CHI '92*, pp. 83-91, ACM Press, 1992.
23. 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.
24. Shneiderman, B. Dynamic Queries for Visual Information Seeking. *IEEE Software*, 11(6), pp. 70-77, 1994.
25. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. *Proceedings of IEEE Visual Languages '96*, pp. 336-343, IEEE Press, 1996.
26. 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.

