Starfield Information Visualization with Interactive Smooth Zooming

Ninad Jog† and Ben Shneiderman*
Human Computer Interaction Lab
Institute for Systems Research
University of Maryland
College Park MD 20742-3255 USA
e-mail: ninad@visix.com, ben@cs.umd.edu

† Current address: Visix Software, Inc. Reston, Virginia.
* Department of Computer Science

ABSTRACT

This paper discusses the design and implementation of interactive smooth zooming of a starfield display (which is a visualization of a multi-attribute database) and introduces the zoom bar, a new widget for zooming and panning. Whereas traditional zoom techniques are based on zooming towards or away from a focal point, this paper introduces a novel approach based on zooming towards or away from a fixed line.

Starfield displays plot items from a database as small selectable glyphs using two of the ordinal attributes of the data as the variables along the display axes. One way of filtering this visual information is by changing the range of displayed values on either of the display axes. If this is done incrementally and smoothly, the starfield display appears to zoom in and out, and users can track the motion of the glyphs without getting disoriented by sudden, large changes in context.

KEYWORDS

starfield display, smooth zooming, animation, zoom bar, dynamic queries, information visualization, focal line.

INTRODUCTION

Exploring large multi-attribute databases is greatly facilitated by presenting information visually. Then users can dynamically query the database using filtering tools that cause continuous visual updates at a rate of at least 15 frames per second [5]. Such dynamic query applications typically encode multi-attribute database items as dots or colored rectangles on a two-dimensional scatter gram, called a starfield display, with ordinal attributes of the items laid out along the axes.

Geographic applications arise as natural candidates for dynamic queries by representing latitude and longitude along the axes - thereby making the starfield display a map of locations. Other databases exploit the starfield display by mapping two ordinal attributes along the axes and using a third to color code the dots. Additional attributes can be controlled by widgets such as sliders and buttons.

Many applications can employ Visual Information Seeking (VIS) principles [1, 14, 10] to facilitate rapid information browsing and empower users to find patterns and exceptions at a glance. VIS principles encompass direct manipulation, rapid query filtering using sliders and buttons, immediate and continuous visual updates of results, tight coupling - i.e. interrelating query components to preserve display invariants and zooming the starfield display to reduce clutter. The users begin with an overview, zoom in on areas of interest, filter out unwanted items and then get details on demand.

Unlike traditional applications such as image browsers that do zooming in large fixed stages, zooming a starfield display should be incremental and flicker-free so that users can track the motion of each rectangle. This gives users a feeling of flying through the data instead of getting disoriented by sudden large changes in view. As Bederson and Hollan affirm in their work on Pad++ sketchpad [3], zooming should be a first-class interaction technique. This paper deals with the design and implementation of zooming on a prototype dynamic queries application, the FilmFinder [1].

Whereas zooming an arbitrary image in real time necessitates computations at every pixel, zooming a starfield display is a simpler problem because computations have to be done only for the colored rectangles - not for the background - and there are just hundreds to thousands of rectangles as opposed to a million pixels.

THE PROTOTYPE APPLICATION

The prototype application that we used as a substrate was the FilmFinder (figures 1.1-1.3), which is a visualization of a database of two thousand movies. Each film has multiple attributes such as title, director, year of release, popularity, the lists of actors and actresses, length in minutes, category (drama, comedy, horror, etc.) and rating.

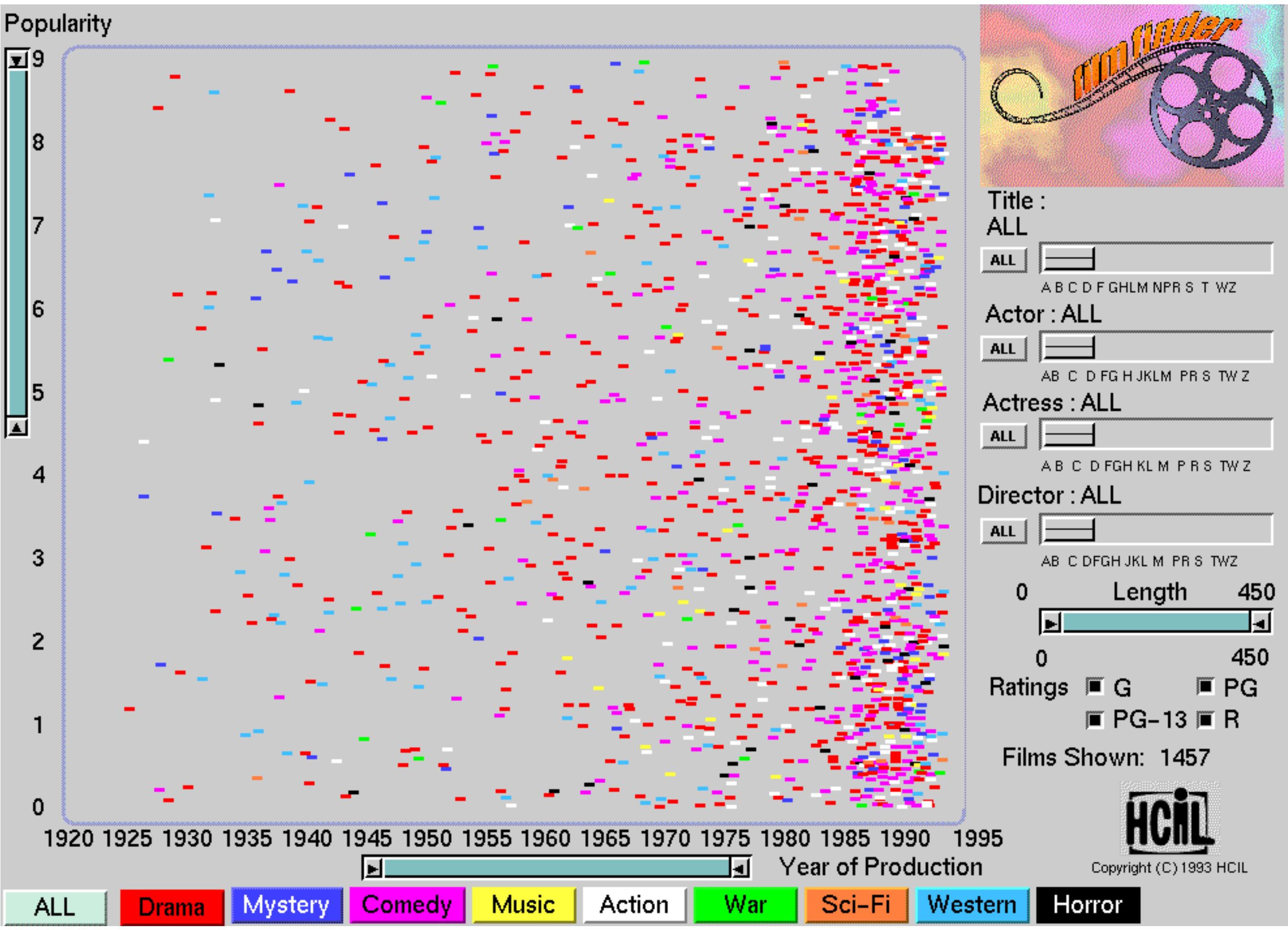

Figure 1.1 The Film Finder with a fully zoomed-out x-axis view, showing films from 1920 onwards.

The starfield display is formed by plotting each film as a small colored rectangle, with its popularity (scale 1 to 10, where 10 = most popular) on the y axis and the year of release (1920 through 1993) on the x-axis. Therefore recent popular movies appear at the top right of the starfield display. Categories are color coded, so that dramas appear as red rectangles, musicals as yellow, etc. The database is static and has more recent movies than old ones i.e. the distribution of data is non-uniform.

Clicking on a rectangle pops up an information card that lists the attributes of the selected film and shows a still picture from it. The number of rectangles that appear in the starfield display can be controlled by selecting attributes of the films (e.g. show just dramas) or by changing the scale on the x and y axes (e.g. show films made between 1940 and 1970).

These query filters are implemented using widgets such as toggles for category and rating; Alpha Sliders [2] for the title, actors, actresses, director; a range selection slider [5] for movie length and a new widget called the zoom bar for varying the scale on the x and y display axes.

GLOBAL AND LOCAL EFFECTS OF VARYING ATTRIBUTES

We distinguish query filters that have local effects and those whose effects are global. For example filters such as an Alpha Slider for selecting the name of an actor affect a small number of display rectangles compared with the total number of films in the database, whereas category toggles have a global effect both in terms of the large number of films they affect and the large display area over which these changes take place.

Since a zooming action changes the scale on one of the display axes, forcing a redisplay of all rectangles, the scale change filter is a global-effect filter. The classification of attributes as global-effect and local-effect ones is highly application-dependent. For example if a query such as "Display all films whose directors' names begin with B" were to be supported, the director would be classified as a global-effect attribute. Because changes to global-effect attributes take a longer time to render, special data storage and access techniques were designed to speed up the display refresh rate.

THE ZOOMING MECHANISM

Zooming is done by changing the range of attributes on either axis individually - e.g., when the upper limit of years is continuously decreased from 1993 to 1960, the scale on the x-axis keeps increasing. Rectangles representing recently-made movies keep leaving the display and the ones that are within the range grow in size and move so that the display range occupies the entire width of the starfield, looking like a patterned rubber mat getting stretched.

On the other hand, varying any of the non-axis attributes such as category or film length causes rectangles to drop out of the starfield or get added to it, but there is no scaling or movement involved.

A TAXONOMY OF ZOOMING METHODS

An image can be zoomed continuously or in discrete steps. Zooming in discrete steps is employed when substantial computations are involved in drawing the new view (e.g. zooming an arbitrary picture), or when there is nothing to be gained by doing a continuous zoom, as in changing the view size of text in a desktop publishing application.

Continuous zooming is important to give the user a feel of flying through a space - say a world of 3-D graphical objects as in virtual reality applications, or in an information visualization like a starfield display. This allows users to get more detail in areas of intense interest and preserve the sense of location in the surrounding items.
Since continuous zooming requires rapid redrawings, the image must consist of simple objects that can be hierarchically structured. It is difficult to do it on an arbitrary image. Another way of classifying zooming is based on the effects it has over the entire image.

1) Some zooming methods typically use a lens that can be moved over an image. The magnified portion appears within the lens boundaries or in a separate window, while the rest of the image stays undistorted. In either case, there is a sharp discontinuity at the boundaries, so users need to mentally integrate the two views. Ghostview, an X-Window application for viewing postscript files uses such a lens, as does the Magic Lens [4].

2) With a fisheye lens [12], the selected object is magnified the most, and surrounding objects are progressively diminished in size, giving a perspective view that retains both the focus and the context. Fisheye views free the user of the burden of mentally integrating two discontinuous pictures, but they always retain all the information on the screen, making it look cluttered.

3) The third type of zooming changes the scale on one or all of the display axes, causing information to leave or enter the viewing area. It changes the amount of information viewed within the focus area without changing the focus area size, and is identified as the canonical *adjust* operation in [9]. This has the advantage of uncluttering the screen, but if it is done in big discrete jumps, the user can feel disoriented.

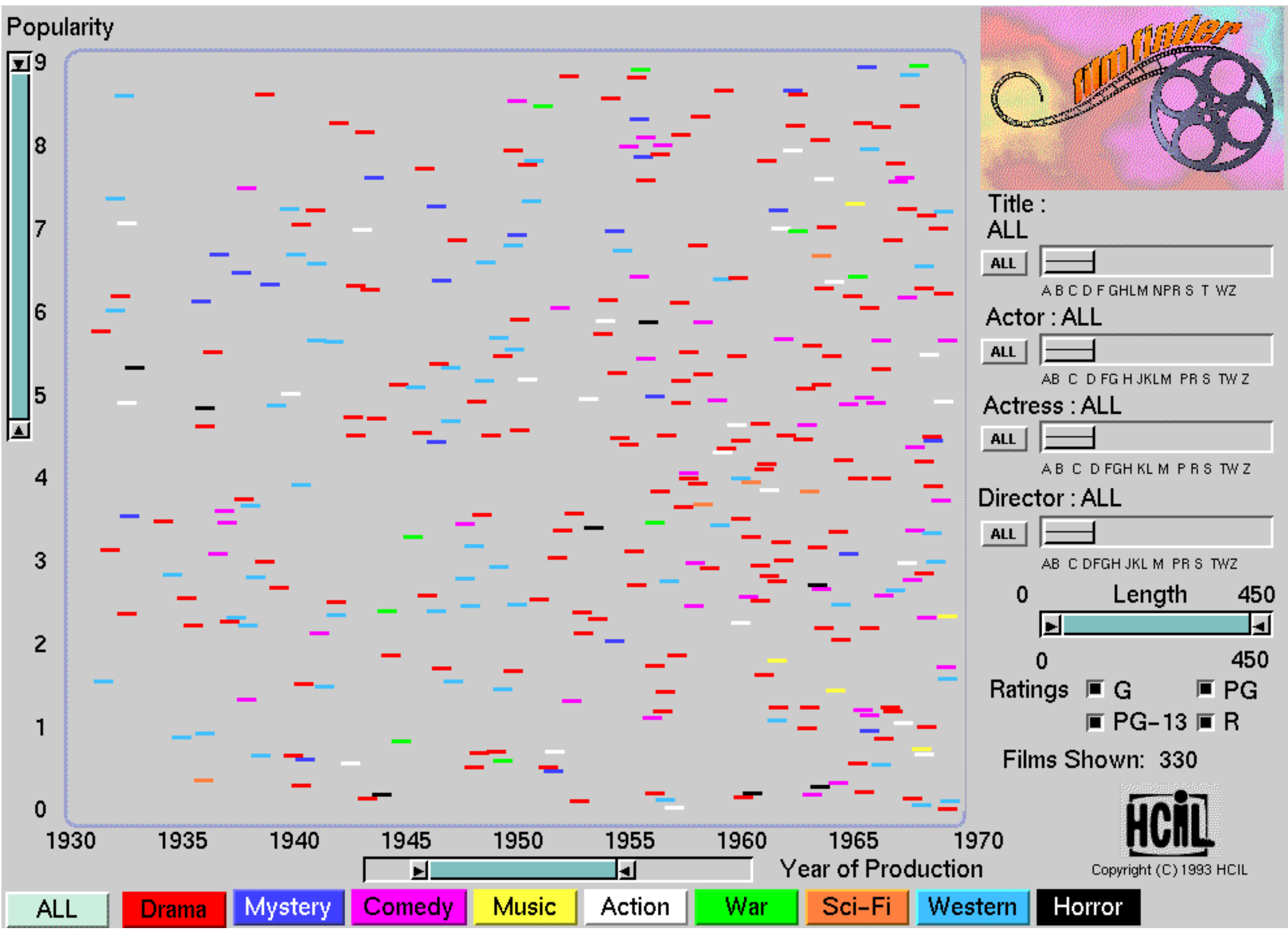

Figure 1.2. A partially zoomed-in view, showing films from 1930 thru 1970

Any of these three zoom methods can be done either continuously or in discrete jumps depending upon the complexity of the image. The FilmFinder uses the third of the above methods in a continuous manner. Figures 1.1 through 1.3 show three successive views of the starfield display when zooming is done along the x-axis by decreasing the upper range boundary of the films' year of release.

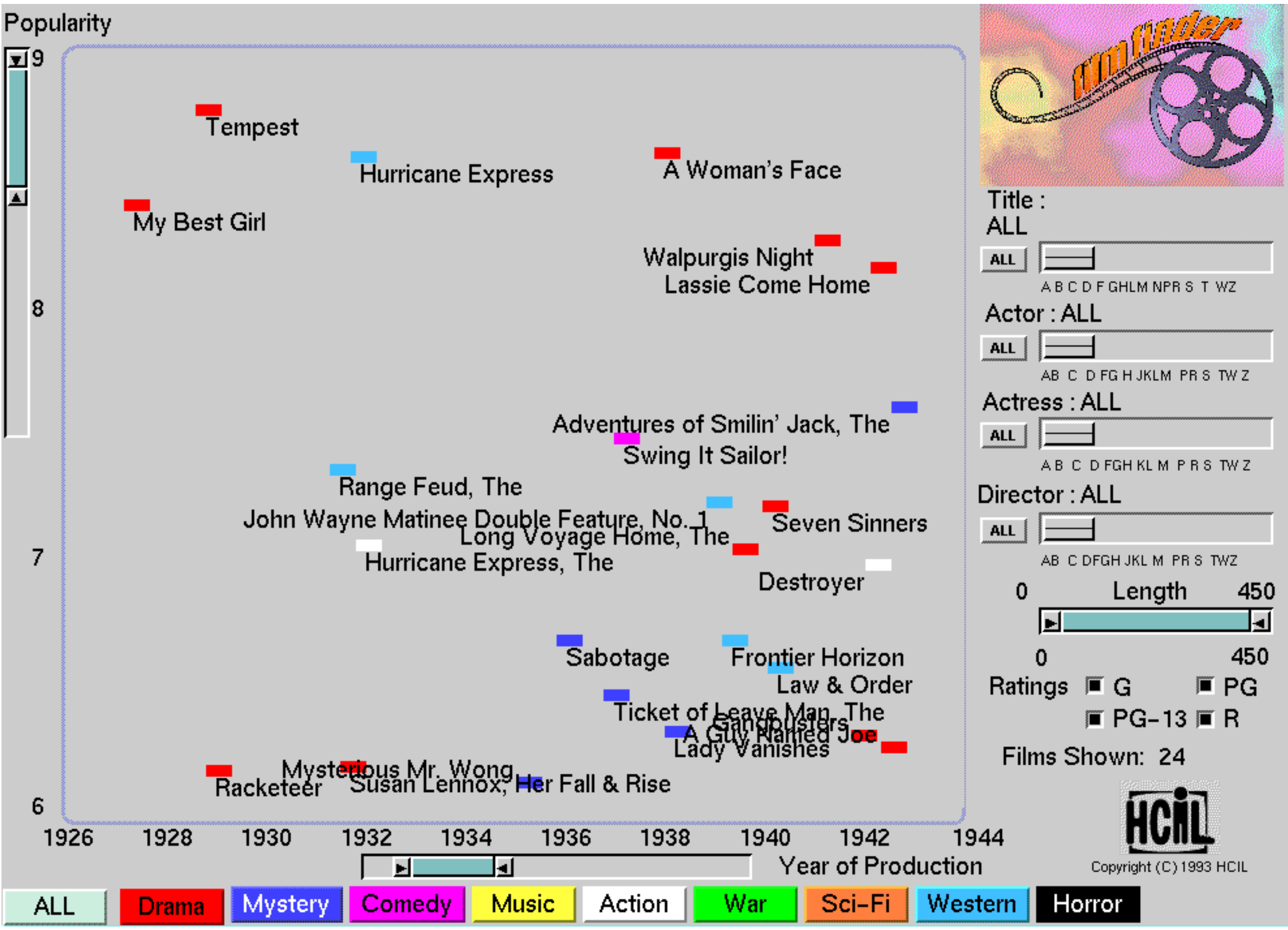

Figure 1.3. A zoomed-in view, showing films from 1926 thru 1944

PURPOSE OF ZOOMING

Zooming an information visualization display can have twin purposes. When used on images or hierarchy-oriented diagrams such as network node-link diagrams [13], successive views can reveal previously hidden detail. For example, each node in a node-link diagram of a network may function as an icon of a sub network, so that zooming in would reveal the details of the sub network as another node-link diagram one rung lower in the hierarchy of networks - what is termed semantic zooming [3].. This use of zooming is akin to magnifying an image at a selected point.

The other use of zooming is to reduce visual clutter by filtering out data points that lie outside the new zoom range. The FilmFinder uses zooming to achieve the latter effect. Due to zooming, rectangles that overlapped partially in the zoomed out view get spread apart, making it easier to click on them.

To make the zooming seem more realistic, rectangles change size as they are zoomed, but the change is bounded to prevent them from shrinking into nothingness or growing to occupy a large part of the screen. It is also easier to select a larger rectangle than a smaller one, and the amount of zoom can be gauged by looking at the size of the rectangles.

Zooming is trickier to do than panning:

1) Panned objects translate by the same amount without changing in size but zoomed objects change size and also move by different amounts depending upon their distance from the focus. This means that zooming exacts more extensive geometrical recomputations than panning.

2) The amount of translation of each zoomed rectangle in either the x or y direction is a function of both the current range boundaries, making it impractical to precompute increments and store in a lookup table.

ZOOM BAR

Existing zoom widgets such as user-draggable lens tools are suited for applications where zooming takes place in jumps over small image areas, but they don't work when zooming is continuous over the entire screen and is triggered by changes to a range boundary.

Initiating a zoom using a mouse button works when we wish to zoom towards or away from a fixed point, but the interface is not obvious to the user - there is no on-screen widget to provide a visual cue. Another disadvantage of this method is that there is no feedback to users about the degree of zooming in the current display.

We tried using a pair of buttons for increasing and decreasing each range boundary, but the use of these buttons proved confusing to users - not only were they unsure of whether a button decreased or increased a boundary, but it was also hard for them to see the link between changing a range boundary and the concomitant zooming effect.

We overcame this deficiency by developing a new widget called the zoom bar (See figure 2), which is a slider with three thumbs. The two extreme thumbs are used to adjust range boundaries. When the right thumb is moved, the upper range boundary is increased or decreased, causing a zoom out or a zoom in by changing the scale on the corresponding display axis. Similarly, the left thumb controls the lower range.

The middle thumb is used to pan over the display range. Its size varies according to the positions of the left and right thumbs, thus changing the width of the window that pans over the data.

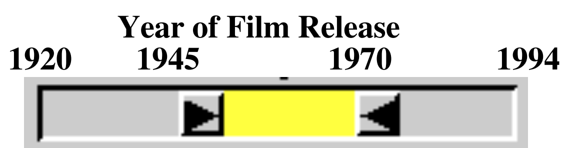

Figure 2. The zoom bar: The viewing range is varied by sliding either of the two thumbs. Panning is done using the middle thumb.

The middle thumb has a minimum width, which means that the left and right thumbs can come close together by no more than a specified separation. This separation defines the maximum zoom in of the view. When the two thumbs are at the opposite ends of the slider, the view is fully zoomed out and the middle button is disabled. The scale of each of the x and y display axes is varied individually by its own zoom bar.

When the zoom bar is clicked in the channel on a position other than the thumb, the thumb closest to the clicked point snaps to that location. This causes a jump in one of the range values and results in discrete zooming.

Strengths and Weaknesses

The zoom bar is intuitive and easy to use because of its similarity with a scrollbar. It occupies a small rectangular area, saving on precious screen space and its operation is rapid because of its small size. The zoom bar also provides clear feedback of the degree of zooming in the current display. It is well suited for both continuous and discrete zooming as well as for panning (sliding a fixed range over the data). However, its scope is restricted to one display dimension at a time.

AN ALTERNATIVE TO THE CAMERA MODEL

The single-axis-at-a-time-zooming of the prototype application cannot be described in terms of a camera using a focal point. This section describes an alternative model.

While traditional continuous zoom techniques give users the impression of flying perpendicular to a plane (the starfield display plane) towards a fixed point, our novel zooming technique gives the impression of a rubber carpet getting stretched and contracted, with the user standing at a fixed distance from it. This fundamental difference arises because our zooming scheme provides for independently alterable zoom factors for the x and y axes.

These different zooming techniques can be described as the view seen through a camera which has a variable focal length and whose position can be changed.

Traditional zoom techniques employ one of two possible schemes. Either the camera stays at a fixed place and the focal length changes over time - as in a real camera, or the camera's field of view remains fixed and the camera moves towards or away from a fixed point. If either case were used in a starfield display, rectangles would appear to move away from the focal point in all directions during zooming in, and appear to converge toward the focal point during zoom out, as shown in figure 3.

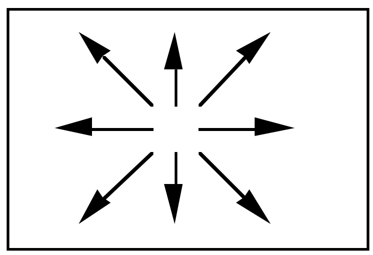

Figure 3. Zooming in causes rectangles to move radially outward from the focal point.

In contrast, zooming in the FilmFinder is done independently in the x and y directions, so there is no single focal point. Instead, there is a *focal line* corresponding to the range boundary that remains fixed. This line could be any of the four sides of the starfield display.

For example, if zooming is done in the horizontal direction and the upper range is increased, the focal line is the left side of the display. Rectangles that lie close to the left boundary move by very small amounts compared with the ones near the right boundary, though all rectangles shrink by the same amount.

As the range increases, rectangles enter the viewing range from the right and start moving leftwards. Figure 4 shows this behavior. In other words, rectangles flatten as they move toward the focal line and elongate as they move away from it.

If a traditional zooming technique with a user-defined focal point of interest is used in the absence of an additional overview screen, the user may feel lost even if the zooming is smooth. However, when the individual-axis zooming of this paper is used, the user's view is always firmly anchored to one of the sides of the starfield display. We conjecture that this leads to a better grasp of location and thereby improved user satisfaction.

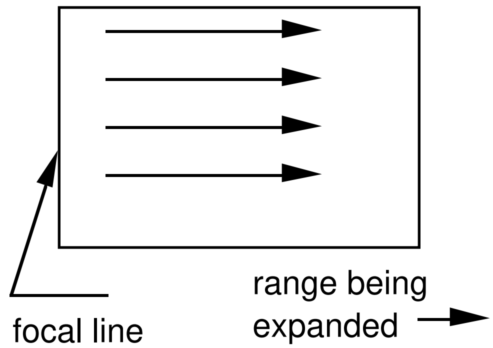

Figure 4. Zooming along x-axis. Arrows show motion of rectangle.

IMPLEMENTATION

The FilmFinder has been implemented on a Sun SPARCstation 1+ using Galaxy/C, a cross-platform application environment developed by Visix Software Inc. It can be ported to several platforms - from a slow DOS machine running Windows to a fast Sun, so it becomes imperative to seek software speedups and optimizations to get a rapid display refresh rate instead of relying solely on the use of faster hardware.

Galaxy's object-oriented constructs were used to build customized widgets and other parts of the program. The film data is in flat-file format, taken from the Internet. The entire data is read into a linear array and sorted by several other attributes like length, actors, etc. Each of the individual sorted lists contains pointers (indices) to the linear-array database i.e. the set of all the records are always in the linear array in the order in which they were read in.

ATTAINING SMOOTH ZOOMING

The following strategies were used to attain smooth zooming.

Efficient storage

The items from the database are stored in an array and direct indexing based on the attribute values is used to access them.

Rapid access

Items that are being displayed are cached into a contiguous array, so when the display range changes, searches are limited to a smaller subset of items.

Double buffering

Successive frames are composed off-screen and dumped onto the display, thereby eliminating the flicker caused by an erase and redraw operation pair.

Increased axis resolution

The display positions of item rectangles in successive frames are placed close together so that the animation appears smooth.

FUTURE WORK

Zooming can be made faster and smoother by extending the above techniques. When the screen has a large number of rectangles, draw just half of them alternately, as done in the X-Window application xgas. The buffering technique can be varied: instead of erasing and redrawing rectangles on the buffer, an updated image of the union of the old and new areas of each rectangle can be copied onto the buffer.

An important challenge is to find an upper limit on the number of rectangles that can be displayed before the illusion of zooming fails, and to get a concrete measure of the maximum speed of the rectangles (say 1 cm. per second) that can be tolerated.

A more challenging problem is to visualize a database containing 50,000 items. Such a visualization can be made by displaying a small number of representative rectangles [8] and zooming in to reveal the hidden ones - using zooming for its traditional purpose of revealing more detail. Handling such huge amounts of data would necessitate the use of linked data structures like k-d trees, range trees or quadtrees [11].

A 3-D display of rectangles might be an appealing alternative, but there's the danger of items obscuring each other and of the user getting lost in the "star-tank".

Although the zoom bar is an adequate tool, an alternative tool like a resizable rectangle roving over a miniature overview of the starfield display would permit both zooming and panning over both axes to be done with the same widget.

Another goal is a flexible starfield widget that can visualize many databases - such a visualization would determine the type of each attribute (integer, string, etc.) and display the corresponding proper widget for controlling it (such as an ordinary slider or an Alpha Slider, etc.)

CONCLUSIONS

This paper discussed the design and implementation of a smooth zooming mechanism in a dynamic queries application, presented a taxonomy of zooming methods and introduced the zoom bar, an intuitive and rapid widget to facilitates zooming and panning. Smooth zooming of items in a database visualization was achieved by reducing the data access and display bottlenecks.

ACKNOWLEDGMENTS

We thank Christopher Ahlberg for implementing the initial version of the FilmFinder, and for his suggestions and review of the paper. We thank Bruce Chih-Lung Lin for identifying efficient storage and access techniques, Andries van Dam, John Hughes and Marko Teittinen for helping define zooming with the camera model, and David Mount for helpful suggestions.

Thanks are also due to Richard Chimera for numerous suggestions, to Richard Potter for his review and to Visix Software Inc. for providing us their cross-platform application builder, *Galaxy*. Thanks to Teresa Casey and Ara Kotchian for helping with the images. Finally, we thank the Institute for Systems Research for their support. This research was supported by grants from the National Science Foundation, NSFD CDR-8803012 and NSF EEC 94-02384.

REFERENCES

1. Ahlberg, Christopher and Shneiderman, Ben. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *Proc. of CHI 1994.* ACM, New York, pp 313-317.

2. Ahlberg, Christopher and Shneiderman, Ben. The AlphaSlider: A Compact and Rapid Selector. *Proc. of CHI 1994.* ACM, New York, pp 365-372.

3. Bederson, Benjamin and Hollan, James. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proc. of UIST 1994*, ACM, New York.

4. Bier, Eric, Stone, Maureen, Fishkin, Ken, Buxton, William and Baudel, Thomas. A Taxonomy of See-Through Tools. *Proc. of CHI 1994*. ACM, New York, 1994, pp 306-312.

5. Carr, David, Jog, Ninad, Kumar, Harsha, Teittinen, Marko, Chimera, Rick and Ahlberg, Christopher. The HCIL Widgets: Motivation, Specification, Development and Use. *CAR-TR-734* University of Maryland, College Park 1994.

6. Foley, James, van Dam, Andries, Feiner, Steven and Hughes, John. *Computer Graphics, Principles and Practice.* 2nd edition, Addison Wesley, Reading, MA 1990.

7. Jain, Vinit and Shneiderman, Ben. Data Structures for Dynamic Queries: An Analytical and Experimental Evaluation. *Proc. Conference on Advanced Visual Interfaces '94,* ACM, New York 1994, pp 1-11.

8. Keim, Daniel, Kriegel, Hans-Peter and Seidl, Thomas. Visual Feedback in Querying Large Databases. *Proc. of IEEE Visualization 1993,* pp 158-165.

9. Rao, Ramana and Card, Stuart. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. *Proc. of CHI 1994,* ACM, New York, pp 318-322.

10. Robertson, George, Card, Stuart and Mackinlay, Jock. Information Visualization Using 3-D Interactive Animation. *Communications of the ACM, 36*, Number 4, April 1993, pp 57-71.

11. Samet, Hanan. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, Reading, MA, 1990.

12. Sarkar, Manojit and Brown, Marc. Graphical Fisheye Views of Graphs. *Proc. of CHI 1992,* ACM, New York, pp 83-91.

13. Schaeffer, Doug, Zuo, Zhengping, Bartram, Lyn, Dill, John, Dubs, Shell, Greenberg, Saul and Roseman, Mark. Comparing Fisheye and Full-Zoom Techniques for Navigation of Hierarchially Clustered Networks. *Research Report # 92/491/29,* University of Calgary, November 1992.

14. Shneiderman, Ben. Dynamic Queries for Visual Information Seeking. *IEEE Software,* Volume 11, Number 6, November 1994, pp 70-77.