Non-Linear Image Magnification

T. Alan Keahey*
tkeahey@cs.indiana.edu

Edward L. Robertson
edrbtsn@cs.indiana.edu

Department of Computer Science
Indiana University
215 Lindley Hall
Bloomington, IN 47405
Phone: (812) 855-3609
Fax: (812) 855-4829

April 24, 1996

Abstract

In this paper we describe a new level for the application of non-linear magnification transformations. The transformations are applied at the *image-space* level, composed of discrete pixels in a discrete domain. This domain offers a broad range of applicability to visualization tasks, and can serve as a "visualization front-end" to any graphical display. These techniques make use of increasingly available graphics hardware to provide a highly interactive viewing and magnification environment.

Keywords

Visualization, image, magnification, fisheye, texture mapping, warping.

*This work was supported by US Dept. of Education award number P200A502367.

1 Introduction

The problem of limited screen space is becoming more significant as current computer systems and applications become increasingly graphics oriented. Although screen sizes of 1280×1024 are not uncommon, they are still not large enough to display many complex visual scenes. Conversely, as screen sizes become larger, more amounts of information can be placed on a single screen with the potential for overloading the user. General purpose techniques to enhance visualization of some screen area(s) could allow the user to better focus on areas of interest. These concerns are of particular importance for visually impaired users, who often have difficulty viewing the contents of a screen.

A wide variety of magnification techniques – far beyond the conventional magnifying glass – can be used to alleviate these problems. This paper will discuss principles and implementation of general purpose non-linear magnification techniques which provide for continuous levels of non-occluding magnification across a variety of domains.

In section 3, we briefly discuss some general techniques for non-linear magnification transformations. In the following sections, we will show how special computer graphics techniques can be used to apply these transformations to a specific – and very broadly applicable – set of visualization tasks. Special graphics hardware is becoming increasingly available which allows these techniques to be applied in a highly interactive user-oriented fashion.

2 Levels of Application

There has been much work in the Visualization/HCI community involving application of non-linear (fisheye, hyperbolic, distortion) magnification techniques to different domains. Most of the work involves the application of these techniques to discrete objects in a continuous coordinate space, before the objects are rendered (the most common examples involve some instance of graph visualization). We refer to this method as an application of non-linear magnification in *render-space*.

In this paper we will show that there is another level at which these techniques can be applied: *image-space*. Here we are applying the transformations to discrete pixels in a discrete coordinate space. At this level there is no concept of "objects" to be rendered, other than individual pixels. Instead we are applying the transformations to objects or scenes which have *already been rendered*.

The image-space techniques presented here are not intended to supplant the render-space techniques, but rather to add another level at which visualization can be enhanced through non-linear magnification transformations.

3 Magnification Transformations

In this section we briefly describe some issues and techniques for magnification. The discussion is provided largely as a basis for following sections.

3.1 Linear Transformations

The most familiar magnification technique involves the application of linear transformation functions as shown in Figure 1. The result of these transformations is a constant level of magnification across the domain, very much similar to what someone would perceive through an or-

dinary magnifying glass. Such techniques have an advantage of providing a close analogy to familiar experiences for the user, however certain limitations are inherent with this approach:

Figure 1: Magnified View of Boris Yeltsin's Nose

- The user is forced to create a mapping between disjoint levels of resolution in the image. While it can be argued that the user will already be somewhat familiar with this task, it can also be argued that very rarely in the real-world does our perception of objects follow this type of behaviour: when viewing a physical object, our levels of perceived detail tend to follow a smooth continuum, rather than jump in and out to different levels of resolution.

  We would like to implement a system which more naturally mimics the continuous levels of detail offered in real world systems. The user should be able to gradually draw closer to areas of interest (thus increasing the resolution) and move away from areas of disinterest (decreasing the resolution).

- If the magnified representation of some area of the screen is to appear in some other part of the screen (i.e. a separate window), then the user will be forced to make abrupt transitions on *two* cognitive/perceptual levels to tie the representations together. The first transition is spatial: the eyes (and possibly input devices) will have to move from the normal resolution version of the image to the higher resolution version. The second transition is that of resolution, the user will be forced to create a mapping between the displayed levels of resolution.

- If we attempt to use *in situ* linear magnification techniques, where the magnified image sits on top of the normal resolution image, then we introduce an additional problem, that of *occlusion*. Because of the linear nature of this magnification, the magnified representation of an image must necessarily be larger than the non-magnified representation, the result of this is that the magnified image will block neighboring areas of the non-magnified image, so that the user is no longer able to see the entire image (at any resolution) without moving the magnifying area to see what is underneath it.

3.2 Non-Linear Transformations

In order to overcome some of the problems with constant magnification, several different *non-linear* (or *distortion-based*) magnification transformations have been developed. These systems all share the properties of enhanced local resolution with some preservation of global context. *Fisheye zoom* [2] [12] is perhaps the most widely known example of such a technique, although many other systems have been developed, including piecewise linear approximations. Leung and Apperly [8] provide an overview of some of these techniques. Here we briefly present a few ways in which non-linear transformations can be applied effectively. These examples are illustrated by showing the effect of such a transformation on a regular grid of points. Many other transformations are possible and have been implemented.

Vertical: For some applications, a simple magnification in only one dimension is useful. For example, magnification in the vertical

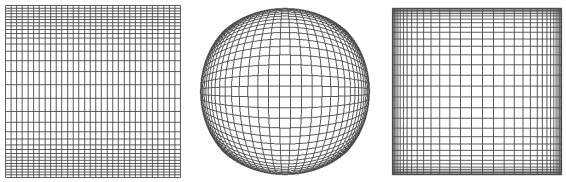

Figure 2: Vertical, Radial and Orthogonal Transformations

dimension can allow for laying out large text files into a single tall and narrow frame which can be scrolled through. Figure 3 shows a simple example of how a structured text file that would normally require several screens to display can be laid out in a single screen [5]. Another example of a text viewing application using uni-dimensional magnification (with multiple focus points) can be found in [4].

Figure 3: Simple Text Viewing

Radial (Fisheye): The fisheye lens effect can be produced by transforming the radius component of the polar coordinates of points within the domain. This transformation preserves angles relative to the center of the magnification image, and the fisheye effect is familiar to most users, providing a ready analogy for the user to relate to. A single magnification parameter controls degree of magnification in all directions.

Orthogonal: This transformation is the result of applying the transformation to the x and y components of a point separately, thus making the magnification in one dimension *orthogonal* to magnification in other dimensions. This transformation preserves horizontal and vertical lines within the domain, and also allows for independent control of the magnification parameters for the x and y axes.

Different levels of *dimensionality* are present in the vertical (1D), radial (1.5D), and orthogonal (2D) transformations. This will be relevant in our discussion of discrete domain characteristics in section 5.

3.3 Combined Linear/Non-Linear

A major benefit of linear transformations is that they produce relatively distortion-free zooming (when the magnification factor is the same in both dimensions, although aliasing effects may still be present). For example, a user viewing textual data would usually prefer to see the letters at the area of maximal magnification without the distortions presented by non-linear transformations.

Figure 4 shows two examples of combining linear and non-linear transformations. In the left image, constant magnification is applied to a rectangular area, and the surrounding points are transformed by an orthogonal non-linear magnification. In the right image, constant magnification is applied to a round area, with the surrounding points being treated by a radial non-linear magnification. In both cases we are able to combine the best properties of linear (distortion free) and non-linear (non-occluding) transformations. A fuller discussion of techniques for combining linear and non-linear transformations can be found in [7] and in [13].

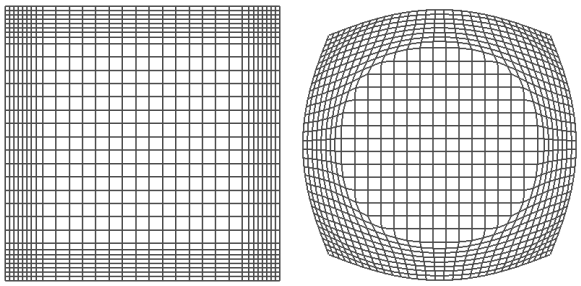

Figure 4: Combining Linear and Non-Linear Transformations

3.4 Bounded Transformations

Often it is the case that we do not want the transformation to apply to the entire source domain, but would rather perform non-occluding magnification on some sub-area of the domain, constraining the transformed points from that sub-area to the same sub-area (to avoid overlap). This allows for a localized, non-occluding magnification which can be moved over the source domain. This localization offers the benefits that the global context now remains more static, and does not change as the center of the magnification is moved. Figure 5 shows examples of applying constrained domains to the same transformations described in section 3.3, compare these results with Figure 4. Techniques for producing bounded transformation are discussed in [7].

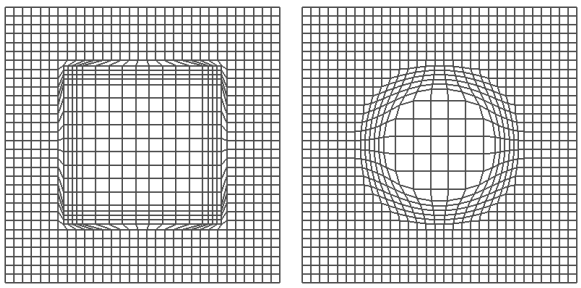

Figure 5: Bounded Transformation Domains

3.5 Compound Transformations

There are many different ways in which multiple transformations can take effect simultaneously on a given domain. Figure 6 illustrates instances of three different approaches. These are briefly described below, and more fully in [7].

Maximal Ray Clipping: produces a partitioning of the domain space similar to Voronoi diagram partitioning methods from computational geometry [11]. The effect of this technique is a partitioning of the space in which each transformation has its own "domain of influence", and none of the partitions overlap each other.

Weighted Averaging: produces a smooth contour representing the average of multiple transformations. However changes to one transformation are not localized with this method, and all points in the domain will be affected.

Composition: gives the effect of a stack of lenses layered on top of the display, each of which can be controlled independently.

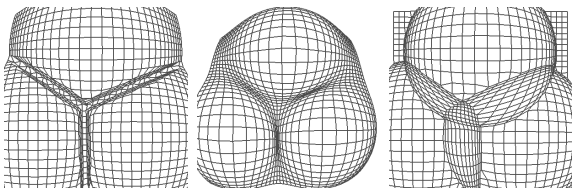

Figure 6: Clipped, Weighted Average, and Composition Transformations

3.6 Filtering Transformations

Independent of the complexity of the transformations employed, it is often useful to have a single mechanism for determining the degree to which

the warping transformations are to take effect. This can allow the user to smoothly zoom between the warped and unwarped representations of the space being displayed. Such functionality can be provided through a simple filter as described in [7]. Figure 7 shows an example of changing the filter parameters to alter the degree of the effect of several transformations simultaneously.

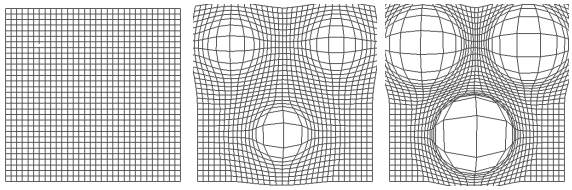

Figure 7: Filtered Transformations

4 Image Magnification

All of the transformations discussed so far have been described in terms of transforming a grid of points. In this section we will discuss how such transformations can be used to transform pixelized images. Such images might be ordinary gif or jpeg files, however they can also be any other pixel format, such as the general screen data described in section 6.3. The primary mechanism used for this is *texture mapping*, wherein pixel coordinates in an image are mapped to coordinates defining a surface. As the surface coordinates change, the image changes correspondingly. The image that is used in this process is called a *texture*, and the individual pixels of the texture are called *texels* (texture pixels). Texture mapping is a well known technique which is described in most graphics texts such as [1].

Figure 8 shows a texture being mapped onto a regular grid of points defining a flat, square surface. In figure 9 we transform the grid with a combined linear/non-linear radial transformation producing non-occluding flat magnification of the image and show the final image without the grid (see also color plate 1). Figure 9, also illustrates that the transformation function used does not take into account destination window boundaries, and small areas of the image can sometimes be clipped. The simplest solution to this problem is to create a narrow "buffer region" around the image into which the image can be expanded.

Figure 8: Mapping Images onto Regular Grid

The left-most image in figure 10 shows an example of *aliasing*, where magnification of pixels causes "jaggies" and interference patterns to develop (see also Colour Plate 2). This is the result of a single pixel in the source image being applied to more than one pixel in the destination image. For cases where accuracy of pixel representation is paramount, such aliasing is desirable. Several different *anti-aliasing* methods are available to help smooth out the magnified image, with the implementation described in this paper linear interpolation can be used to set destination pixels to be some linear combination of neighboring source pixels when the image is magnified. This *linear texture magnification* [14] will reduce the amount of aliasing, but may result in inaccuracies in the magnified image. This primitive

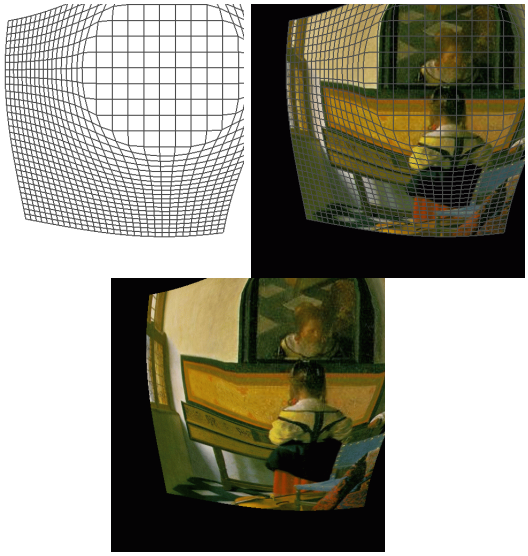

Figure 9: Distortion of Grid and Texture

anti-aliasing is more useful with images where dithering or anti-aliasing techniques were applied to the original image.

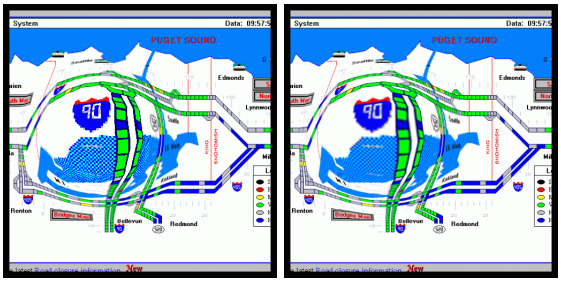

Figure 10: Nearest and Linear Texture Magnification

5 Continuous/Discrete Domain Characteristics

For tasks such as graph visualization, the domain consists of discrete objects in a continuous coordinate space, and (in general) adjacencies between nodes can be maintained through rendering of edges directly between them, regardless of the type or level of distortion introduced by magnification transformations. For this reason, graph visualization tasks are very amenable to non-linear transformation techniques, and there are many graph visualization systems which take advantage of this [2], [12], [10] and [6].

However, for discrete coordinate spaces (such as the texture mapping applications presented in this paper), adjacency requirements become more of a concern. This is particularly the case when linear/non-linear combinations or constrained boundaries are introduced. In these cases, boundaries between areas of different transformations must be carefully thought out to preserve adjacency information. The basic task is to ensure that all mappings between areas generate boundary conditions which are consistent with the other transformation functions that are used. Referring to figure 11, we can say that the boundary conditions at the perimeter of a region of magnification A should be the same for both $f(A)$ and $g(\sim A)$.

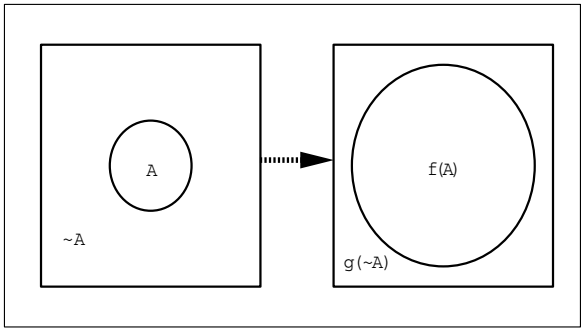

Figure 11: Boundary Conditions

With these considerations in mind, we can see in Figure 12 and Colour Plate 3 (particularly in the top left quadrant of each image) that the radial transformation is well suited to circular bounded domains, as both transformations (domain and magnification) only involve transforming the radius component of the polar coordinates, and produce the same boundary conditions. However, we can also see a problem presented when combining the orthogonal magnification and rectangular domain transformations.

Here the values are not the same for the two transformations at the perimeter of the rectangular boundary, with a resulting distortion.

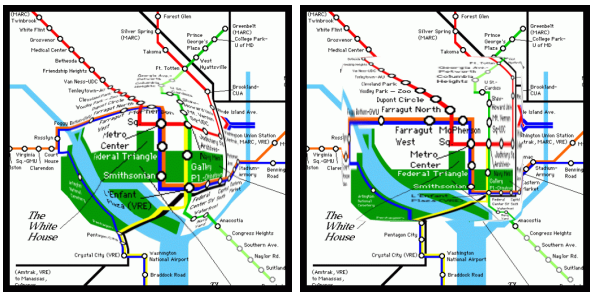

Figure 12: Circular and Rectangular Boundary Effects

In general, it is simpler to enforce consistent boundary conditions when the transformations apply to only a single variable. Thus transformations of lower dimensionality (as described in section 3.2) will be easier to construct constrained domains for.

6 Applications

6.1 Map Viewing

Non-linear magnification transformations are particularly well suited for tasks such as map viewing, where occlusion of neighboring areas is not desirable. As figure 13 shows, the radial transformation allows the viewer to follow streets from the magnified portion of the map to the unmagnified portion without having to deal with any discontinuities. This property of preserving lines between magnification zones is unique to non-linear image magnification, and would be particularly useful for such tasks as visualization of contour maps as shown in figure 14.

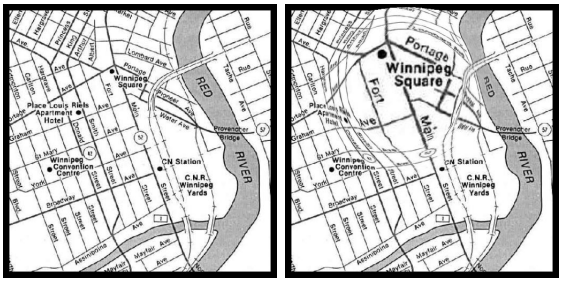

Figure 13: Street Map Magnification

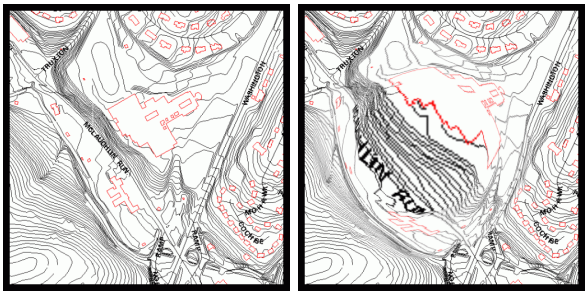

Figure 14: Contour Map Magnification

6.2 Text Viewing

Xdvi is a commonly used tool available on Xwindows systems for viewing the output of TEXtypesetting programs. Because of limitations in screen size and resolution, xdvi is not able to display an entire page at the resolution of a postscript printer. To allow for higher resolution views of the page layout, xdvi provides a pop up magnifying glass, figure 15 shows a typical example of this magnifying glass in use.

Figure 16 shows an application of non-linear magnification to the screen output of the xdvi program. While this example does illustrate high-levels of non-occluding magnification, it also points out an important difference between xdvi's magnification and the non-linear magnification of the xdvi screen output. When xdvi magnifies portions of the page, it has access to the higher resolution representation of the text, and will be able to more accurately fill in the details at higher levels of magnification. When

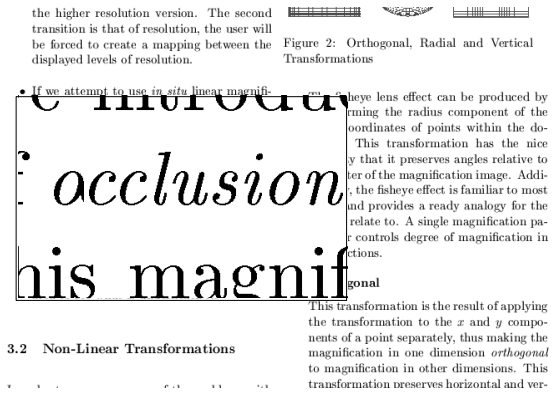

Figure 15: Xdvi Normal Magnification

we perform true screen magnification on the output of xdvi (as in figure 16), the magnification routine has no access to the higher resolution representation of the text being displayed. This is a limitation of the data access method, which could be resolved by applying the magnification at the render-space level rather than solely in image-space.

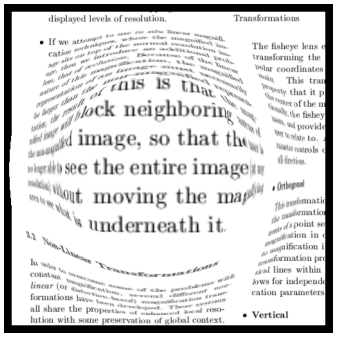

Figure 16: Non-Linear Magnification of Xdvi Output

6.3 General Screen Viewing

As mentioned previously, these techniques can be applied to more than just image files. Any pixelized data can also be used for the magnification. In particular, it is possible to grab and magnify arbitrary regions of the screen with these techniques.

Figure 17 shows an example of such a program in use. As the user moves the mouse over different areas of the screen the program grabs a rectangular region around the mouse, performs the non-linear magnification of the pixels, and displays the result in the magnification window. The viewer is then able to treat the texture that was grabbed from the screen as if it were an ordinary image file, altering the transformation(s) that are applied to it.

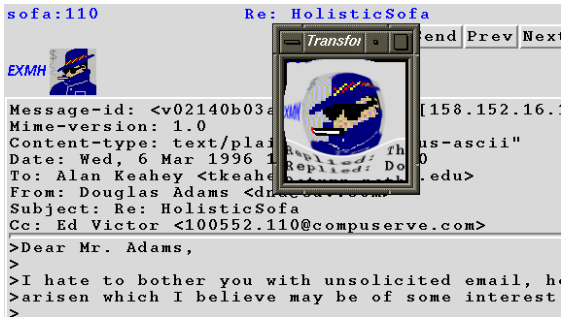

Figure 17: Screen Magnification

This result suggests a completely general purpose technique for applying non-linear magnification transformations to the output from *any* program. Although the current implementation does not yet support *in-situ* screen magnification, such functionality is certainly possible (albeit highly platform dependent).

This allows us to think of screen magnification as a post-processing phase, which applies to program and/or windowing system output. Program designers should be able to easily add an invisible "magnification window" to their programs which will rest on top of the other windows, and provide the user with a magnified, non-occluding view of areas of interest. In addition, it should be possible to incorporate such functionality at the windowing system level, designing interfaces which provide the user with in-situ non-occluding magnification of any arbitrary region(s) of the screen, without requiring modification of existing programs.

7 Implementation

All of the examples discussed in this paper were implemented with the FAD [1] toolkit developed by the author, which allows for construction and application of non-linear magnification transformations (both interactively and programmatically) over a wide range of domain types.

The FAD toolkit uses *OpenGL* [9] to implement the texture mapping used for the image magnification. OpenGL is widely available on many platforms, and appears to be emerging as the industry standard library for sophisticated interactive graphics programming. Furthermore, it is increasingly common to see desktop computers and workstations which offer hardware accelerators for OpenGL routines such as 3D viewing and texture mapping.

The development platform for these tools is a Silicon Graphics Crimson Reality Engine (150 MHZ MIPS R4400 CPU). The Reality Engine offers hardware support for texture mapping. Interactive frame rates using texture mapping to magnify a 512×512 image on a surface defined by a 32×32 point grid average approximately 65 FPS.

8 Conclusions

In this paper we have demonstrated how non-linear magnification techniques can be applied in the *image-space* domain. This domain puts additional constraints on which transformations are "useful" (e.g. boundary consistency constraints), and also does not offer all of the power possible in *render-space* transformations, since the data has already been rendered, and the transformation routine has no access to the original objects in the domain.

However, we have also shown that this transformation domain offers benefits in terms of very broad applicability, allowing non-linear magnification techniques to be applied to *any* data which has been rendered into a set of pixels. Such broad applicability offers great potential for the realization of visual enhancement techniques which are application and program independent. In effect this provides a "visualization front-end" which can be applied to any existing visual representation.

9 Further Work

Procedural texture mapping offers the potential for a more powerful class of transformations, which lie somewhere between the *image-space* and *render-space* techniques. This approach could allow the transformation to have some access to the data underlying the image when necessary.

Some OpenGL implementations allow sub areas of a given texture to be replaced with new textures. This could allow for higher resolution textures to be inserted into the magnified areas of an image. This multi-level texturing offers intriguing possibilities for the visualization of inherently hierarchical data.

References

[1] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[2] G. W. Furnas. Generalized fisheye views. *Human Factors in Computing Systems, CHI '86*, pages 16–23, April 1986.

[1] FAD == Future Acronym in Development

[3] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of CHI '95 Human Factors in Computing Systems*, 1995.

[4] S. Greenberg and C. Gutwin. Sharing fisheye views in relaxed WYSIWIS groupware applications. Technical Report 95/577/29, Department of Computer Science, University of Calgary, November 1995.

[5] T. Alan Keahey and Julianne Marley. Viewing text with non-linear magnification. Technical Report 458, Department of Computer Science, Indiana University, April 1996. Submitted to ACM UIST '96.

[6] T. Alan Keahey and Reid Rivenburgh. HyperLINK: A program for visualizing traversal of the WWW. Demonstrated at HyperText '96, 1996.

[7] T. Alan Keahey and Edward L. Robertson. Techniques for non-linear magnification transformations. Technical Report 455, Department of Computer Science, Indiana University, March 1996. Submitted to IEEE Visualization '96, Information Visualization Symposium.

[8] Y.K. Leung and M.D. Apperly. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[9] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide*. Addison-Wesley, 1993.

[10] Emanuel G. Noik. Exploring large hyperdocuments: Fisheye views of nested networks. In *Hypertext '93: ACM Conference on Hypertext and Hyeprmedia*, 1993.

[11] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.

[12] Manojit Sarkar and Marc H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–84, 1994.

[13] Manojit Sarkar, Scott S. Snibbe, Oren Tversky, and Steven P. Reiss. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *UIST '93, Proceedings of the ACM User Interface Software and Technology*, 1993.

[14] Mark Segal and Kurt Akeley. The OpenGL Graphics System: A Specification. Technical report, Silicon Graphics, Inc., 1995.