Generalized and Stationary Scrolling

Randall B. Smith
Sun Microsystems Laboratories
901 San Antonio Rd. MTV29-110
Palo Alto, CA 94303
+1 650 336 2620
Randall.Smith@Sun.COM

Antero Taivalsaari
Sun Microsystems Laboratories
901 San Antonio Rd. MTV29-117
Palo Alto, CA 94303
+1 650 336 1957
Antero.Taivalsaari@Sun.COM

ABSTRACT

We present a generalized definition of scrolling that unifies a wide range of existing interaction techniques, from conventional scrolling through pan and zoom systems and fish-eye views. Furthermore it suggests a useful class of new scrolling techniques in which objects do not move across the display. These "stationary scrolling" techniques do not exhibit either of two problems that plague spatial scrolling system: discontinuity in salience and the undermining of the user's spatial memory.

KEYWORDS: scrolling, window management, portable devices.

INTRODUCTION

The "screen real estate" problem affects users who wish to display more objects than can fit on the screen. The issue is getting more troublesome, as recent developments in computing have introduced devices with very small screens, such as cell phones, PDA's, and pagers. In many visions of our near future, more and more is demanded of such devices, but a user who manages even a moderately complex task through such a small display immediately confronts the screen real estate problem.

Scrolling is a conventional and useful way to deal with limited screen real estate, whether in a text editor or on a virtual desktop crowded with icons. When the user runs out of screen space, the user scrolls so that old objects get out of the way and new space becomes available. We will carefully define scrolling, in a rather mathematical way, as being a mechanism that changes the *salience* of a group of display objects while preserving their relative position. The salience of an object is some subjective function measuring the prominence of the object on the display, and its availability for user input. Thus a large object near the center of the user's field of view would normally be more salient than a small iconified document at the periphery. This definition, because it refers to changing salience as opposed to just display position, admits new kinds of scrolling mechanisms. That is, we generalize away from the screen position aspect of salience to allow a scrolling operation to move objects through a more abstract space of display attributes, including non-positional dimensions such as transparency and size.

Our approach is fairly simple: we think of some larger, more abstract space in which the objects are located. A scrolling offset, also some point in this space, is associated with the user. It is the relative difference between the object's location and the user's offset that produces an actual display coordinate for that object. So, to scroll through this more general space, the user varies the scrolling parameter and the appearance of each object changes: the object's absolute position in the space remains unchanged, but the display coordinate changes.

We will emphasize stationary scrolling, a useful special case of generalized scrolling in which the salience is decoupled from display position, so that the user scrolls only through non-positional display attributes. We argue that stationary scrolling solves two problems that plague conventional scrolling: discontinuous salience and the undermining of spatial memory.

Our framework suggests that multiple dimensions can be scrolled simultaneously, and we note in particular that stationary scrolling can be more usable if one simultaneously scrolls the drawing order dimension with any non-spatial dimensions.

To this method of displaying we add the notion of filtering out user input events as visual salience decreases. In spatial scrolling, when an object gets carried off screen, its accessi-

UIST '99. Asheville, NC

bility for user input also disappears. In our more generalized scrolling framework the analogous notion of removing access to an object when the visual salience drops below some critical threshold can also be useful. The variation of display salience and user input accessibility as an object is moved through an abstract space of display attributes constitutes our approach to generalizing scrolling.

GENERALIZED SCROLLING

Consider a set of displayable objects, $\mathbf{O} = \{O_i\}$ with each element O_i located in some abstract *N*-dimensional space, $\boldsymbol{S}$. The location of each object O_i in this space is some *N*-dimensional coordinate, $\boldsymbol{x_i}$. We call the vector $\boldsymbol{x_i}$ the *display location* in $\boldsymbol{S}$. The numbers in $\boldsymbol{x_i}$ determine how an object will appear on the screen, and therefore affect the salience of the object as perceived by the user. The components of $\boldsymbol{x_i}$ may represent quantities such as horizontal and vertical components of position on the screen, the object's size, its transparency, the saturation of the object's colors, the speed with which it blinks, the degree to which the object is in focus or blurred, and so forth. Note these elements affect the object's salience, but not its identity: moving an object through this space $\boldsymbol{S}$ will not substantially affect the user's perception of what the object is, merely how it looks. (For example, simply changing the position of some document icon does not change the users ability to identify it, whereas scrambling the colors, replacing the shape, and embedding an arbitrary bitmap in its surface may make identification difficult.)

To define scrolling, we associate with the user a *scrolling offset* parameter $\boldsymbol{p}$ in $\boldsymbol{S}$. In conventional text scrolling for example, $\boldsymbol{p}$ is a single number representing the vertical offset of the user's current view into the document: the value of $\boldsymbol{p}$ is determined by the position of the scroll bar. We now offer our definition of scrolling: set $\mathbf{O}$ is *scrollable* if each object O_i also has an *absolute location* $\boldsymbol{x_{Ai}}$ in $\boldsymbol{S}$ which is related to the display location $\boldsymbol{x_i}$ through the scrolling offset, $\boldsymbol{p}$, and a *scrolling function*, $\boldsymbol{f}$.

$$\boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{x}_{Ai} - \boldsymbol{p})$$

To reduce notational clutter we usually drop the subscript *i*, so we can simply say that each object has some display location $\boldsymbol{x}$, and some absolute location $\boldsymbol{x}_A$.

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}_A - \boldsymbol{p}) \qquad (1)$$

Both $\boldsymbol{f}$ and $\boldsymbol{p}$ are in general vectors so they may affect more than one aspect of the display location, and, in order to be useful in a conventional way, $\boldsymbol{f}$ usually takes on values associated with greatest salience at or near $\boldsymbol{f}(\boldsymbol{x} = 0)$. Several example functions in the next section illustrate the role of equation (1).

We supplement this definition with a less quantitatively articulated rule for the behavior of objects with respect to accepting user input. In general, scrolling systems reduce an object's availability for input as the visual aspects of salience are reduced. The following examples will make it clear that for some scrolling systems, this rule follows naturally from the geometry of the display scheme. For other systems it can be useful to employ a simple clipping policy: when an object drops below some threshold in visual salience, it no longer becomes available for user input.

Because $\boldsymbol{f}$ is in general a vector, an appropriate choice of $\boldsymbol{f}$ can mean that changes in $\boldsymbol{p}$ will cause changes in several components of $\boldsymbol{f}$ simultaneously. The effect will be that as the user scrolls, even if only through one dimension (e.g.: with a scroll bar), several display attributes of the screen objects change simultaneously. We find it can be useful to couple motion through the drawing order dimension with any other non-spatial scrollable dimensions. After all, if object A is partly occluded by object B, yet A is rendered in an otherwise more salient way, the system is sending a kind of mixed message. Furthermore, mixing in drawing order supports the association of high salience with availability for user input, as visual occlusion of an object normally correlates with blocking of input events to that object.

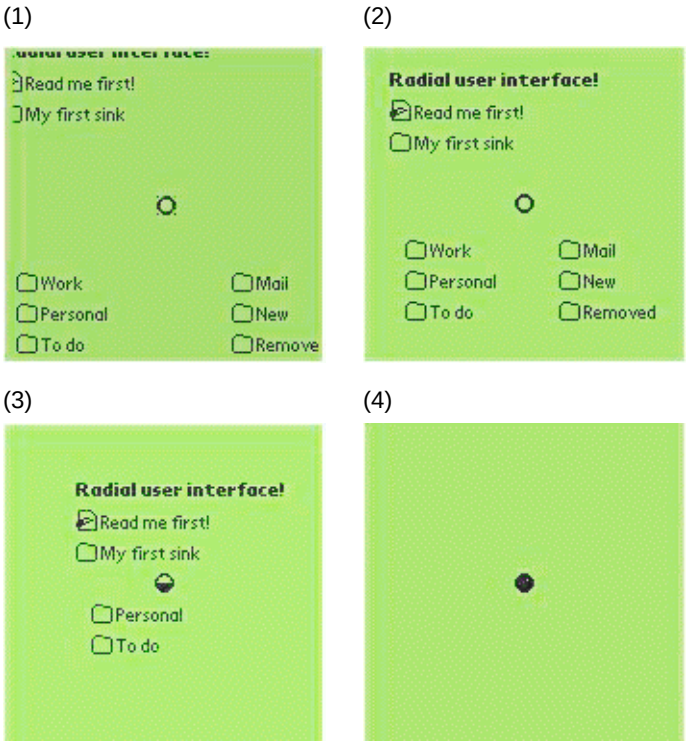
The relation summarized in equation (1), and the drop of input availability as a contribution to reduce salience comprise our approach to generalized scrolling.

EXAMPLES: VARIATIONS ON SPATIAL SCROLLING

A whole range of scrolling behaviors, both conventional and novel can be expressed through this mechanism. Here we look at several screen management systems, all of which move objects across the display in response to certain user inputs. Each can be described by the generalized scrolling framework.

Linear scrolling

As a simple example, consider the set $\mathbf{O}$ to be the set of characters in a text editor. The display location $\boldsymbol{x}$ is simply the screen position (x, y) of the character, and the parameter $\boldsymbol{p}$ is some vertical offset reading taken from a scroll bar, $\boldsymbol{p}$ = (0, p). The absolute location $\mathbf{x}_A = (x_A, y_A)$ is the "real" position of the character within the entire document: while the document is being scrolled, $\mathbf{x}_A$ remains fixed for each character, it is only the displayed location (x,y) that will change under scrolling.



$$(r, \theta) = \boldsymbol{f}(\boldsymbol{x}_A - \boldsymbol{p}) = (r_A, \theta_A) - (p, 0)$$

Figure 1 An implementation of radial scrolling on a hand-help PDA. As the user scrolls, each object moves towards or away from the central point on the display. An object with negative radius is simply not displayed, so an icon disappears upon reaching the center. Here, note the circle at the center at first displays itself as empty ((1) and (2)), then partially full (3), or completely full (4), depending on how many of the objects have been scrolled into the center (i.e.: mapped to negative radial coordinates).

For simple vertical scrolling within such a document, the function $\boldsymbol{f}(\mathbf{x_A} - \boldsymbol{p})$ is a two-dimensional quantity

$$(x, y) = \boldsymbol{f}(\boldsymbol{x}_A - \boldsymbol{p}) = (x_A, y_A) - (0, p)$$

so that an increase in the scroll bar position decreases the displayed y value of every object in the set **O**.

Radial Scrolling

A simple variation on linear scrolling can be achieved by representing the display space as radial coordinate system, with each point specified by a radius *r* and angle θ

$$(r, \theta) = \boldsymbol{f}(\boldsymbol{x}_A - \boldsymbol{p}) = (r_A, \theta_A) - (p, 0)$$

One of us (Taivalsaari) has implemented this on a hand-held PDA, as illustrated in Figure 1 [14]. In any such polar representation, the display system must interpret negative values of *r*: in this implementation, the display simply hides objects with negative *r*, so they apparently disappear as they pass into the central point of the display.

One of the drawbacks of linearly scrolling systems is the rather sudden drop to invisibility of an object moving off the edge of the screen. A more graceful drop in salience can be achieved by using "fisheye" scrolling.

Fisheye scrolling

In "fisheye" or "focus plus context" systems [5], [6], [7], [9] a set of objects can be scrolled ever closer to the edge of the screen without allowing them to leave. We limit our example here to fisheye effects on position only, though more commonly such systems also affect the object's size, shrinking them as they approach the periphery.

To achieve this kind of effect in the vertical dimension only, on a screen of height *h* one can choose $\boldsymbol{f}$ according to

$$(x, y) = \boldsymbol{f}(\boldsymbol{x}_A - \boldsymbol{p}) = \left(x_A, \frac{h}{\pi}\arctan[s(y_A - p)]\right)$$

where *s* is an arbitrary scaling factor. The result maps the object's absolute x-coordinate into some point in the display range (*-h/2*, *h/2*). Figure 2 illustrates this approach.

Pan and zoom scrolling

We show how our approach can describe pan and zoom user interfaces such as Pad++ [2], or any 3D world without user

rotations [4], [8], [13]. As a user moves through such a space, the display position and size of each object varies accordingly. For this case, the space ***S*** can be taken as two spatial dimensions, one size dimension, and an extra "helper" dimension, z, which influences both size and placement. (In practice of course, z feels like it is playing the role of a third dimension, directed into the screen.) The size factor multiplies the width and height of each object in order to give the appropriate perspective effect. In this case

$$\boldsymbol{x} = \begin{bmatrix} s \\ x \\ y \\ z \end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}_A - \boldsymbol{p}) = \boldsymbol{f}\left(\begin{bmatrix} s_A \\ x_A \\ y_A \\ z_A \end{bmatrix} - \begin{bmatrix} 0 \\ x_p \\ y_p \\ z_p \end{bmatrix} \right)$$

$$= c\frac{(\boldsymbol{x}_A - \boldsymbol{p}) - \hat{z}(\hat{z} \cdot (\boldsymbol{x}_A - \boldsymbol{p}))}{\hat{z} \cdot (\boldsymbol{x}_A - \boldsymbol{p})}$$

where $\hat{z}$ is the unit vector in the z direction, and c is a constant characterizing the width of the "viewing frustum." Note that the functional form including the extra non-positional component of size is a function of $(\boldsymbol{x}_A - \boldsymbol{p})$, and so fits in our framework. In practice, the function ***f*** for 3D systems is a bit more complex in that it includes supplemental clipping rules to suppress the rendering of objects behind the viewer and to avoid the troublesome singularity at $\hat{z} \cdot (\boldsymbol{x}_A - \boldsymbol{p}) = 0$. But such considerations are again functions of the quantity $(\boldsymbol{x}_A - \boldsymbol{p})$ (specifically, they depend on $\hat{z} \cdot (\boldsymbol{x}_A - \boldsymbol{p})$).

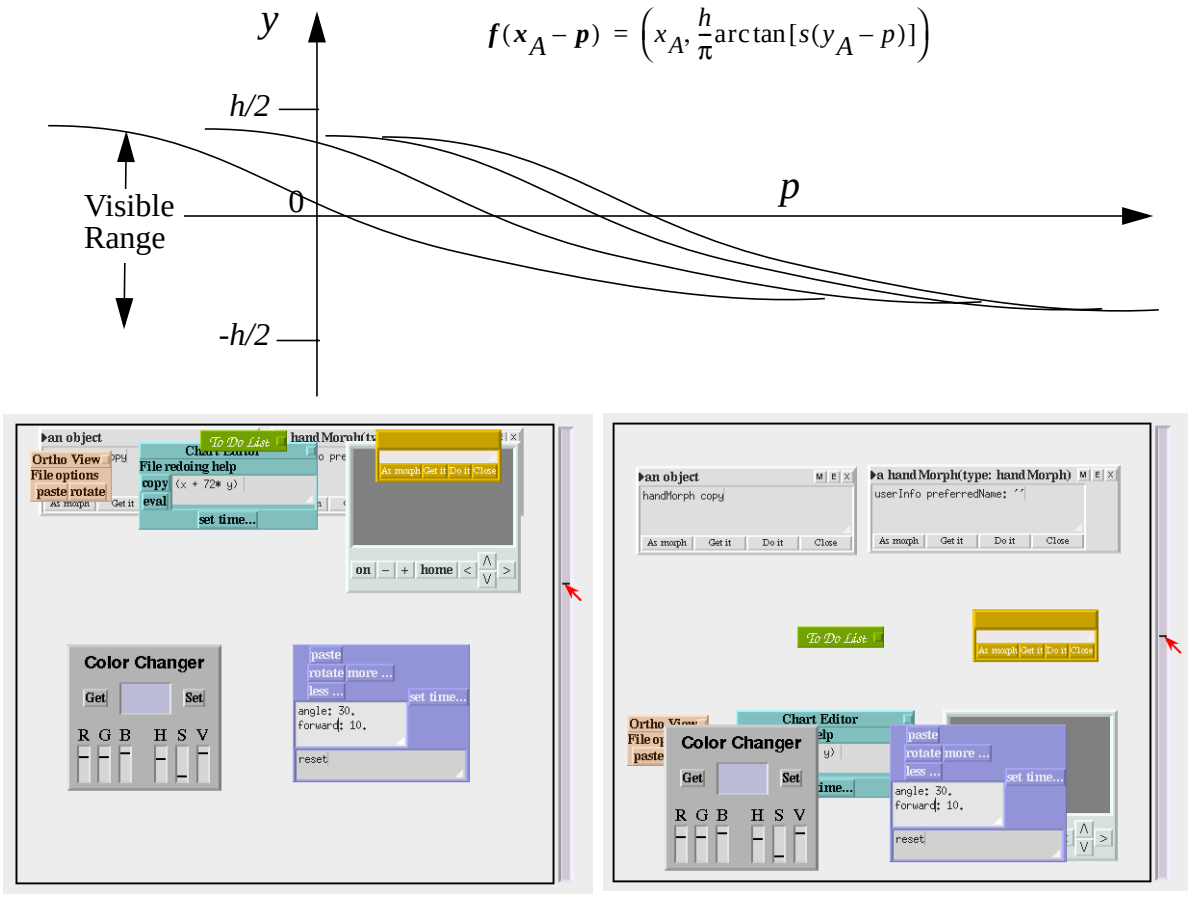


Figure 2. Using an arctan or other doubly asymptotic form as the scrolling function ***f***(z) achieves a kind of "fisheye" effect by mapping all objects into the visible portion of the display. The graph shows this function for four objects, each with a different absolute location $\boldsymbol{x}_A$. At the bottom are two views at different scrolling values ***p***. The objects tend to crowd near the top or bottom, except when ***p*** is nearly equal to $\boldsymbol{x}_A$, in which case they appear more vertically centered. (In this example we limit scrolling to the vertical dimension only).

PROBLEMS WITH SPATIAL SCROLLING

Discontinuity in salience

The smooth motion towards objects in a pan and zoom system gives an appealing kind of gradual revelation of detail. This is a direct benefit of smoothly changing salience. However, as one moves close to a group of objects, the objects begin to spread farther and farther apart across the screen. This may be fine at first, but as one continues to zoom, eventually some objects in the group may partly or entirely disappear off the periphery. Or, in 3D systems, if one continues further towards or even through an object, it will at some point be clipped away, and will not be displayed at all. That is, the salience can suddenly drop from near maximal values to zero.

Put in a more general way: two objects that are close together in absolute positions $\boldsymbol{x}_{A1}$ and $\boldsymbol{x}_{A2}$ should be close together in the display space $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Most systems that clip objects from a display region, such as conventional text scrolling, panning, or motion in 3D spaces, will violate this continuity rule. It is however possible to avoid clipping by crowding objects closer together in the periphery, as is done in the fisheye class of techniques. This crowding actually changes relative positions of the objects somewhat, and this exacerbates a second problem with any technique that moves objects across the screen, namely, undermining the user's spatial memory.

Undermining spatial memory

An approach such as the desktop metaphor yields benefits because users can arrange display objects as they wish. The act of placement creates an association in the user's memory between the object and that location. But in spatial scrolling systems, although ordinal positioning is maintained as one pans across a field of objects (e.g.: object A is always to the left of B) users can still easily lose track of their own location in the larger space, and the association between object and location is weakened. Undermining the user's spatial memory will be an issue in any system in which the user can place an object at one screen coordinate, yet see it rendered at another. Neither this problem nor the discontinuity in salience need affect stationary scrolling.

STATIONARY SCROLLING

A number of authors have developed the idea of holding a set of objects in place on the screen while some of them are selected for highlighting based on a value associated with each object. This technique is sometimes called "painting" or "brushing" of scatter plot data [1], [10]. The work of McDonald, Stuetzle and Buja [3], [11] provides multiple views through which one can change the range of highlighted values, and generalizes to display objects other than simple data points. Our emphasis though is for interactive displays in which input and drawing order is part of the story and we think it important to vary the salience in a smooth way with a value readily changed by the user (i.e: the scroll parameter from a scroll bar). In the case of "fading," which we discuss as an example below, the work of Olsen et. al. [12] seems conceptually similar and fits as an example within our framework, although that work addresses the problem of how to point at objects by investigating novel ways to highlight them.

For simple stationary scrolling, we assign some non-spatial coordinate, *t*, to each object. The quantity *t* may represent opacity, blur, size, fadedness, drawing order, skew, color saturation, blinking rate, or any such attribute other than display location. When $t = 1$, we have normal fully present rendering, the value with maximal salience. As *t* approaches zero, the object fades away. The scrolling parameter $\boldsymbol{p}$ is also given a *t* component: in the simple case this p_t is the only scrollable dimension. In Figure 3 we illustrate scroll-by-fading of stationary objects, having taken a particular form for $\boldsymbol{f}$, namely

$$\begin{bmatrix} t \\ x \\ y \end{bmatrix} = \boldsymbol{f}(x_A - p) = \begin{bmatrix} \dfrac{\lambda^2}{\lambda^2 + (p_t - t_A)^2} \\ x_A \\ y_A \end{bmatrix} \qquad (2)$$

$$= \frac{\lambda^2 \hat{\boldsymbol{t}}}{\lambda^2 + (\hat{\boldsymbol{t}} \cdot (\boldsymbol{x}_A - \boldsymbol{p}))^2} + ((\boldsymbol{x}_A - \boldsymbol{p}) - \hat{\boldsymbol{t}}(\hat{\boldsymbol{t}} \cdot (\boldsymbol{x}_A - \boldsymbol{p}))$$

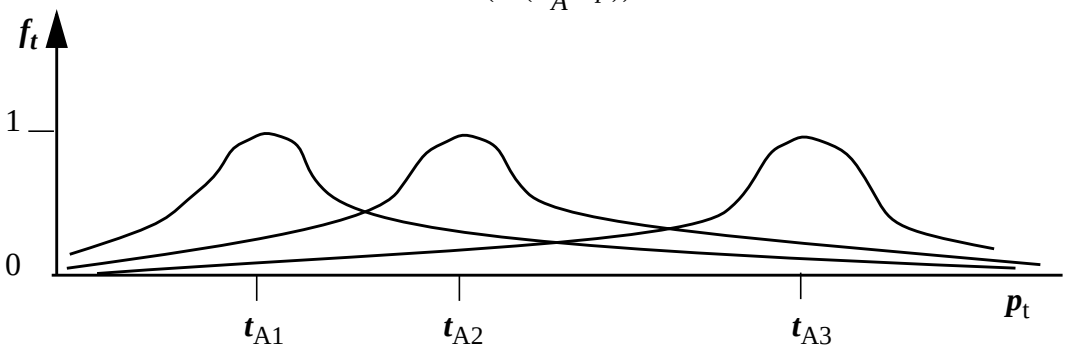
where the second form emphasizes that the function depends purely on the quantity $\boldsymbol{x}_A - \boldsymbol{p}$, introducing the unit vector in the non-spatial direction, $\hat{\boldsymbol{t}}$. The quantity λ determines the "width" of the curve. Many different forms for $\boldsymbol{f}$ are possible of course, the above example is relatively efficient to compute. Instead of fadedness, any non-positional attribute can be used. Figure 4 illustrates the effect of taking the $\boldsymbol{t}$ parameter to be interpreted as the size of the object.

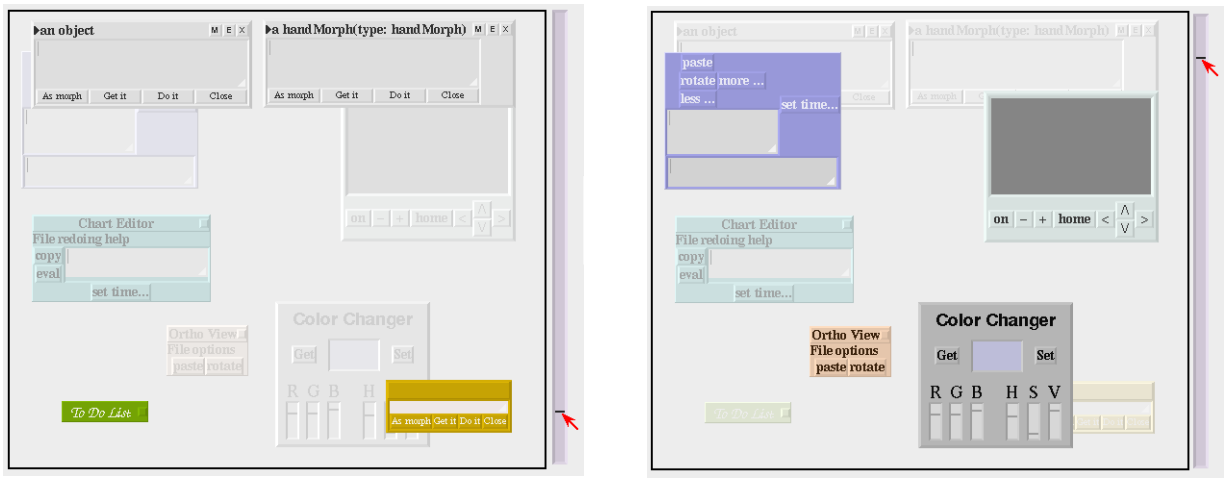
Multi-dimensional stationary scrolling

It is natural to extend the non-spatial coordinate to multiple dimensions, for example fadedness *and* size. The user's scrolling parameter $\boldsymbol{p}$ might then be read from a (possibly virtual) joystick, and both size and fadedness vary as the users moves about. Consider an *N*-dimensional space whose characteristic length in each dimension is *L*: the number of regions of radius λ that fit in the space is roughly $(L/\lambda)^N$. So if one dimension is sufficient to distinguish among up to 10 different object groups scattered along the scrolling dimension, then as a rule of thumb, two dimensions will provide for 100 different groupings, three would provide for

Figure 3. Fading the colors of objects to decrease salience is one way to scroll without changing position. The t component of the function f is shown plotted for three different objects, each with its own absolute coordinate t_A. At bottom, two screen snapshots are shown one for each of two different scrolling parameters, p. (In this case we are simultaneously scrolling in the drawing order dimension, so that the more faded objects are drawn behind less faded objects.)

$$f(x_A - p) = \frac{\lambda^2 \hat{t}}{\lambda^2 + (\hat{t} \cdot (x_A - p))^2} + ((x_A - p) - \hat{t}(\hat{t} \cdot (x_A - p)))$$





1000, and so on. Although there can be lots of space, we have found it can become somewhat frustrating to wander through multiple non-spatial dimensions in order to navigate to a spot where some particular object becomes salient.

This problem is not an issue when the user's scrolling parameter is restricted to a single dimension that moves diagonally through the multiple non-spatial coordinates. The user's scrolling parameter p would have a one-dimensional range of values affecting several non-spatial coordinates simultaneously. As an example, taking the non-spatial coordinates to be fadedness and drawing order, the user's scrolling parameter would cause an object to become more and more faded while simultaneously causing it to be drawn under more and more objects. This kind of "coupling" of the drawing order attribute with other non-spatial attributes is in fact very useful. Figure 3 illustrates scrolling of both fadedness and drawing order.

To show how this can be treated in the mathematical framework, equation (2) can be extended so that the space has two non-spatial components, one for drawing order, and one for fadedness. The components of the each object's display location x includes then t_D and t_F to correspond to these display attributes. When these parameters are near 1, they are associated with maximum salience (drawn in front, least faded). Each object has fadedness and drawing order absolute locations: t_{FA} and t_{DA}. For simplicity, we will give the

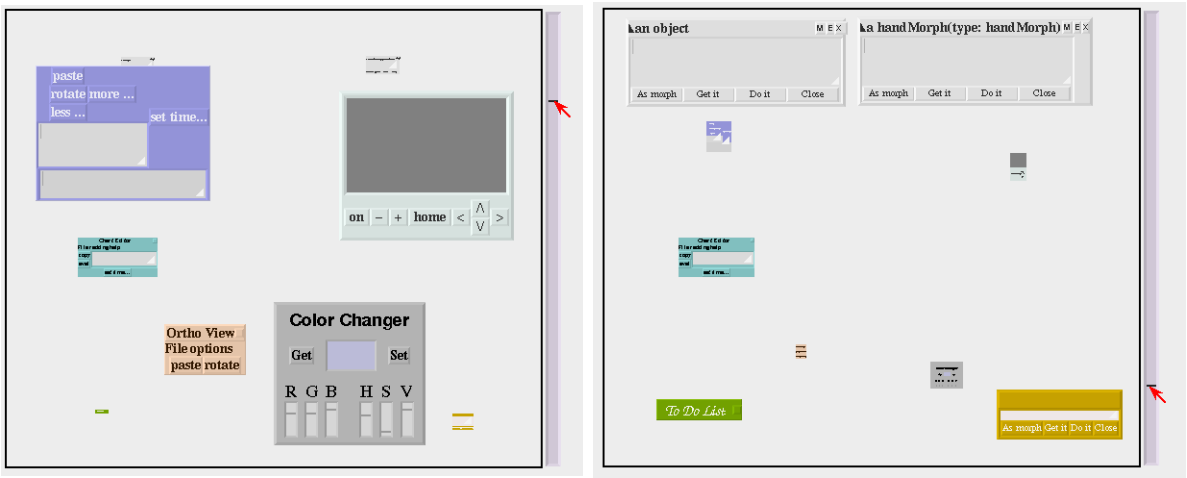


Figure 4. Stationary scrolling in which the scroll operation affects size.

vector ***p*** the same scalar projection *p* on the drawing order and fadedness dimensions, though in general they could be different but linearly proportional to each other. The equation becomes

$$\begin{bmatrix} t_D \\ t_F \\ x \\ y \end{bmatrix} = \boldsymbol{f}(\boldsymbol{x_A} - \boldsymbol{p}) = \begin{bmatrix} \dfrac{\lambda^2}{\lambda^2 + (p - t_{DA})^2} \\ \dfrac{\lambda^2}{\lambda^2 + (p - t_{FA})^2} \\ x_A \\ y_A \end{bmatrix}$$

where we have assumed the same curve widths λ for both dimensions, though in general of course they may differ.

NON-SPATIAL PLACEMENT: CHANGING AN OBJECT'S ABSOLUTE COORDINATE

In our model, each object has an absolute coordinate $\boldsymbol{x}_A$ and from time to time the user may wish to change this, for example, to regroup objects. In traditional spatial scrolling, a display object can either be dragged from one place to another, or, as with text, the user can cut the objects from one location to paste them into another. However, in stationary scrolling, the abstract scrolling dimension is intrinsically non-spatial, so changes to absolute coordinates perhaps feel less tangible.

The task of assigning a new value to the parameter $\boldsymbol{x}_A$ associated with some object can be realized in any number of ways. We mention one here: because an object is not spatially changed in stationary scrolling, a single gesture is sufficient to change an object's absolute coordinate. For example, a pop-up menu selection called "jump to here" can be provided for each object. Selecting "jump to here" causes the object to set its absolute coordinate $\boldsymbol{x}_A$ to the value of the user's current scrolling parameter, ***p***. In contrast, a direct manipulation spatial scrolling system will typically require the user to denote two locations (initial and final value).

MULTIPLE SCROLLING SETS

Interesting effects can be achieved by having the same display space house multiple sets of objects, each set with its own scrolling behavior slaved to the same user scroll parameter ***p***. Besides the single peak function we discuss above, two others are of particular utility, a constant function and multiply peaked functions.

Objects that participate in a set whose scrolling function is constant appear not to scroll at all, as though "riding along" with the user as the user changes ***p***. This "stick with the user" effect has been employed in some spatially scrolled window systems. Objects can appear multiply salient by creating a scrolling function with multiple peaks. A peak that does not reach a full maximum might represent a region where an object is only partially relevant.

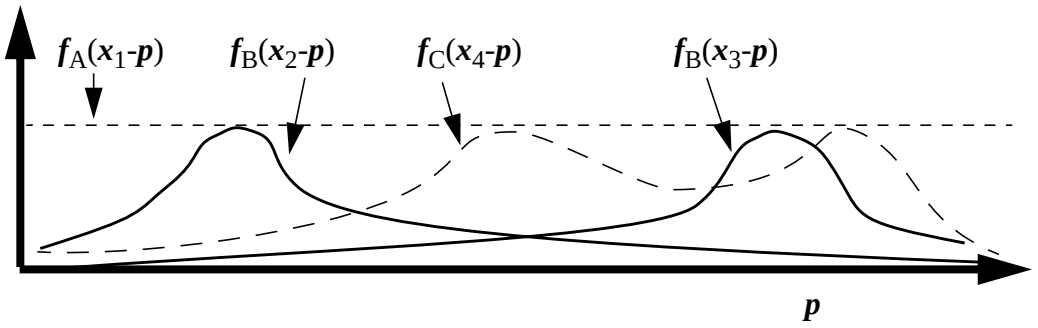


Figure 5. Different scrolling functions for different groups of objects can be useful. Object 1, associated with f_A,a constant, always appears fully salient, regardless of the scrolling parameter. Objects 2 and 3 use f_B, a function with a single-maximum. The function f_C is used by object 4, which appears with maximal salience at two locations as the user scrolls. A function with a peak at less than full value might be used to express the notion of an object partially relevant to that location in the space.

CONCLUSIONS

The recent advent of powerful devices with small displays has renewed the interest in finding new screen real estate management techniques. Our generalized definition of scrolling is centered on the notion of salience, and describes scrolling as a function whose domain is a relative offset between a scrolling parameter, and an object's intrinsic position in an abstract space of display attributes. Our approach unifies many different existing systems, such as conventional scrolling, motion through 3D space, pan and zoom systems, and fisheye views. Furthermore it suggests a useful class of new scrolling techniques: stationary scrolling, in which objects do not move across the display.

We consider an object's availability for user input to be part of what is managed in scrolling systems. Most existing systems automatically reduce input availability as a side effect of reducing screen real estate. However, stationary scrolling systems may not reduce screen real estate, so an explicit policy can be introduced to achieve this effect.

Most spatial screen management systems are based on metaphors which offer the user a chance to reuse existing intuitions. However, operating within a metaphor may of course not be the optimal way to perform a particular task. Specifically, scrolling systems in which objects move across or even off screen can suffer from either or both of two problems: discontinuity in salience, and undermining the user's spatial memory. By abstracting out the essence of scrolling systems in a general way, we hope we have offered new ways to address an old problem.

Acknowledgment

Our thanks to George Furnas for enthusiasm and useful comments.

REFERENCES

[1] Becker, R.A., Cleveland, W.S., Dynamic graphics for data analysis. *Statistical Science* vol 2, pp.355-395.

[2] Bederson, B.B., Hollan, J.D., Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology UIST'94* (Marina del Rey, California, November 2-4), 1994, pp.17-18.

[3] Buja, A., McDonald, J.A., Stuetzle, W., Viewing High Dimensional Data by Painting Multiple Views. In *Proceedings of the Visualization'91 Conference*, 1991.

[4] Card, S.K., Robertson, G.G., York, W., The WebBook and the Web Forager: an information workspace for the World-Wide Web. In *Proceedings of the CHI'96 Conference* (Vancouver, B.C., Canada, April 13-18), 1996, pp.111-117.

[5] Furnas, G., The fisheye view: A new look at structured files. Bell Labs, Technical Memorandum 82-11221-22, 1982.

[6] Furnas, G.W., Generalized fisheye views. In *Proceedings of the CHI'86 Conference*, 1986, pp.16-23.

[7] Lamping, J., Rao, R., Laying out and visualizing large trees using a hyperbolic space. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology UIST'94* (Marina del Rey, California, November 2-4), 1994, pp.13-14.

[8] Lucas, P., Schneider, L., Workscape: A scriptable document management environment. In *CHI'94 Conference Companion*, ACM Press, 1994, pp.9-10.

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

[10] McDonald, J.A., Interactive graphics for data analysis. Ph.D. thesis, Department of Statistics, Stanford University, Technical Report Orion 11, 1982.

[11] McDonald, J.A., Stuetzle, W., Buja, A., Painting multiple views of complex objects. In *OOPSLA/ECOOP'90 Conference Proceedings* (Ottawa, Canada, October 21-25), ACM SIGPLAN Notices vol 25, nr 10 (Oct) 1990, pp.245-257.

[12] Olsen, D.R., Boyarski, JD., Verratti, T., Phelps, M., Moffet, J.L., Edson, L.L, Generalized Pointing: Enabling Multiagent Interaction, In *CHI'98 Conference Proceedings*, ACM Press, 1991, pp.526-533

[13] Robertson, G.G., Czerwinski, M., Larson, K., Robbins, D.C., Thiel, D., van Dantzich, M., Data Mountain: using spatial memory for document management. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology* UIST'98 (San Francisco, California, November 1-4), 1998, pp.153-162.

[14] Taivalsaari, A., The event horizon user interface model for small devices. Sun Microsystems Laboratories Technical Report SMLI-TR-99-74, March 1999.