VISUALIZING OFF-SCREEN LOCATIONS ON SMALL MOBILE DISPLAYS

SEAN GUSTAFSON

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada

ABSTRACT

Mobile devices, such as smartphones and other personal devices, are increasingly used to view maps and other large datasets. Their necessarily small displays can only show a small portion of the data at one time. Researchers have developed various visual techniques that overlay icons or shapes onto the edge of the display to provide the user with hints regarding the existence and location of undisplayed points of interest. However, current techniques fail in practice on mobile devices because they are confusing, do not scale or take up too much valuable screen space.

In this thesis, I describe a new technique to visualize the location of off-screen points of interest. This technique, called Wedge, addresses specific shortcomings of existing techniques. This thesis details the design and implementation of Wedge and summarizes the results of a thorough experimental evaluation. Furthermore, I present a preliminary model of user performance that I use to highlight design suggestions for practitioners using Wedge.

PUBLICATIONS

Some ideas and figures in this thesis have appeared previously in the following publications by the author:

Sean Gustafson and Pourang Irani. Comparing visualizations for tracking off-screen moving targets. In *Extended Abstracts on Human Factors in Computing Systems (CHI '07)*, pages 2399–2404, 2007.

Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. Wedge: Clutter-free visualization of off-screen locations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 787–796, 2008.

The following figures have been reproduced from other sources with permission:

Figure 2a on page 10 from [7] © 2006 ACM, Inc.

Figure 2b on page 10 from [58] © 2003 authors.

Figure 3b on page 11 from [12] © 2004 ACM, Inc.

Figure 4 on page 12 from [31] © 2006 ACM, Inc.

Figure 7 on page 16 from [39] © 1994 ACM, Inc.

Figure 8 on page 18 from [33] © 2006 ACM, Inc.

The greatest enterprise of the mind has always been and always will be the attempted linkages of the sciences and the humanities.

— E.O. Wilson, *Consilience* [56]

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Pourang Irani for his incredible personal and professional support. He is a great advisor and mentor. Thank you Pourang.

I thank my family for their support and confidence in me. From my father I acquired a great love for knowledge and technology while my mother supplied a little of her wonderful patience. I couldn't have done this without either of you.

Thanks to my sister Jennifer and her beautiful new family. And to the rest of my family: thank you for all the love, support, and delicious food.

Thank you to my friends. You are all amazing – never once doubting that I should be doing this or that I was capable. You all provided an outlet for frustrations, an ear for my half-brained ideas, tried out my experiments and read my work. Thank you.

Thank you to my collaborators: Dr. Pourang Irani, Dr. Carl Gutwin, Dr. Patrick Baudisch, Dr. Sriram Subramanian, Kang Shi, Bowen Hui, and Mahfuz Rahman. I learned so much working with you.

Thank you to my committee members, Dr. Parimala Thulasiraman and Dr. Todd Mondor, for your time, helpful comments, and support. Also thank you to Dr. Rasit Eskicioglu for chairing my thesis defense.

Thank you to the support staff here at the University of Manitoba, in particular Jeff Durston for his technical support with my experiments. And thank you to Colin Oullette and David McDine for administering pilot studies for me at the University of Saskatchewan.

Thank you to Dr. Irani, the Government of Manitoba, Faculty of Graduate Studies, Faculty of Science, Graduate Students' Association, Alumni Association and the Department of Computer Science for their generous financial support throughout my degree.

And finally, thank you to my fellow students in the HCI lab, in particular Grant Partridge for his ideas and editing prowess, but also to Erum Tanvir, Mahtab Nezhadasl, Kang Shi, David McCallum, Ed Mak, Fouad Alallah, Nivedita Kadaba, Jon Cullen, and Bowen Hui for your support, ideas, and help.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ANOVA	Analysis of Variance
BRETAM	Breakthrough, Replication, Empiricism, Theory, Automation, Maturity
CTAN	Comprehensive TEX Archive Network
GIS	Geographical Information System
GPS	Global Positioning Satellite
HCI	Human-Computer Interaction
IDE	Integrated Development Environment
PDA	Personal Digital Assistant
SDAZ	Speed Dependant Automatic Zooming
ZUI	Zoomable User Interface

1

INTRODUCTION

In many computer-based tasks (such as map viewing, web browsing or computer-aided design) there is much more data to display on the screen than can fit. New generations of mobile devices, such as smartphones and Personal Digital Assistants (PDAs), especially Apple's *iPhone*, now have the processing power and connectivity to provide functionality previously only available on desktop and laptop computers. However, mobile devices will always be limited by their display sizes. When a dataset, such as a map, does not fit all on the screen at one time, an application must provide a mechanism to choose what to display, to navigate to the undisplayed regions, and possibly, to inform the user of what is not displayed.

There are many strategies for displaying and navigating datasets that are larger than the display. My work focuses on *off-screen pointing* techniques that consist of graphical elements overlaid upon the screen, where each element represents a specific point of interest that is not currently shown. The off-screen pointer informs the user of a location or point of interest in the periphery of the current view. Knowing the location of an off-screen point of interest is obviously valuable if the user wants to navigate to that point, but it is also helpful to provide the user with a more complete mental model of the surroundings that is used in complex tasks such a route planning.

Figure 1 shows three common techniques for providing the user with information regarding nearby points of interest. The first of these, overview+detail (Figure 1a), is one of the most commonly used techniques. The other two techniques, simple arrows (Figure 1b) and Halo (Figure 1c) are occasionally used and belong to the off-screen pointing class of techniques.

I have identified a number of problems with the existing techniques for visualizing undisplayed data and I designed a new technique called *Wedge* that improves on how well the user can determine the

precise location of an off-screen point of interest, while avoiding the clutter that plagues other techniques. In addition, I present a preliminary model to describe user performance with Wedge and related techniques. Using this model I was able to provide specific design suggestions aimed at implementers of Wedge and similar techniques.

The rest of this thesis is structured as follows. First, in Chapter 2, I discuss the work related to displaying and navigating large datasets on small devices; I then present, in Chapter 3, a detailed problem statement that enumerates the many problems with existing techniques. Followed, in Chapter 4, with a thorough description of the Wedge technique and its evaluation in a lab setting. In Chapter 5, I present a preliminary formal specification of Wedge performance followed by, in Chapter 6, a summary of the thesis and some directions for future work.

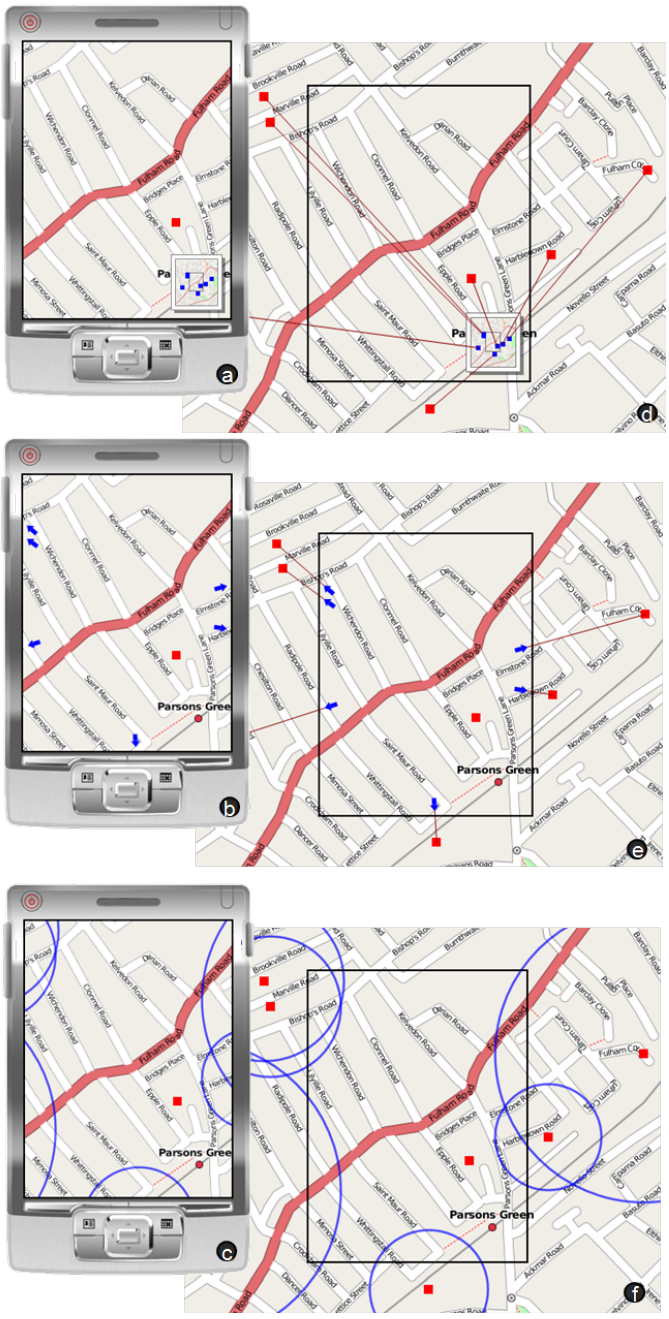


Figure 1: Three existing techniques for visualizing large datasets on a small device: a) overview+detail; b) arrows; c) Halo. Beneath (d-f) are larger sections of the map showing the off-screen locations with the respective visual techniques.

2

RELATED WORK

Researchers have studied the problem of how best to represent and interact with a dataset that is larger than the display for many years. The obvious solution is to show only part of the data and then provide a mechanism for navigating to an undisplayed region. These traditional methods of navigation and interaction are the first set of related work that I will survey. I will then describe more throughly the concept of an off-screen pointer, along with related work from perceptual psychology that forms the theoretical base of some of the pointing techniques. Following this, I will survey distortion-based techniques that affect the presentation of the data to fit more on the screen at one time, lessening the need for navigation and off-screen pointers.

2.1 TRADITIONAL TECHNIQUES

Traditionally, navigation through a large dataset is accomplished with a combination of *panning*, *zooming* and *multiple windows*.

Panning/scrolling (the term panning is generally used for maps, scrolling for documents) is the simple operation of moving the viewing window to display information not currently on the screen as is done in all current word processors and web browsers.

Zooming is another simple technique that allows the user to adjust the current level of detail for viewing. The ZoneZoom [48] smartphone-based mapping application is an example of a zoom-only interface. The application divides a map into nine areas, each of which can be zoomed into by pressing on one of nine keys on a phone's keypad. The zoom-only interface matches well with this application and the limited input capabilities of a mobile phone.

In practice, pan and zoom are usually used together because they perform complementary functions. Bederson and his colleagues

developed Pad++ [3] – and later, Jazz and Piccolo [4] – to investigate the concept of a Zoomable User Interface (ZUI), now often referred to as a *multi-scale interface*. In a multi-scale world the user may pan in any direction and may zoom in and out freely. Multi-scale interfaces are widely used today for map viewers, such as *Google Maps* [21], in various experimental commercial projects including *Microsoft Live Labs' SeaDragon* [36], and extensively in games.

With Pad++, Bederson and Hollan also introduced the idea of *semantic zooming* where different information could be presented at each zoom level. Semantic zooming allows a computer system designer to provide more information about an object as a user zooms in on it. Semantic zoom was employed in the Summary Thumbnails [37] PDA web browser. Instead of compressing a full web page to fit it into the display of a PDA (which would make it unreadable), Summary Thumbnails maintains the layout of the page and only displays extracted keywords from each paragraph. The user can zoom in on a section of a page to read the full text.

A new class of techniques has emerged in the last several years that combines panning and zooming into one seamless operation. Speed Dependant Automatic Zooming (SDAZ) is a technique proposed for conventional-sized displays by Igarashi and Hinkley [30] where the zoom-level automatically decreased as the panning speed increased. The effect results in a fluid zooming out when the user chooses to move the current focal point. Because the user is not generally inspecting, in detail, the content while panning, the system can automatically zooms out to provide better context information to the user that aids their navigation task. As soon as the user stops panning, the zoom level is returned to normal. Cockburn and Savage [14] performed a full evaluation of the technique and corroborated the original informal evaluation results, showing that SDAZ outperforms simple scrolling and pan/zoom interfaces.

Pan and zoom-based techniques provide *temporally separated* views of the data. Other techniques utilize *spatial separation* of the data views by presenting the users with multiple windows at the same time, often overlaid. The most common multiple-windows technique involves one large detail window and a small overlaid window that

shows the location of the detail window in the larger problem space. This technique is called *overview+detail* [46] and is very common in map viewers and games. Multiple window techniques that do not use the overview+detail model are possible, but rarely used outside of specialized visualization software (like GeoZui4D [1]). One notable exception is DragMag [55] and similar lens-based techniques [45] which invert the overview+detail model, displaying the detail view in a small magnifying window that can be dragged around the large full-sized overview.

Nekrasovski et al. [44] compared pan-and-zoom interfaces with and without overviews. Although they found no difference in task completion time, users preferred the interface with the overview. Büring et al. [9] found a different result: in their study, users did not have a significant preference for either interface and users performed tasks faster *without* the overview. The Büring study was done on a small display, where the overview occupied a larger portion of the screen than that used by Nekrasovski, suggesting that, for small displays, a large detailed view can outweigh the benefits of an overview window. Chittaro [11] also notes this problem, hypothesizing that limited size and detail of the overview window makes it difficult to relate the two displays.

Recently Burigat, Chittaro and Parlato [8] followed up on the issue of using overviews on small mobile displays. They compared a classic zoom and pan interface with no overview, with a traditional overview and with a completely transparent wire-frame overview window. They found that the overview was beneficial by both improving task search times and by increasing spatial recall of the content explored. This finding directly contradicts that of Büring et al. [9], suggesting that those results are not generalizable and that particular care must be paid to ensure overviews are implemented in a beneficial manner.

One popular application which quite effectively uses a combination of panning, zooming, and multiple views is Google Maps [21]. Google Maps is an online mapping application that provides maps and satellite images for most of the world. The user can click and drag on the screen at any point to pan the display in any direction. Panning is also possible by dragging in the overview window located

in the bottom-right corner. In the case of Google Maps the overview window does not show an overview of the entire dataset as that would be much too large (i.e., the entire world), instead when the user changes the zoom level, the overview window's zoom level is also changed to keep it representing roughly three times the area represented by the detail window.

Google Maps also employs semantic zooming. Views at different zoom levels show different content. For example, when the user is at a country-wide zoom level only the major roads and cities are shown but when the user zooms in, secondary roads and smaller towns are shown.

The major problem with traditional navigation and presentation techniques is that they do not provide sufficient *context* to the user. There is either no context at all (as with panning and zooming) or the context is relegated to a separate window. In either case users can get lost navigating the data set – eloquently described as *desert fog* by Jul and Furnas [35].

2.2 OFF-SCREEN POINTING TECHNIQUES

Traditional navigation techniques can be supplemented by *off-screen pointers* – icons or shapes placed at the edge of screen to signify the existence of a point of interest, along with the direction and possibly the distance to the point. The canonical off-screen pointer is an arrow that points off-screen in the direction of a particular off-screen point of interest; however, there are many other techniques that provide the same function.

In order of importance, the fundamental spatial information provided by off-screen pointing techniques are: *existence* (whether or not a point of interest is available on the map); *direction* (which way the user must pan to locate the point of interest), and *distance* (the Euclidean distance from the current focal point to the point of interest). Existing techniques differ with respect to the spatial information that they provide and can be categorized as 0D, 1D, 2D, or 3D, which refers to the number of spatial dimensions employed

		Direction	*Distance*
0D	Icons and Lists	No	No
1D	Arrows	Yes	No
2D	Scaled and Stretched Arrows	Yes	Partial, arbitrary mapping of distance to size
	City Lights	Yes, indirectly	Partial, arbitrary mapping of distance to colour
	EdgeRadar	Yes, indirectly	Partial, arbitrary mapping of distance to position
	Halo	Yes, via position	Yes, via position
3D	3D Arrows	Yes, 2D	Partial, arbitrary mapping of distance to length

Table 1: Spatial dimensions supported by off-screen pointing techniques.

by the pointer. Table 1 lists several off-screen pointing techniques and the functionality provided by each.

Perhaps the most important information that an off-screen pointer can provide to the user is the confirmation (or denial) of the existence of a nearby point of interest. A traveler searching for a subway station is in a much better position if her cell phone map indicates that one is nearby, rather than if she did not have that knowledge.

Each of the techniques discussed in the rest of this section provide an explicit indication of the existence of a point of interest while providing some information regarding its location. Techniques that provide no information about location are really not pointers at all, but they can be considered in the same class as the other off-screen pointing techniques because they can provide much of the same functionality.

A pointer that indicates the direction to its corresponding point of interest is very helpful to users of some systems, particularly maps. A user driving in a certain direction is only interested in restaurants located along his planned route, not those in the opposite direction.

The basic example of a directional icon, a 1D pointer, is a simple arrow.

Arrow indicators (Figure 1a) are used extensively in Global Positioning Satellite (GPS) navigation systems to provide step-by-step directions from a starting point to a destination. Arrows are also common off-screen pointers in games. They are especially prevalent in overhead-view team sports games. Early arcade football games like *Tecmo Bowl* (released in 1987) and current *EA Sports* titles alike use simple arrows on the edge of the screen to signify the location of team members in the backfield or defensive zone.

The off-screen pointing technique City Lights [58], shown in Figure 2b, also provides an indication of direction, but does so without a pointing icon. Named after the glow that a city produces on the horizon when approaching from a distance, City Lights indicates an off-screen point of interest by drawing a thin coloured rectangle at the edge of the screen orthogonal to a point of interest. When the point of interest is not orthogonally located (i. e., the target is beyond the corner of the viewport), a convention must be used to properly indicate its direction from the corner. City Lights simply places an arrow-like triangle in the corner of the display to represent any number of off-screen objects in that corner. The corner problem is pervasive; most off-screen pointing techniques suffer from it to some degree.

An off-screen pointer that combines distance and direction to a point of interest (i. e., a 2D pointer) provides more information about the location of the point of interest than the techniques described above. Burigat et al. [7] used scaled or stretched arrows that mapped distance to the size or length of the arrow (Figure 2a). Unfortunately, these techniques do not provide enough information to accurately pinpoint the location of an off-screen point of interest. However, the simplicity of this type of technique could outweigh its inability to provide exact position information.

A related technique, EdgeRadar [25] (Figure 2c), provides the same function as an overview+detail configuration, by displaying off-screen points of interest as *blips* in a compressed view of off-screen space. But instead of the conventional inset windows, EdgeRadar

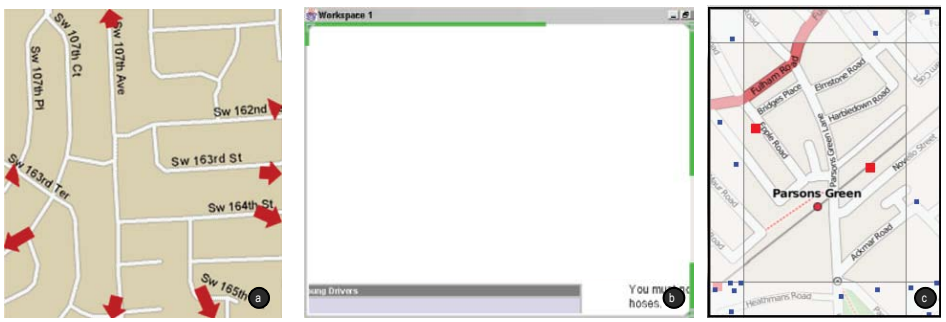


Figure 2: Two dimensional off-screen pointing techniques: a) Scaled arrows (Image from [7] © 2006 ACM, Inc. Reprinted by permission.); b) City Lights (Image from [58] © 2003 authors. Reprinted by permission.); c) EdgeRadar

splits up the off-screen space into four elongated regions and places them directly at the edge of the screen. The end result is a display very similar to City Lights that also provides a measure of distance from the screen edge. But, again, just like Burigat's arrows, EdgeRadar does not provide enough distance information to properly position the target in off-screen space.

A technique that does provide absolute positioning information is the Halo [2] off-screen pointing technique (Figure 1c), and as such can be considered truly two dimensional. A halo is a circle drawn around a point of interest such that a small arc intrudes onto the edge of the display. From this arc users can imagine the full shape using their perceptual ability to reproduce a complete object from only a small part. This ability is known as *amodal perception* which I will discuss in more detail in Section 2.2.2.

Halo has been used commercially in the *Second Life* virtual world and in various systems described in Human-Computer Interaction (HCI) literature. For example, Nacenta and colleagues [42, 43] have used Halo to indicate the location of off-screen cursors in a multi-monitor system. Halo has also used as a target indicator in a study comparing pan and zoom to rubber sheet navigation [44].

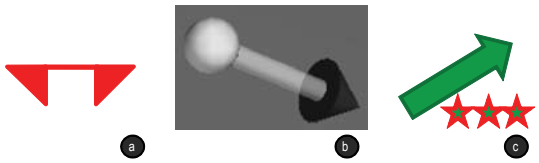


Figure 3: Three dimensional off-screen pointing techniques: a) *FreeSpace*-like pointer; b) 3D arrow (Image from [12] © 2004 ACM, Inc. Reprinted by permission.); c) simple arrow augmented with extra glyphs.

2.2.1 *Beyond two dimensions*

Driving-based video games and many GPS navigation systems use arrows extensively to provide directions for the user. Although these systems have a strong three dimensional component, the third dimension (i. e., height/zenith) is unneeded for navigation. Because of this a conventional 1 or 2D off-screen pointer may suffice for these applications.

Three dimensional virtual reality systems without a ground-based bias (e. g., flight simulators, non-spatial data exploration, etc.) could fully exploit an off-screen pointing technique that indicates the location of a target in three dimensions. Many systems use arrow-like indicators to point in the direction the user must rotate to have the target in view and may also include an indication of distance as well, as in the space simulator game series *FreeSpace*. The FreeSpace pointer uses a simple equilateral triangle to point to nearby off-screen targets but for distant objects the triangle splits into two triangles of half the original size, connected by a single straight line segment (see Figure 3a). The distance between the two triangles indicates how far away the target is.

To be fully 3D, however, the cue must show distance to the target and direction in two dimensions. Chittaro and Burigat developed the 3D arrow off-screen pointer, Figure 3b, and evaluated its use for navigation in 3D virtual environments [6, 12]. They compared user's performance navigating two virtual environments: a realistic environment where the user *walked* and in an abstract environment where the users *flew* around in three dimensions. The users were tested using the 3D arrow, a 2D arrow, an overview+detail-like rep-

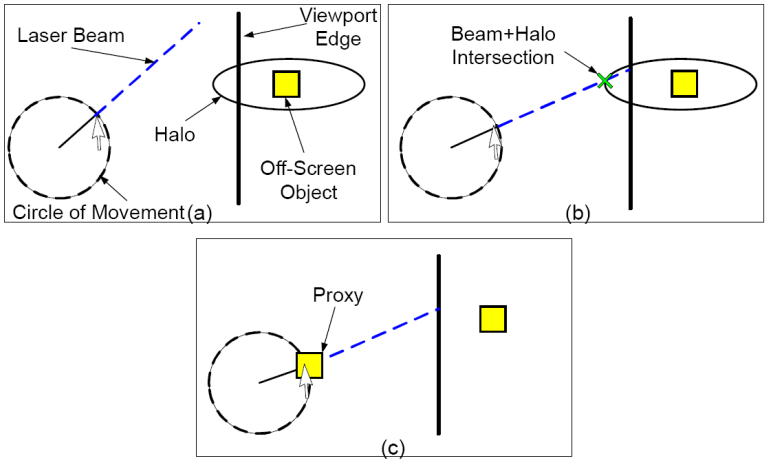


Figure 4: The Hop interaction style: the laser beam is created (a) when the user drags, when the beam intersects a halo (b) a proxy icon is created (c). Image from [31] © 2006 ACM, Inc. Reprinted by permission.

resentation, and a control condition with no navigational aid. The researchers found that 3D arrows outperformed the other conditions in all situations except when experienced users navigated the realistic environment, in which case the 3D arrow performed no worse than the other techniques

Other than spatial dimensions, an off-screen pointer can use available visual features to encode any data dimension required by the designer. For example, the designer could use transparency to indicate relevancy, colour as a type (with a corresponding legend), or even attach extra glyphs such as stars to indicate, as the example in Figure 3c shows, a restaurant's rating.

Finally, a pointer can provide more than just information to the user. It may also act as a convenient handle that can be used as a surrogate object, offering similar interaction modalities to those supplied by the actual target.

Off-screen pointers can be used to facilitate navigation, such as Halo in the Hop [31] and WinHop [32] navigation techniques, illustrated in Figure 4. These techniques, developed by Irani et al., allow the user to first inspect properties of an off-screen object and then easily navigate to that object. Dragging on the screen invokes a *laser beam* – a vector drawn from the starting drag position through the current cursor position to the edge of the display. When the laser

beam intersects a halo arc, the system draws a proxy icon close to the cursor. The user may then investigate the proxy (and possibly choose to pan to the object's true location) by simply clicking on the icon. WinHop differs from Hop by displaying the off-screen point of interest in a separate overlaid window, allowing the user to investigate the area around the point before leaving the currently displayed region. Off-screen pointers are an essential component of the Hop and WinHop interaction style, where they provide both visual cues for undisplayed points of interest and handles that can be exploited to improve navigation through a map.

2.2.2 *Theoretical support for Halo*

The human perceptual system is able to reproduce a circle from an arc quite accurately. The Halo technique takes advantage of this ability to give clues to the user of the location of interesting points located close to the viewable area. This perceptual ability is, in part, explained by the Gestalt principle of closure [54]: as shown in Figure 5a, we perceive full, closed shapes even when they are partially occluded. Additionally, the Gestalt principle of symmetry [54], which states that we tend to perceive symmetric objects, comes into play here.

Beyond the traditional Gestaltist principles there is a modern theory to explain how humans interpret a Halo arc. The theory of *amodal perception* describes how our visual system completes partially occluded shapes [17, 50]. Amodal perception is rooted in the evolutionary adaptation of our visual system and allows humans to recognize partially visible objects in their environment [51].

There are two prevailing models for amodal perception. The *global* model describes how the perceptual system tends to adopt the most regular or symmetrical solutions [52] (Figure 5b); this might explain why halos are perceived as complete wholes based solely on the visible arc portion. The *local* model suggests that the visual system completes the occluded part by connecting the extension of the visible contours [52] as shown in Figure 5c. Continuity and simplicity are the prevailing principles of the local model. There are proponents

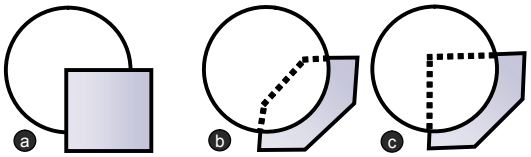


Figure 5: (a) Processes of amodal perception lead the visual system to infer a full circle covered by a square. (b) The shaded figure is completed based on symmetry/regularity – described by the global model. (c) In this case the figure is completed based on continuation – described by the local model.

for both theories but it seems that both local and global processes are involved when humans perceptually complete occluded objects [50].

Cinematographers use a concept similar to amodal perception called *partially out of the frame* [40]. This is a well-known technique for suggesting the presence of an object off-screen by partially intruding it into the recorded frame. The viewer is cued to the presence of the object even if they cannot completely recognize it. Baudisch and Rosenholtz [2] invoked this concept when describing how Halo is interpreted.

2.2.3 *Poggendorff effect*

A user's ability to interpret Halo may be limited by the Poggendorff effect [29]. This effect is the basis of an illusion first described by Johann Poggendorff in 1860 which involves the incorrect perception of a line segment's orientation when partially occluded. As shown in Figure 6a-b, the left line segment is interpreted as continuing on the right with the top-most line when it is actually continued with the bottom-most.

Despite their lack of straight line segments, halos may be affected by this phenomenon when the intersection angle of the circle's tangent is misinterpreted, leading to the perception of a smaller circle. Figure 6c-d shows how the tangent is typically perceived and how that leads to a smaller circle.

Although this consistent underestimation of circle radius and of off-screen target distance was noticed by Baudisch and Rosen-

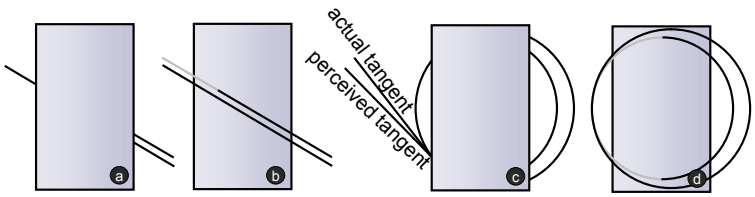


Figure 6: Two versions of the Poggendorff illusion. Traditional version: a) the left line segment appears to continue with the bottom-most line on the right, b) removing the occlusion shows that this is not the case. Similarly, the circular version: c) the left arc appears to be part of the smaller circle, but d) it is actually part of the larger.

holtz [2], the solution they proposed of simply drawing a larger circle will not effectively compensate for the Poggendorff effect because the effect increases with the steepness of the intersection angle [29] and a larger circle will (all else being equal) have a steeper intersection angle. Furthermore, increasing the circle radius of the Halo will increase the size of its on-screen arc, leading to more clutter and severe cropping by the opposing edges of the display.

Automatically correcting for the Poggendorff Effect is also problematic because the effect differs between males and females, as well as between right and left-handed users [41]. The effect diminishes with practice and prolonged inspection [47] as well.

2.3 DISTORTION-BASED TECHNIQUES

Another approach to the problem of displaying large datasets on smaller displays is to distort the data to emphasize the most important information at the expense of poor visibility for less useful data, what Cockburn et al. [16] call a seemless integration of focus and context. First introduced by Furnas [19], distortion-based techniques visually compress some of the contextual data to provide space for a highly emphasized focal point. The choice of what to emphasize varies among techniques, but in general, the emphasized data points are near a user-chosen focal point. Such systems must provide a mechanism for choosing a new focal point, which means that navigation is still an issue in these environments.

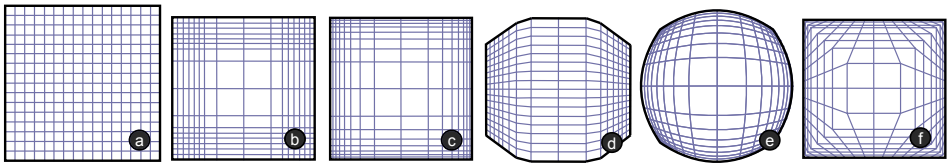


Figure 7: Traditional distortion techniques: a) undistored; b) Bifocal Display; c) progressive fisheye; d) Perspective Wall; e) optical fisheye; f) normalized optical fisheye. Image adapted and redrawn from Leung and Apperley [39] © 1994 ACM, Inc. Reprinted by permission.

Distortion-based techniques are often called fisheye views because resemble the effect of fisheye lenses used by photographers. The fisheye lens provides an exaggerated field of view by emphasizing the center of the image and de-emphasizing the periphery. The fisheye view mimics how our biological vision system works: Ware [54] explains that as light enters our eye, the current scene is sensed by photoreceptors on the retina in the rear of the eye. The density of photoreceptors is not uniform, but is instead mostly concentrated at one small location called the *fovea*. The fovea contains a large percentage of the total photoreceptors but is responsible for only a very small portion of our field of vision. Of the total $200^\circ \times 120^\circ$ that humans are generally able to see [57] (as cited in [16]), the fovea covers 2 degrees at most [54]. It is very difficult to perceive any detail using our peripheral vision, but a wide field of vision is crucially important for helping us maintain the context of our current focus. The same principle is exploited in fisheye visualization techniques to provide the user with a highly focused area and the surrounding context.

Figure 7 shows some of the classic distortion techniques surveyed by Leung and Apperley [39]. One can see that there are many ways to distort the view. The Bifocal Display [53], shown in Figure 7b, has two distinct amplification levels. The progressive fisheye (Figure 7c) is similar to the Bifocal Display, except that the Bifocal Display de-emphasizes all peripheral content by the same amount, while the progressive fisheye de-emphasizes peripheral content over a gradient. Figure 7d, shows the Perspective Wall [49], which exploits our perceptual ability to correct distortion when viewing a 2D projection of

a 3D space. Figure 7e mimics an optical fisheye lens by distorting the distance in a polar coordinate system, but has the obvious disadvantage of a non-rectangular external shape. Figure 7f is a normalized version of Figure 7e that corrects the distorted shape but maintains the desired effect.

Semantic Fisheye views [34] are a class of techniques that are applied not to a visual representation, but rather to the underlying data from which the visual representation was created. In this case, the distortion comes in the form of changing the attributes of peripheral data points (such as colour, font, size, etc.) or perhaps selectively choosing the most important data points to be displayed.

The Furnas Fisheye [19] is an early example of a semantic fisheye. Furnas created a source code browser that selectively displayed important lines of code, such as function declarations, in the periphery. Furnas generalized his selection algorithm as $DOI(a, b) = API(a) - D(a, b)$, where $DOI(a, b)$ is the degree of interest in line a, given the current focal point of line b. $API(a)$ is the *a priori* importance of line a (e. g., function declarations are viewed as more important than lines containing a variable declaration) and $D(a, b)$ is the number of lines of code from a to b.

The computed DOI was then used to determine if a given line was to be displayed or not. Therefore, for a given line to be displayed, it must either be important or close to the focal point (or a little bit of both), leading to a display that shows all lines near the focal point (regardless of importance), but only important lines in the periphery. The DOI function creates an effect similar to a fisheye lens, but the effect is accomplished by selecting data points (i. e., line of code) instead of compressing pixels.

The Furnas Fisheye, though extremely influential, had not until recently been formally evaluated. Jakobsen and Hornbæk [33] implemented a fisheye browser for Java source code, based on Furnas's ideas, and embedded it into an Integrated Development Environment (IDE). In their subsequent evaluation they found that users performed tasks significantly faster when using the fisheye view and users preferred it over a traditional source-code editor.

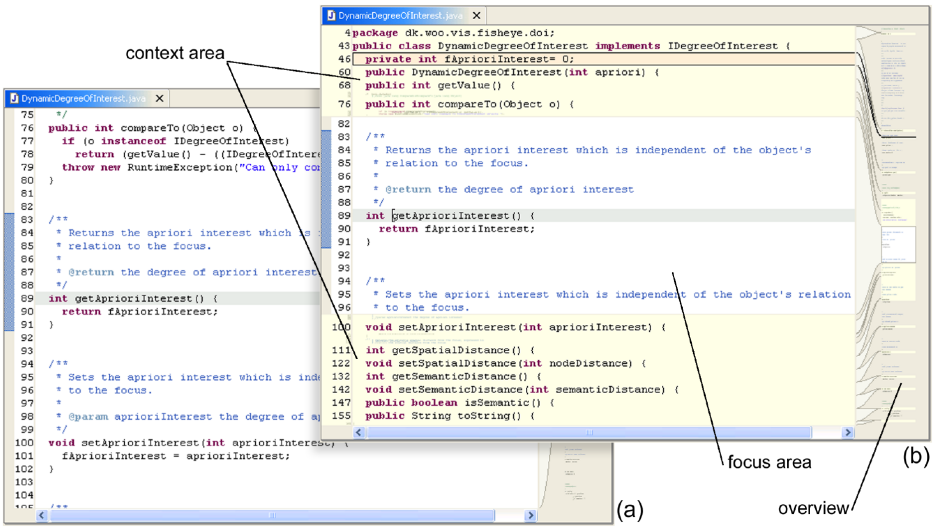


Figure 8: Fisheye Java sourcecode browser, based on the ideas from the Furnas Fisheye. Image from [33] © 2006 ACM, Inc. Reprinted by permission.

Although Gutwin and Fedak [28] found that fisheye techniques outperform panning and zooming in some applications, the distortion creates strange visual effects that users must correct for. Gutwin [27] showed that users have difficulty with targeting tasks in distorted displays because the target appears to move as the focal point approaches it. Gutwin and Fedak [28] also found that users prefer to use non-distorted techniques, despite performing better with the distortion techniques.

3

CHARACTERISTICS OF AN OFF-SCREEN VISUALIZATION TECHNIQUE

I have identified seven qualities of off-screen visualization techniques that characterize a technique's usefulness. The characteristics are: unobtrusiveness, positioning ability, ranking ability, scalability, even spread, distinctness, and continuity.

The techniques for visualizing off-screen points of interest I presented in the previous chapter are all inadequate in some respect. In this chapter, I discuss how each established off-screen visualization technique performs with respect to the seven desirable characteristics and show how each existing technique fails to provide a balance of these important characteristics.

3.1 UNOBTRUSIVENESS

The most important characteristic of an off-screen pointing technique is that it is *unobtrusive*. The technique should not extensively encroach into the user's main area of focus. Working with off-screen points of interests is often secondary to the more important task of investigating the detail currently displayed (e.g., when using a mobile device to help navigate through an unknown city). If the visualization technique occupies too much of the main window, performance of the primary task will suffer [9]. Therefore any provision given for interacting with off-screen objects should not interfere with the primary task.

The overview+detail technique provides a low-resolution view of off-screen data. However, in order to be useful, the overview window must take up a significant portion of a mobile device's small display. The occlusion of the main view by the overview window makes this technique less suitable, though not completely unsuitable, for a range of situations on small screen devices.

Similarly, distortion-based techniques use large amounts of screen space for displaying contextual information, making them mostly inappropriate for deployment to devices with small displays.

Arrows (simple and scaled) are usually implemented as small glyphs placed near the edge of the screen and as such are quite unobtrusive.

City Lights was designed to be minimally intrusive and only occupies a very small band along the screen edge. To accomplish this, City Lights is deficient in several other areas. However, it is exceptionally unobtrusive and an excellent choice for designers that are severely limited in available screen space.

EdgeRadar uses a transparent overview band located along the screen edge. The band is larger than that employed by City Lights but because of its transparency it is also quite unobtrusive. Recall that EdgeRadar is effectively an overview+detail view where the overview window has been split up and placed at the edge of the screen, lessening the main drawback of overview+detail by not obscuring the central area of the window.

Halo is also limited to the screen edge where the arcs are overlaid onto the map content. Similar to EdgeRadar, the map content can be seen through the visualization making it fairly unobtrusive.

Overall, most existing techniques are unobtrusive to the main task except, to some extent, overview+detail and distortion-based interfaces.

3.2 POSITIONING ABILITY

Another important characteristic is the technique's *positioning* ability, that is, how well can a user determine an object's position in off-screen space from the displayed graphical elements. Some techniques only provide relative distance information that require exact knowledge of some other off-screen object's position to determine the desired object's position.

Overview+detail provides the users with two distinct views (at different zoom levels) which can create problems for resolving the location of a target in one space based on its view in the other.

However, overview+detail implementations usually provide a full miniaturized view of the world in the overview window, allowing the user to match distinctive landscape features between the two views and reasonably determine the position of points of interest in off-screen space.

Distortion-based displays use continuously variable zoom levels for different levels of the display (with more detail near the focal point and progressively less the further from that point). This makes it somewhat difficult to determine the exact position of a point of interest but because, like overview+detail, salient map features are also scaled, making it generally possible to place a point of interest at the correct location on the user's mental map.

However, users suffer a cognitive penalty when compensating for the visual effect of the distortion, inhibiting the usefulness of this type of interface [27]. This is especially true for map viewers, where a distorted image will turn otherwise recognizable landmarks into incomprehensible shapes and straight roads into confusing curves.

Simple arrows do not provide enough information for precise positioning – only the direction to the target, not the distance. Scaled arrows, on the other hand, provide relative distance information by scaling the size of the arrow proportionally to the distance of the target from the screen edge. Using the size of a scaled arrow for a known position it is possible to roughly position a target, by comparing arrow sizes.

City Lights lacks the ability to accurately show the position of an object in off-screen space. The deficiency is partially addressed by using two distinct colours to signify that some points of interest are near and others are far.

EdgeRadar is similar to overview+detail with respect to positioning, except only the targets are shown and not the landscape, making it only possible to accurately position a target if a reference point is known.

Halo, on the other hand, solves many of the problems inherent with the techniques just discussed. It provides both direction and distance information explicitly in the size and shape of the on-screen arc. Although Halo provides enough information for users to accurately

determine the position of the object via the angle of the arc, users tend to consistently underestimate the distance to the target [2].

Positioning is a difficult characteristic to excel at and all existing techniques leave much to be desired. Of the available techniques overview+detail and Halo provide the best positioning performance.

3.3 RANKING ABILITY

For many tasks, off-screen objects must be easily *ranked* by distance from the edge of the screen. This is primarily important for cursory determinations of what are the closest off-screen points of interest. For example, "Where is the nearest fast food restaurant?" The ability to emphasize close targets is desirable in many applications because points of interest closest to the current view are often of greatest importance and should therefore be emphasized.

Overview+detail and distortion-based interfaces provide reasonable support for ranking of showing each object's position in relation to the on-screen area. However, ranks must be calculated by the user from the position – they are not explicitly encoded and available at a glance.

Although simple arrows provide no support for ranking, scaled arrows encode rank as the size of the arrow. The smallest arrow represents the object closest to the screen edge and the largest represents the furthest.

City Lights has very minimal ranking ability. It simply encodes two categories of distance (near and far) with two distinct colours.

EdgeRadar is similar to overview+detail by providing a scaled down view of off-screen space that users can interpret to determine the rank of each point of interest. EdgeRadar is perhaps slightly easier than overview+detail because only objects off-screen are shown.

Halo does provide useful rankings, the smaller an arc the closer the target. As targets get further away the difference in arc size for the same change in distance shrinks and approaches zero. However, that is generally desirable because small differences between distant targets are less important than between close targets.

All techniques except simple arrows and City Lights provide a reasonable mechanism for ranking, while scaled arrows performs quite well in this regard.

3.4 SCALABILITY

A versatile off-screen visualization technique can represent off-screen points of interest at any reasonable distance from the screen edge. Often this means that the growth of visual elements in response to increased distance from the screen edge is bounded. In general, it is important that the technique is reasonably *scalable* with respect to distance.

Overview+detail interfaces are scalable to reasonable sizes. The designer must simply choose a higher ratio of zoom levels between the detail and overview windows. Researchers have proposed a practical limit on the ratio of 20:1 [16]. A higher disparity than this is difficult for people to use.

Distortion-based interfaces behave similar to overview+detail with respect to scalability. The amount of off-screen space that can be represented is governed simply by the ratio of zoom-levels for the detailed focal point and the contextual area.

Simple arrows do not encode distance information at all, so they are infinitely (and trivially) scalable. Scaled arrows are scalable and only limited by the size of the on-screen arrow glyph.

City Lights are scalable but only because of the extremely limited support for representing distance (i.e., two levels encoded with colour).

EdgeRadar is similar to overview+detail and distortion-based interfaces. To represent a larger area, simply increase the ratio of zoom levels.

Halos grow as a function of distance from the edge. Objects that are farther away have larger halos, which results in more clutter, as shown in Figure 9. Within a fairly small distance – around 1.5 screen widths – a halo becomes too large to fit entirely on the screen edge. Therefore Halo is not sufficiently distance-scalable.

Figure 9: Halos grow as the off-screen objects gets farther away. This figure shows the halos for a series of objects at 50 pixel increments from the screen edge.

All techniques, except Halo, are reasonably scalable with respect to distance, with overview+detail, arrows and EdgeRadar outperforming the rest.

3.5 EVEN SPREAD

It is also desirable for the technique's graphical components to be *evenly spread* around the screen edge. This is a very important characteristic; with most off-screen visualization techniques, the placement of the graphical element is chosen such that the off-screen object is orthogonal to the screen. It is not possible for objects located in the area extending from the corner of the display to be displayed in the same area as those for the side. Because of this, the graphical elements for these objects are generally placed in a small area at the corner to avoid conflicting with objects on the edge. This placement is reasonable for small off-screen areas but for a larger off-screen area the corner area grows exponentially. Figure 10 shows how the size of the off-screen area, in relation to the on-screen area, affects the area containing corner objects. The center rectangle represents the on-screen area and the full square is the area represented by the visualization. When the off-screen area grows relative to the on-screen, a greater proportion of the off-screen area is represented by the corners (shown above as shaded areas).

The corner problem is not an issue with overview+detail interfaces. However, some distortion-based interfaces such as the 2D Bifocal Display [53] can be affected because the corners are rendered differently than the sides. Most other fisheye displays evenly spread the targets radially from the focal point.

Figure 10: A large off-screen area leads to a concentration of off-screen pointers at corners. The shaded area represents the portion of the screen not orthogonal to the screen.

Simple and scaled arrows are generally not affected by this problem because the arrows are often laid-out radially, pointing out from the center of the display.

City Lights is seriously affected by the corner problem. Any number of objects located beyond the screen corner are represented by a single small arrow.

EdgeRadar also suffers from the corner problem but to a lesser degree than City Lights. The corner regions are compressed into a small square placed in the screen corners. The amount of on-screen representation space given to corner areas is much less than that given to regions orthogonal to the screen edges.

With Halo, arcs for corner targets are drawn in the corner and are severely cropped by the opposed screen edge. Baudisch and Rosenholtz [2] alleviated this to some extent by giving the corner targets more space (see Figure 11), but, as they admit, this only corrects a small amount of the arc cropping.

Many of the off-screen visualization techniques suffer from the corner problem due to uneven spread of the on-screen representation. Only overview+detail and arrows are immune.

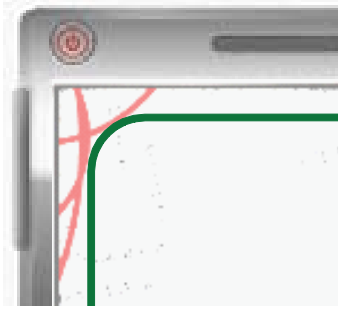


Figure 11: The fix used by Baudisch et al. [2] to partially overcome the corner problem with Halo. The green rounded rectangle represents the maximum intrusion depth for halo. The rounded corners allow more on-screen space for halos representing targets beyond the screen corner.

3.6 DISTINCTNESS

Another important characteristic is that the graphical components of the technique for each off-screen object should not overlap, or if they do, each should remain visually *distinct*. Furthermore, when off-screen objects are in motion, the visual components remain distinct as two objects pass.

With overview+detail interfaces each target is represented unambiguously as a *blip* (as on a radar screen) that will only overlap if the off-screen objects themselves are collocated.

In distortion-based interfaces, each target is scaled at the same rate as its surrounding environment and is distinct as long as the scale is sufficient to represent the object.

Simple and scaled arrows have some risk of overlap when off-screen targets are roughly collocated. This problem can be addressed with more sophisticated placement algorithms that intelligently place arrows to avoid overlap.

In the case of City Lights, two or more targets roughly collocated are only represented by a single on-screen representation causing this technique to perform quite poorly for this characteristic.

EdgeRadar is based on the same idea as overview+detail, where each target is represented by a blip and as such each target is distinctly identifiable.

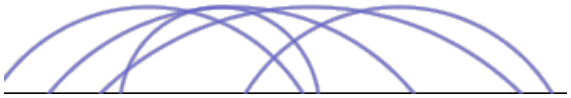


Figure 12: Halos become easily cluttered with a few off-screen targets. In this case there are only five targets.

With Halo clutter is a major problem. A proliferation of off-screen objects results in many overlapping arcs that are difficult to distinguish and interpret, as shown in Figure 12. Often two overlapping arcs can create a third arc-like shape that may be misinterpreted as a third target.

In general, the set of techniques performs well with respect to distinctness. However, City Lights and Halo experience problems in this area in some situations.

3.7 CONTINUITY

When an object enters or leaves the screen there should be a fluid transition from the on-screen to off-screen representation. Without this *continuity*, the user must reacquire an object that they are tracking when it leaves the screen, incurring a performance penalty and introducing a source of confusion.

In all cases except overview+detail, the techniques seamlessly change from an on-screen object to an off-screen representation when the target leaves the screen (usually from the user panning in the opposite direction). Overview+detail forces the users to abruptly adjust their focal point when they are tracking a target.

3.8 SUMMARY

To summarize the discussion in this chapter, Table 2 lists how these leading techniques perform with respect to each of the characteristics I just laid out. For each characteristic, I provide a subjective score based on my opinion from zero to four check marks, where zero means the technique is completely inadequate; one means the technique has some features that support this characteristic but they

	unobtrusive	positioning	ranking	scalable
Overview+detail	√	√√	√√	√√√
Distortion	√	√	√√	√√
Simple Arrows	√√√	N/A	N/A	√√√
Scaled Arrows	√√√	√	√√√	√√√
City Lights	√√√√	N/A	√	√√
EdgeRadar	√√√	√	√√	√√√
Halo	√√√	√√	√√	√

	spread	distinct	continuity
Overview+detail	√√√	√√√	√
Distortion	√√	√√√	√√√
Simple Arrows	√√√	√√√	√√√
Scaled Arrows	√√√	√√√	√√√
City Lights	√	√	√√√
EdgeRadar	√	√√√	√√√
Halo	√	√√	√√√

Table 2: Performance of current techniques for each of the desirable characteristics of a mobile off-screen visualization technique.

are mostly ineffective; two means has adequate support for the characteristic; three means the technique provides very good support; and four means the technique is exceptional in this respect. These ratings are purely subjective and serve only as a summary of the discussion in each section.

As you can see in the table and discussion in this chapter, all existing techniques are deficient in some respect for the purpose of representing off-screen targets on mobile displays. In the next chapter, I introduce a technique that I designed to overcome some of these deficiencies.

4

DESIGN AND EVALUATION OF WEDGE

In this chapter, I describe the design and evaluation of a new off-screen pointing technique called *Wedge*. I specifically designed Wedge to overcome some of the problems with existing techniques I outlined in Chapter 3.

Wedge combines the positioning ability of Halo, using the same principles of amodal object completion, with the small size and possibility of adaptive layout of arrow-based techniques.

The design of Wedge came about iteratively by investigating the effects of various modifications to Halo to reduce the effects of clutter. In this chapter, I recount that design process, then describe in detail the final version of Wedge and discuss an experimental evaluation of Wedge, comparing it to Halo.

4.1 WEDGE DESIGN PROCESS

The design of Wedge resulted from a rigorous design-implementation-evaluation cycle. Initially, I investigated if changes to the visual characteristics of Halo (such as colour, fill, etc) could reduce clutter by allowing users to visually separate overlapping arcs. Following this, I studied several structural changes to the Halo shape before finally coming to the Wedge technique itself. I carried out small pilot studies throughout the design process to evaluate my designs. The outcome from each phase resulted in design decisions that led to incremental improvements of the system.

SURFACE PROPERTIES

In this phase, I manipulated the visual properties of the halo arcs such as colour, fill-in, transparency and by adjusting intrusion depth through various schemes. Some of the modified versions of Halo are

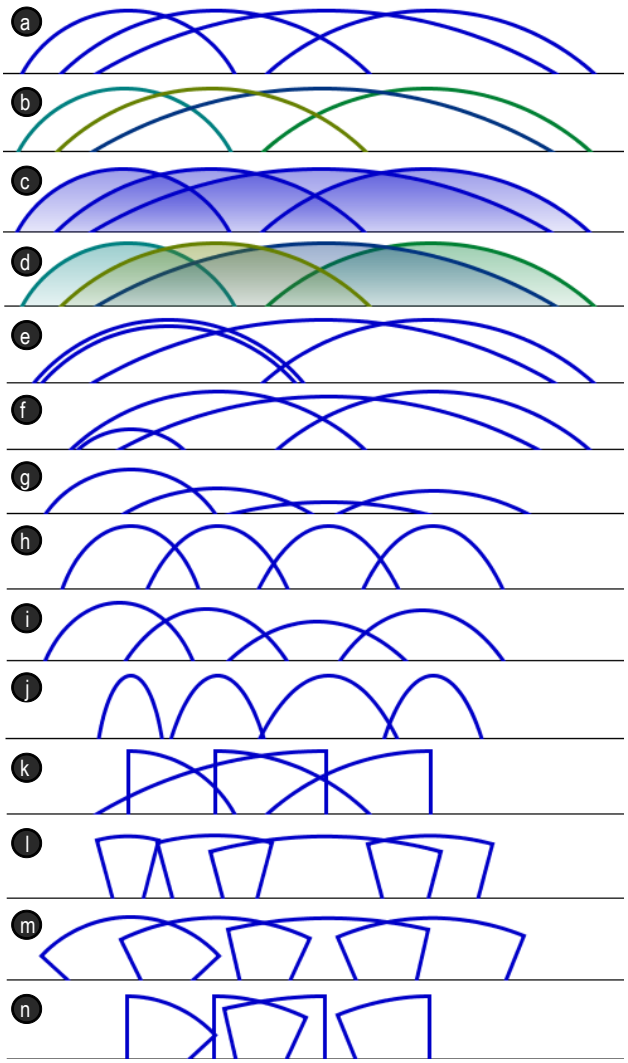


Figure 13: Early designs of an improved Halo: a) regular Halo; b) colour; c) filled in; d) fill and colour; e) clustering; f) varying intrusion depth; g) constant arc length (intrusion); h) constant arc length (compression); i) constant arc length (hybrid); j) oval (compression); k) half arc; l) pie slice (constant aperture); m) pie slice (constant arc length); n) pie slice (rotated).

shown in Figure 13. The goal of this phase was to determine if Halo variations can reduce clutter from overlapping arcs and improve accuracy in terms of identifying exact locations of off-screen objects.

I performed a series of informal pilot studies where participants were asked to complete a location estimation task (similar to the Locate task in the formal evaluation presented later in this chapter). This allowed me to determine which of the changes to the visualization allowed users to best estimate the location of off-screen objects in the presence of clutter.

In all cases, I found very little improvement in user interpretation of the Halo arcs. Beyond giving me an opportunity to fine-tune the experimental task, this phase allowed me to discard modifying surface properties as a legitimate method of reducing clutter among Halo arcs.

STRUCTURAL CHANGES

Instead of changing the surface properties of Halo to make them stand out amongst clutter, in this stage I looked at modifying the shape and structure of the halos to make them smaller and therefore avoid clutter from the beginning.

I tried two methods of making halos smaller: 1) compression – changing the halo from a circle to an oval; and 2) cropping – lopping off a portion of the circle.

To determine if these approaches were worthy of more in-depth investigation, I performed a small study comparing ovals to half arcs to full halos. Five student volunteers participated in the experiment and were asked to click on the screen where they believed each off-screen target was present (similar to the Locate task presented later in this chapter).

The experimental system presented each participant with a series of randomly placed targets (the set was the same for each technique and each participant) between 0 and 640 pixels from the screen edge in groups containing 1, 4 or 8 targets.

The system calculated error amounts as the distance from the participant selected point to the actual target position. A repeated

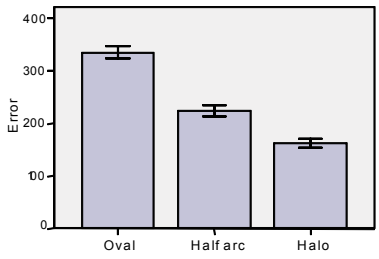


Figure 14: Results from Oval vs Half arc vs Halo study. Error bars indicate ± 1 standard error.

measures Analysis of Variance (ANOVA) showed an overall main effect of visualization type on error rates ($F(1,3)=34.090$, $p=0.009$). Posthoc pairwise comparisons show that each visualization was significantly different from the others (all $p<0.01$) with error amounts for Halo (mean 162.81 pixels) less than Half arc (mean 223.91 pixels) which were in turn less than Oval (mean 334.84 pixels). The results are summarized in Figure 14 and the raw statistics are available in Appendix A.1.

Although the study did not show much promise for changing the Halo shape, it did provide an important contribution: oval halos, as used in other published systems [31], are much more difficult to interpret than standard Halo and I would not recommend their use to avoid clutter in systems that depend on the accuracy of user interpretation.

APPROACHING THE WEDGE SHAPE

The final design phase was based on the insight that cropped halos performed poorly despite their smaller size because the irregular shape was very difficult to interpret when they overlap (even though they were less likely to overlap than standard halos).

In this phase I developed a series of cropped Halo versions, implemented an adaptive placement algorithm that actively avoided overlap, and eventually ended up with the Wedge off-screen pointing technique described in detail in the proceeding sections.

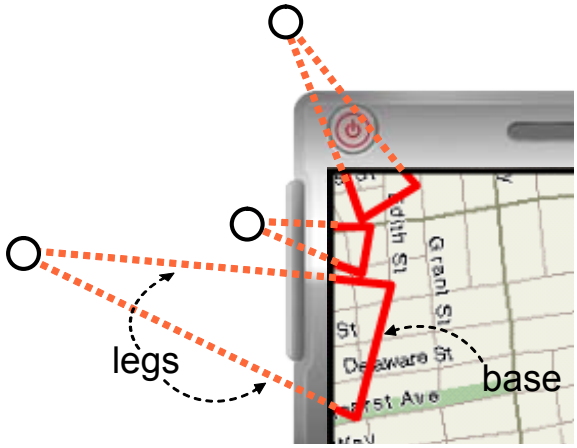


Figure 15: The Wedge off-screen pointing technique. The pair of legs point toward the target, converging off-screen at the position of the target. The base connects the legs and completes the shape.

4.2 WEDGE TECHNIQUE

After the thorough iterative design process I chose a final Wedge design, shown in Figure 15. Each wedge consists of three line segments: two *legs* of equal length and one terminating line called the *base*. The legs are the key element. In order to locate the off-screen object referred to by a wedge, users visually trace the legs, extrapolate them across the display edge, and estimate where they intersect. The intersection point is the location of the off-screen object.

The base plays an important role by connecting the legs. On a screen with multiple wedges, it is the bases that allow users to match legs, thus preventing users from tracing a pair of legs which belong to different wedges. To function properly, the bases should overlap as little as possible.

The base may be a straight line or it can be an arc with its center point located at the off-screen location (see Figure 16). Both form factors have benefits and drawbacks. The angles produced by the straight base can serve as an additional cue reinforcing distance. The curved base, in contrast, offers a distance cue by means of circle completion, as introduced by Halo. In the case of a curved base, however, vertex angles do not provide any additional cues, as they are always 90 degrees. The choice has little consequence and for

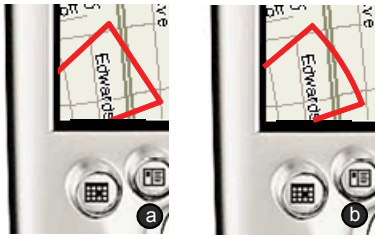


Figure 16: Options for the base of a wedge: a) straight or b) curved.

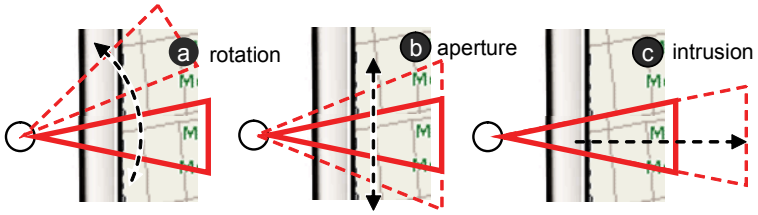


Figure 17: Each wedge has three adjustable degrees of freedom: a) rotation, b) aperture/base length and c) intrusion that can be used for various purposes.

clarity I illustrated the Wedge technique with a straight base in most of this thesis. The experiment described later in this chapter used the curved base version of Wedge.

The key aspect that distinguishes Wedge from its predecessors is its three degrees of freedom. As shown in Figure 17, we can change the a) rotation, b) aperture, and c) intrusion of a wedge, while it still points to the same location. Only modification to intrusion depth was possible with Halo.

The three degrees of freedom of each wedge can be used for three distinct purposes:

1. *Avoiding overlap with other wedges.* To resolve the overlap of the two wedges, we can either rotate the wedge away from each other (Figure 18b), reduce their aperture/base length (Figure 18c), or in some cases reduce the intrusion of one wedge (Figure 18d).

2. *Maximizing the location accuracy communicated by a wedge.* Each wedge exists to allow users to accurately locate an off-screen target, but a look at Figure 18 suggests that some of these wedges work better than others. The thin wedge in Figure 18c,

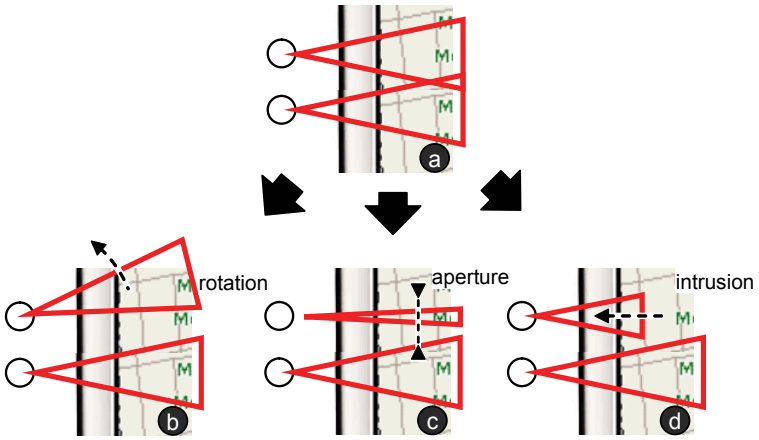


Figure 18: Overlap between two wedges can be resolved by adjusting wedge b) rotation, c) aperture, and/or in some case d) intrusion.

for example, might not work as well as the rotated wedge in Figure 18b.

3. *Serving as an additional cue or proxy for distance.* Aperture and intrusion can also be used to communicate a certain target property by convention. For example, larger wedges could refer to more distant off-screen objects. An easily-interpreted distance cue is important for tasks not involving absolute distances, such as determining which of two targets is closer.

The primary goal of Wedge design is to achieve maximum accuracy. On the other hand, it is clear that overlap and clutter have a huge affect on the readability of wedges, ultimately impacting accuracy more than any other factor. I therefore designed a Wedge layout algorithm that prioritizes as follows: 1) avoid overlap, 2) maximize location accuracy, and 3) provide an additional distance cue. This prioritization is not strict, since I must still attain a balance of the three goals described above.

In terms of the desired characteristics for off-screen visualization techniques laid out in the previous chapter, the Wedge layout algorithm attempts to improve the *distinctness* and *even spread* of the on-screen elements by actively avoiding overlap. Wedge also was designed to make gains with respect to *positioning* and *ranking* by virtue of its more easily interpretable shape.

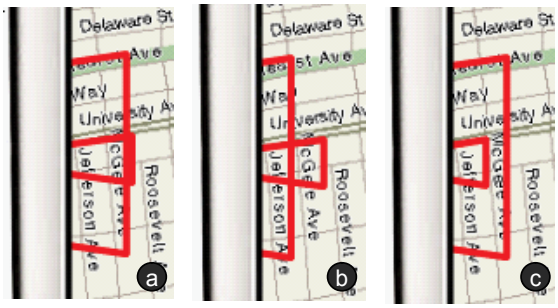


Figure 19: Two overlapping wedges shown in each of the three options for leg length: a) constant leg intrusion; b) shorter leg intrusion for further targets; c) longer leg intrusion for further targets.

4.3 WEDGE LAYOUT ALGORITHM

The layout algorithm chooses base and leg lengths that are proportional to the distance of the target from the edge of the screen. This choice improves location accuracy and provides a cue for distance (i. e., large wedges represent distant objects). Once the wedges have been placed on the screen in their initial configuration, the system runs an avoidance algorithm to dynamically resolve overlap by rotating wedges away from one another.

The layout algorithm creates an acceptable layout of wedges by manipulating intrusion depth, base length, and rotation, as described in this section.

Intrusion

I considered three primary options for mapping distance to intrusion: a) constant intrusion, b) inversely proportional: shorter intrusion for further distances, and c) directly proportional: longer intrusion for further distances (Figure 19). While constant intrusion led to increased overlap between wedge outlines (Figure 19a), the other two mappings naturally reduced overlap. These two mappings also have the potential to serve as an additional distance cue.

I chose the directly proportional mapping (Figure 19c), because it increased accuracy (see Chapter 5 for a detailed discussion as to

why) and as a positive side effect, it allowed me to nest wedges in some cases as shown in Figure 19c.

I calculated wedge leg intrusion using a function that contains a non-linear component causing leg intrusion to gradually level off for very distant objects. This ensured that leg intrusion depth did not grow in an unbounded fashion, which would occlude the main display. The function used to set total leg length was:

$$leg = dist + \ln(\tfrac{(dist+20)}{12}) \times 10$$

where *leg* is the length of each leg in pixels and *dist* is the distance in pixels between target and the edge of the screen. The additive constant of 20 pixels assures a minimum intrusion for close targets.

Base length

The base length of each wedge was mapped linearly to target distance using the following function:

$$base = (5 + dist \times 0.3)/leg$$

where *base* is the length (in pixels) of the base, *dist* is the distance of the target to the edge of the screen (in pixels), and *leg* is the overall length of each leg (in pixels). This mapping serves as the primary cue for target distance. The constants were the result of several pilot studies and they balance positioning performance and risk of overlap: larger base lengths would have led to more accurate positioning and easier ranking, but at the expense of a significantly increased risk of overlap.

During pilot studies I discovered that, just like Halo, participants tended to underestimate the distance to the off-screen target. In an attempt to correct this, I rendered each wedge 25% longer than it should be as in Figure 20. Although I did not formally evaluate the corrected vs. the uncorrected version of Wedge, I believe the correction did not have a large effect.

Figure 20: An attempt to correct the underestimation bias that users experience with Wedge: a) uncorrected version; b) correction version representing a target an exaggerated 40% further away than the actual target.

Corners

Screen corners have traditionally been a challenge for all off-screen pointers because they represent a large proportion of off-screen space [2, 25, 58], as discussed in Section 3.5. At the same time, they offer less space for the proxy representing the target. The halo arcs in Figure 11, for example, are cropped, reducing their accuracy substantially [2].

The additional degrees of freedom offered by wedges help alleviate this problem, yet this case still requires additional attention. Wedges for extremely distant objects will also be cropped by the edges of the screen when displayed in the corners. To alleviate this, the layout algorithm (see Listing 4.1) will increase leg intrusion, up to an arbitrary maximum of 20 pixels, until the wedge is centered in the corner with at least 10 pixels of empty space between each vertex and the edge of the screen. If the new leg intrusion was still unable to show a sufficient portion of both wedge legs, the algorithm would decrease the base length to the point of making the wedge fit in the corner. As a result of this, wedges will always show legs in the corners. The distances used in the experiment later in this chapter were rarely large enough for this correction to occur.

Listing 4.1: Corner correction portion of the layout algorithm.

```
// First, increase intrusion depth
fullArcLength = max possible arc length
arcLength = calcArcLength()
count = 0

while arcLength > fullArcLength - 20
    intrusion += 1;
    arcLength = calcArcLength()

    if( count++ > 20 )
        break;

// Second, if necessary crop arc length
if arcLength > fullArcLength - 20
    arcLength = fullArcLength - 20
```

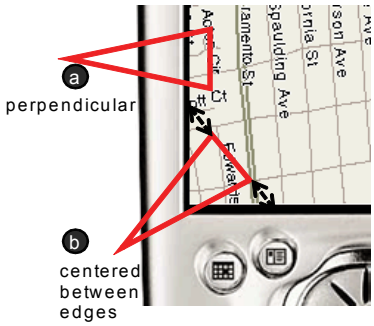


Figure 21: Wedges are initially placed a) perpendicular to the screen edge, or if in the corner, b) centered between the two opposing edges.

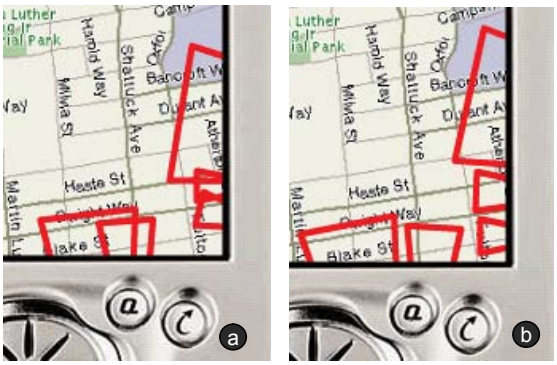


Figure 22: A cluster of wedges a) before and b) after applying the wedge overlap algorithm.

Listing 4.2: Avoidance portion of the wedge layout algorithm.

```
foreach wedge in clockwise order
   for i=0 to 10
       if !overlap(current, previous) and \
          !overlap(current, next)
           break

       if overlap(current, previous)
           if !tooCloseToEdge(previous)
               previous.rotation += 3 pixels

           if !tooCloseToEdge(current)
               current.rotation -= 3 pixels

       if overlap(current, next)
           if !tooCloseToEdge(next)
               next.rotation -= 3 pixels

           if !tooCloseToEdge(current)
               current.rotation += 3 pixels
```

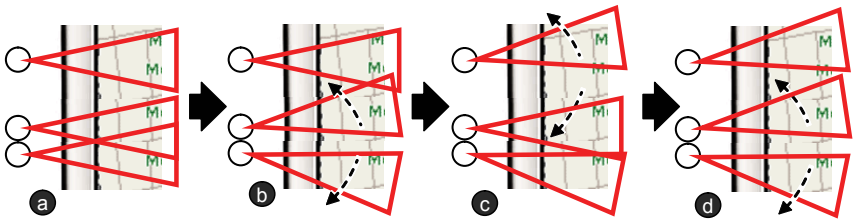


Figure 23: Iterations of the avoidance portion of the wedge layout algorithm: (a) Two wedges overlap, (b) they are rotated away causing new overlap, (c) new overlap is resolved, creating yet another overlap, and (d) all overlap is avoided.

Rotation

Rotation is the primary means of avoiding wedge overlap. Figure 22 shows a cluster of wedges before and after resolving overlap using rotation. Fortunately, rotation has little impact on intrusion and none on aperture, so it does not affect the distance cues conveyed by intrusion and aperture.

Rotation is computed using a simple iterative algorithm. It is reasonably inexpensive computationally and offers real-time performance for maps with up five overlapping wedges.

Initially, wedges located along a screen edge are placed perpendicular to that edge (Figure 21a). If a wedge is near a corner, the algorithm places it such that there is an equal amount of on-screen space on either side of the wedge (Figure 21b). Next, the algorithm iterates to resolve overlap. The algorithm traverses all wedges on screen in clockwise order (according to the location of their base centers, not by target location). The algorithm rotates wedges away, by a small amount, from neighbors with which it overlaps. This can propagate overlap to neighboring wedges and is resolved through repetition as shown in Figure 23 and Listing 4.2. If there is no solution the algorithm will terminate after a fixed number of iterations, leaving wedges with as little overlap as possible.

4.4 USER STUDY TO EVALUATE WEDGE

To evaluate the utility of Wedge, I performed a user study[1] that compared the effectiveness of Wedge with Halo. I hypothesized that Wedge would be more accurate than Halo, primarily when it represents objects that get mapped to the corner of the display. I was also interested in identifying the effects of each of these techniques in high density layouts.

The experiment was conducted using custom software written in Adobe Flash that simulated a handheld PDA. A simulated PDA

1 The study described in this section is an extended version of that presented in [26]. Due to time constraints that publication reported results based on 16 participants. The results here are from an extended study with 36 participants.

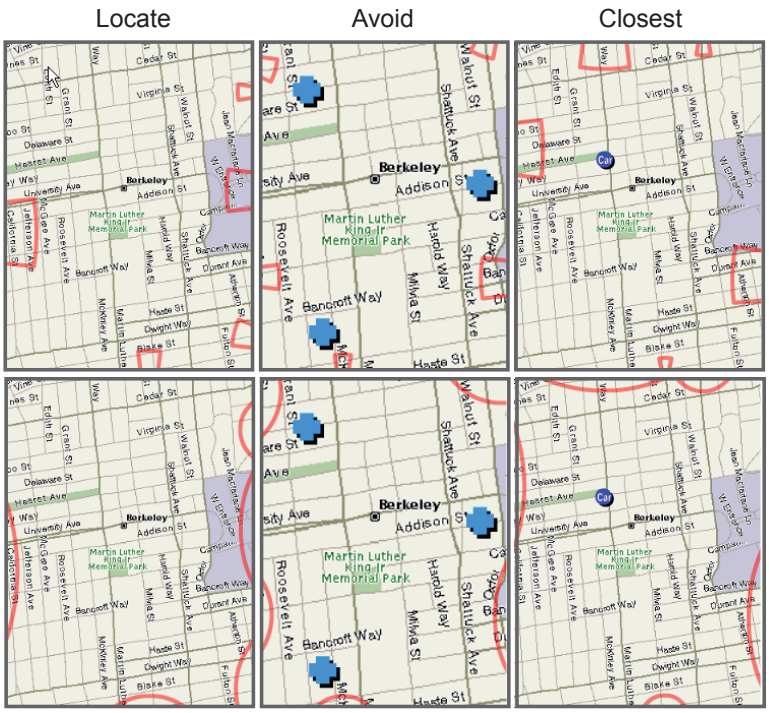


Figure 24: Screenshots of each task from the experiment application used during the evaluation. Wedge in top row, Halo in bottom row.

screen was shown in the center of a 19" monitor at slightly larger than real-life size. Participants interacted with the simulated PDA via a standard mouse interface.

Halo was implemented using the original code written by Baudisch and Rosenholtz [2]. Wedge was implemented exactly as described above, except that each wedge base was curved.

The study used three tasks from Baudisch and Rosenholtz's [2] comparison of Halo to arrows, see Figure 24:

Locate The users clicked in the off-screen space at the expected location of the off-screen targets indicated by each of the two visualizations. Users located targets in any order and the system automatically picked the closest match.

Avoid As an "ambulance dispatcher," the user selected the hospital farthest from traffic jams. Each map contained indicators of five on- or off-screen traffic jams, and three blue cross-shaped icons representing hospitals.

Closest Each map contained a blue car icon and five red wedges/halos representing restaurants. The user's task was to click on the wedge/halo corresponding to the off-screen location closest to the car.

In all tasks the distances to the off-screen targets ignored the road grid and participants were instructed to compare distances "as the crow flies" and not take into account the displayed roads.

The Locate task directly assessed the accuracy of each visualization, while the other two were secondary tasks that looked at how the visualizations could be used in realistic problem solving scenarios.

I controlled the total on-screen line length between the two conditions by choosing functions for wedge base and leg lengths such that the overall average on-screen line length for every target used in the study was equal (75 pixels for both Halo and Wedge). A wedge and halo for a given position may have different on-screen line lengths but *overall*, for all positions used in this study, the average length was equal.

To explore whether overlap affects performance, I tested two different configurations of targets: *sparse* and *dense*. In sparse conditions, the targets were organized such that overlapping halos were minimized. In dense conditions, the five targets were positioned so that all of the halos/wedges were packed into a smaller area. The sparse conditions were programmatically converted to dense conditions by folding the display (once for Avoid and Closest tasks, twice for Locate task) such that each target was placed onto the same side of the display at the exact position as they were on the other side, as shown in Figure 25. As a result, the dense condition simulated the amount of clutter that would be equivalent to 20 (Locate task) or 10 (other tasks) off-screen objects. This procedure ensured that sparse and dense maps were comparable in terms of distance of each target from the screen edge and the location of each target in relation to a corner.

A second issue that can also affect clutter and density is whether or not the on-screen visualizations are placed in the corner or along the edge of the screen. In the Locate task, a target was considered in the corner if it was not located directly orthogonal to the on-screen

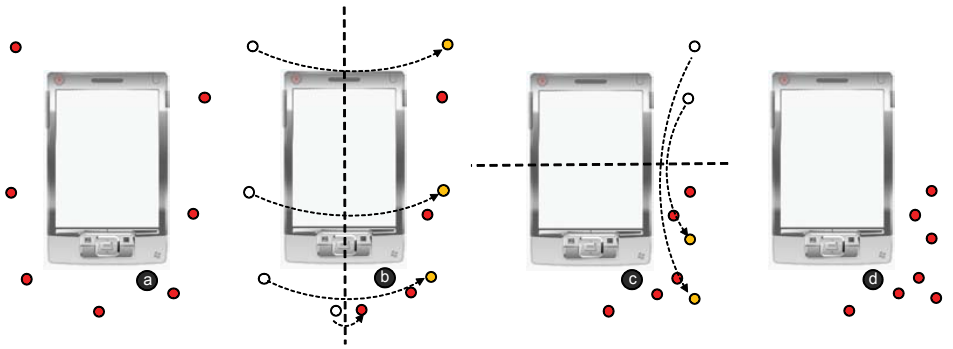


Figure 25: Folding procedure used to create dense conditions from sparse layouts: a) original sparse layout; b) folding along vertical axis – all targets on the left are moved to the right; c) folding along horizontal axis – all targets on top are moved to the bottom; d) final result with all targets in bottom right quadrant.

window. Each Locate trial contained five targets, some of which would be located in the corners.

For the Closest task, a trial was considered a corner trial if one or both of the first and second closest targets were located in the corner. A corner trial in the Avoid task was one where off-screen objects were purposely placed in the corner such that the participant must have considered that location when performing the task.

A total of 39 undergraduate students participated in the study in exchange for course credit. Due to unnaturally large error rates, the results from three participants were removed from the analysis. Of the remaining participants there were 22 males and 14 females. The participant's ages ranged from 18 to 30 and all had normal or corrected to normal vision.

The study used a $2\times2\times2$ factorial design with three factors: Visualization (Halo or Wedge), Density (dense or sparse target clustering) and Position (corner or side of the screen).

Participants were shown the experiment application and were given a brief demonstration of both Halo and Wedge. They were supplied with two sheets of paper showing examples of interpreting Wedge and Halo (supplied in Appendix A.2). They then performed the three tasks; first on four training maps, and then 16 (for Locate) or 32 (for Closest, Avoid) test maps. The order of tasks and display conditions were fully counterbalanced. After the session, participants

	Halo	*Wedge*
Locate	49.7 pixels (24.1)	36.2 pixels (19.92)
Avoid	33.0% (19.6)	31.5% (22.9)
Closest	42.7% (22.8)	36.4% (21.3)

Table 3: Summary of error results – means (standard deviation)

	Halo	*Wedge*
Locate	2.41 sec (1.33)	2.17 sec (0.88)
Avoid	4.16 sec (2.19)	4.20 sec (1.97)
Closest	4.31 sec (3.48)	4.94 sec (3.29)

Table 4: Summary of completion times – means (standard deviation)

were asked to state which visualization type they preferred for each task. The experiment application collected error and completion time data.

4.5 RESULTS

In this section the results are organized by task. Within each task I consider the effects of each of the three factors (visualization type, density, and position) on error and completion time. Note that in all analyses, participant was included as a random factor. Tables 3 and 4 show summary means for all measures and tasks and Appendix A.2 contains all raw results from the analysis.

Task 1: Locate the off-screen location

The first task asked participants to click on the locations of the off-screen objects indicated by each wedge or halo on the screen. The system gathered data about error amount and the completion time to locate each of the five targets. The error amount was the Euclidean distance from the guessed position to the target's position.

Figure 26 shows the error amounts and completion times for Halo and Wedge in all conditions (dense and sparse; corner and side). I carried out a 2×2×2 ANOVA (Visualization × Density × Position)

to test for differences. We found main effects of all three factors: for Visualization, $F(1,35)=19.003$, $p<0.001$; for Density, $F(1,35)=28.843$, $p<0.001$; and for Position, $F(1,35)=135.800$, $p<0.001$.

As can be seen from Figure 26 larger errors were seen in corner trials (mean 53.5 pixels) than in side trials (mean 32.5 pixels). There were also larger errors in dense configurations (mean 46.1) than sparse configurations (mean 39.9). The overall difference between visualizations was about 13 pixels (Halo mean 49.7 pixels; Wedge mean 36.2 pixels).

In addition, there was a significant interaction between Visualization and Position ($F(1,35)=40.745$, $p<0.001$). As shown in Figure 26, the difference between visualization types is considerably larger in corners than on the sides of the screen, which supports my hypothesis that the reduced space in corners causes additional problems for Halo interpretation. There was also an interaction between Visualization and Density ($F(1,35)=4.399$, $p=0.043$), supporting the hypothesis that increased clutter in general also leads to problems interpreting Halo.

We also tested for differences in time to complete the task (see right chart in Figure 26). A $2\times2\times2$ ANOVA showed no significant effects of Visualization ($F(1,35)=3.222$, $p=0.081$) or Density ($F(1,35)=0.640$, $p=0.429$). There was a significant main effect of Position ($F(1,35)=10.107$, $p=0.003$) on task completion time and a significant interaction between Density and Position ($F(1,35)=4.540$, $p=0.040$), but no interactions with Visualization.

Task 2: Avoid the traffic jam

The second task asked participants to select one of three on-screen objects that was furthest from a set of off-screen objects. I gathered error rate and completion time data.

Figure 27 shows error rates and completion times for the different visualizations, densities, and positions. A $2\times2\times2$ ANOVA did not show any effects of Visualization ($F(1,35)=0.541$, $p=0.467$), Position ($F(1,35)=0.699$, $p=0.409$), or Density ($F(1,35)=3.654$, $p=0.064$). In addi-

tion, there were no interactions between any factors except between Position and Density ($F(1,35)=4.422$, $p=0.043$).

A 2×2×2 ANOVA showed no effects of any of the three factors on task completion time (Visualization $F(1,35)=0.037$, $p=0.848$; Position $F(1,35)=0.007$, $p=0.932$; Density $F(1,35)=0.085$, $p=0.772$), and no interactions between any factors.

Task 3: Find the closest restaurant

The third task asked participants to select the closest off-screen object to an on-screen icon. Again, I gathered error rate and completion time data.

Figure 28 shows error rates for the different visualizations, densities, and positions. A 2×2×2 ANOVA showed significant main effects of Visualization ($F(1,35)=4.793$, $p=0.035$) and Position ($F(1,35)=204.200$, $p<0.001$), but not Density ($F(1,35)=0.019$, $p=0.890$). There was a significant interaction between Density and Position ($F(1,35)=45.821$, $p<0.001$), but no interactions with Visualization.

A 2×2×2 ANOVA showed significant main effects of Visualization ($F(1,35)=7.399$, $p=0.010$) for task completion time in favour of Halo and Position ($F(1,35)=14.231$, $p=0.001$), but did not show effects of Density ($F(1,35)=1.764$, $p=0.193$). There were no significant interactions between the factors.

Overall Preferences

After the experiment the participants were asked to state which visualization they preferred for each task. Table 5 summarizes the subjective preferences. In the Locate task there was a clear preference for Wedge ($\chi^2(1,N=35)=6.429$, $p=0.011$). More participants preferred Wedge in the Avoid ($\chi^2(1,N=34)=1.882$, $p=0.170$) and Closest ($\chi^2(1,N=32)=0.500$, $p=0.480$) tasks, but those differences were not statistically significant.

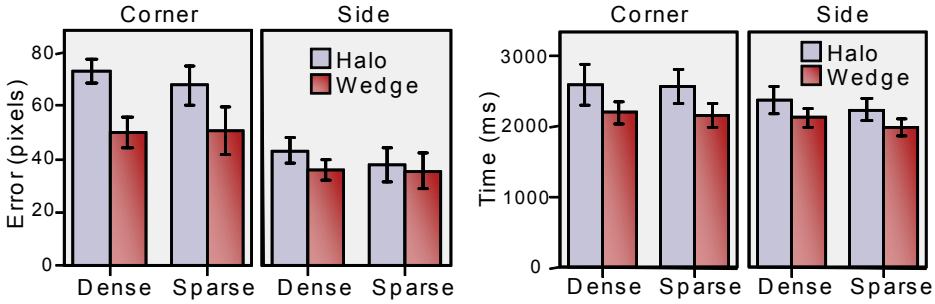


Figure 26: Locate task error amount and completion times by visualization, density, and position. Error bars indicate ± 1 standard error.

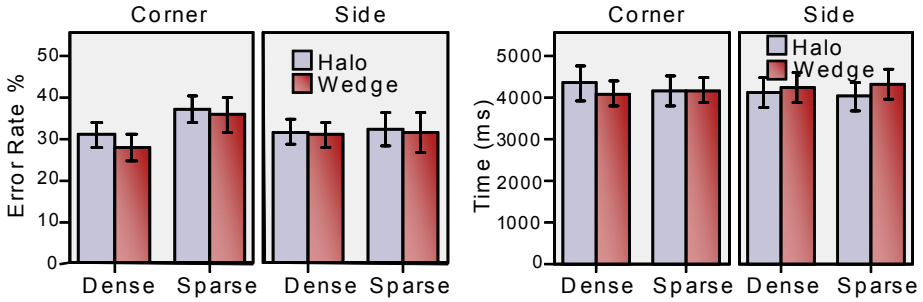


Figure 27: Avoid task error amount and completion times by visualization, density, and position. Error bars indicate ± 1 standard error.

Figure 28: Closest task error amount and completion times by visualization, density, and position. Error bars indicate ± 1 standard error.

	Wedge	*Halo*	*No Preference*
Locate	25	10	1
Avoid	21	13	2
Closest	18	14	4

Table 5: The number of participants who preferred each visualization technique for the three tasks.

Participant Comments

Comments made during the experiment suggested reasons for the advantages for Wedge over Halo. One user said, "I found that when the rings overlap it is almost impossible to tell which is the right ring. Wedges just seem natural." And another stated, "overlapping rings made it very confusing at times. Directional wedges helped a lot, and they also seem to take up less space. More information meant less thinking with the wedges."

Participants' comments also provided some insight into the reasons why Wedge was somewhat less preferred in the Closest task – that the difference between distant and close off-screen objects was easier to determine with Halo, since there is a large visual difference in this case. One participant stated that, "the sizes of the arcs did not require too much calculation or thinking to spot the smallest ring."

4.6 DISCUSSION

My hypotheses were that Wedge would be more accurate than Halo, and that this effect would be stronger in corners and in dense conditions. The Locate task provided evidence in support of all three hypotheses: accuracy with Wedge was significantly higher than Halo, and the difference was larger in corners and in dense layouts.

In the Avoid and Closest tasks where people had to make use of this accuracy, I did not find many significant differences between Wedge and Halo. Users were significantly more accurate in the Closest task with Wedge but they were also significantly slower – suggesting a time/accuracy trade-off. Part of the reason for the lack of difference is that these tasks involved strategy more than the Locate task; therefore, it is possible that strategy choice overshadowed the beneficial effects of Wedge that were seen in the Locate task. In addition, the Closest task revealed an advantage for Halo (the large visual difference between distant and close objects) that I had not considered. Subjective results reinforce these findings - Wedge was strongly preferred for the Locate and Avoid tasks, in which Halo has several problems and few advantages.

Overall, the results confirmed my hypotheses and show the benefits of the new visualization technique. These benefits are more pronounced when off-screen objects are clustered into corners, where wedges allow users to triangulate the location of off-screen objects more precisely. I believe that Wedge's overlap-avoidance algorithm aids users in determining direction and distance. While I chose a brute-force approach for the layout algorithm a proper optimization technique to fit and layout the wedges optimally in the limited display space could prove useful.

To successfully complete these tasks, it appears that participants employ different strategies. It is clear that for the Locate task, participants are extrapolating the full shape of the wedge and halo to locate the off-screen object. In this task, I reason that the visual shape of the wedge more clearly shows the shape completion process needed to perform the Locate task. In the case of the Avoid and Closest tasks, users seem to rely primarily on distance cues. As we see from the results, the distance cues in Wedge are as good as those provided by Halo, and in some cases even better.

Based on the results of the study, I propose the following recommendations to mobile application designers:

Consider Wedge. Off-screen object information could be displayed using Wedge as the primary off-screen pointing technique: it offers significant improvements over Halo.

Reduce overlap. Designers should reduce overlap in any visualization of off-screen objects, as overlap leads to reduced accuracy and greater difficulty identifying objects.

Rotation is better than overlap. None of the participants were concerned about the rotation of the wedges, although several comments were received about the difficulty of the overlapping halos. Therefore, I believe that rotation should be chosen over either cropping or overlap for off-screen pointers.

Corners need special attention. The results confirm that designers need to pay special attention to the design of off-screen pointers so that they work equally well in the display corners.

Strike a balance. Designers need to strike a fine balance in selecting parameters for off-screen pointers to avoid as much overlap as possible, maximize location accuracy and to create an excellent cue for distance.

In designing Wedge I set out to improve the Halo technique with respect to the *distinctness* and *even spread* characteristics, hoping that this will address a major problem with Halo: clutter. By nature of its smaller size and by dynamically avoiding overlap Wedge avoids clutter and allows for more targets to be clearly displayed in the corners. The results of this experimental evaluation provide excellent support that Wedge outperforms Halo in *positioning* and some marginal support for an improvement in *ranking*. The experiment also indicates that this is partially because of the improved distinctness and spread of Wedge. Furthermore, the ability to choose functions for intrusion and base length allows designers to draw wedges that are *scalable* to the distances required by a specific application. Table 6 shows is an updated version of Table 2 that lists the subjective performance of Wedge with respect to Halo.

	unobtrusive	positioning	ranking	scalable
...				
Halo	√√√	√√	√√	√
Wedge	√√√	√√√	√√	√√
	spread	distinct	continuity	
...				
Halo	√	√√	√√√	
Wedge	√√	√√√	√√√	

Table 6: Performance of Halo and Wedge for each of the desirable characteristics of a mobile off-screen visualization technique.

5

IMPROVING WEDGE PERFORMANCE

Most of the published work relating to off-screen visualization are point studies that introduce and/or evaluate a novel technique. This type of work, though very important for the introduction of new ideas, is generally only a small portion of the body of knowledge for a mature field. As the field of HCI matures, more work must concentrate on generalizing specific results, supplying results and introducing robust metaphorical models that can be used to predict future results and assist in the design of new techniques.

The BRETAM framework, introduced by Gaines [20], classifies technology into six categories: Breakthrough, Replication, Empiricism, Theory, Automation, and Maturity. Inspired by this model of technology development, Cockburn and Gutwin [13] suggested that the maturity of a science can be determined by the distribution of published results that are categorized in each of the BRETAM stages. In that respect, most of the published work related to off-screen pointing is in either the Breakthrough, Replication or Empiricism stages and very little in the Theory or higher abstract stages.

In this chapter I cap my thesis with an attempt to move beyond the current practice in the field by introducing a preliminary model to describe user performance with Wedge and similar techniques. The model is based on the fact that users do not necessarily perceive an off-screen object at the indicated location. Targeting error tends to consists of 1) an error distribution/scattering around the target and 2) a systematic biases. The exact nature and magnitude of the effects and their dependence of target location and type of on-screen cue is unknown at this point and I provide some preliminary results and a framework that may eventually lead to a full model.

My model framework breaks Wedge interpretation error down into four measures: *bias*, misinterpretation of the distance to an off-screen target; *offset*, misinterpretation of the direction a wedge is pointing;

orbital length, the variability in distance guesses; *orbital width*, the variability in direction guesses. The purpose of the model is to relate measures of Wedge size and shape to these four measurements of error.

I present a pointing study with 21 participants that was used to gather data on user performance in a triangle completion task. The task has obvious parallels to interpreting a wedge pointing into off-screen space. Based on the results of the study and guided by my model, I present a series of recommendations to implementers of Wedge and similar techniques.

The study results do not directly support much of the model and I was unable to provide much in the way of specific predictive equations for Wedge performance. I present them both here as preliminary work in the area and future work is certainly necessary to improve and verify the model.

5.1 WEDGE MODEL FRAMEWORK

The purpose of this model is to relate measurements that describe a specific wedge (such as its length and rotation) to measurements of user interpretability.

A wedge has many adjustable parameters that describe its look and shape. Here is a mostly complete list:

wedge length Overall length from base to tip.

base length Length of base connecting line.

leg length Overall length of the legs.

top/bottom leg extrusion Length of the off-screen portion of each leg.

top/bottom leg intrusion Length of the visible portion of each leg.

rotation Angle the entire shape is orientated.

aperture Angle separating the legs.

top/bottom leg to edge intersection angle Angle of intersection between each leg and the edge of the screen.

leg to base intersection angle Interior angle at the leg-base vertex.

Many of these parameters are inter-related, for example the leg length is simply the addition of the intrusion and extrusion segments.

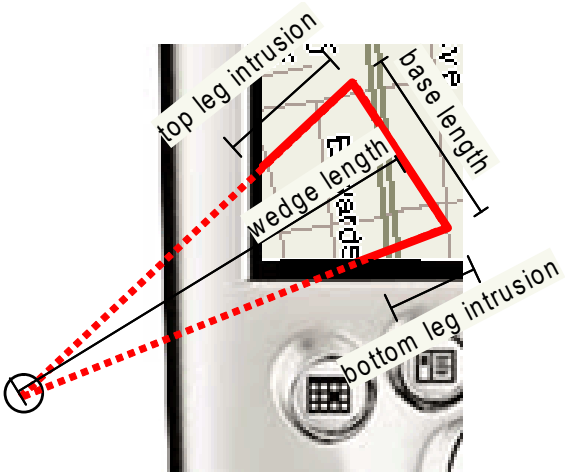


Figure 29: The degrees of freedom for Wedge used in the model.

The difference between the top and bottom intrusion/extrusion lengths depends on the angle of rotation of the entire wedge (or, if located in the screen corner, on the amount of cropping by the opposing edge). Furthermore, the various angles (aperture, rotation, intersection angles) can be expressed in terms of the line segment lengths using simple trigonometry.

I decided to describe a wedge in terms of four parameters, from which all others can be derived. These proved to be the most practical for my purposes. I call these specific parameters the degrees of freedom of a wedge because they are the parameters that can be adjusted by a layout algorithm as presented in the last chapter. The Wedge degrees of freedom I chose are: *base length*, *top leg intrusion*, *bottom leg intrusion*, and *wedge length* (illustrated in Figure 29).

I chose these degrees of freedom because it is a small set that completely describes a wedge's shape and size, they are mostly visible on-screen, they are easy to compute, and they represent salient features of the shape. Also for the purposes of the model it is convenient to have all parameters in the same units (i.e., pixels) – which is the main reason for choosing base length instead of aperture (in degrees).

Corners are one of the more interesting cases because wedges located in the corner are not only rotated but will have one of its legs cropped by the opposing edge of the screen. By representing

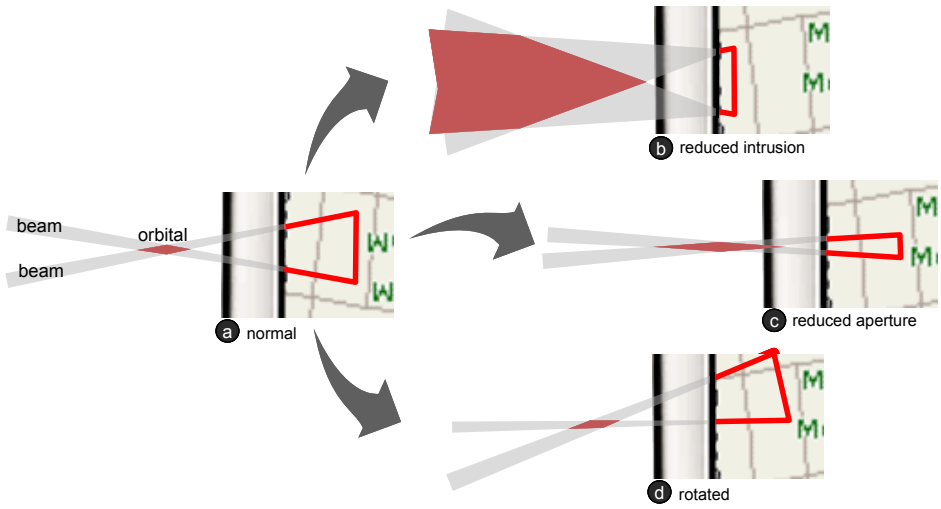


Figure 30: a) The orbital is the area where users expect the target to be located. Its shape and size are determined by the intersection between the two beams emerging from each leg. Changing b) intrusion, c) aperture, and d) rotation of the wedge affects size and shape of the orbital.

a wedge with two separate on-screen wedge leg extrusions, I was able to model both rotated wedges and corner wedges with same parameters.

My goal is to create a formal model that predicts the interpretability of a wedge. Interpretability is a complex topic and to discuss it further, I introduce the term *orbital*. I derive the term from chemistry, where a molecular orbital is the region in which an electron may be found in a molecule – basically, a two dimensional probability distribution. Figure 30 illustrates the concept used with Wedge. While we tend to think of Wedge legs as the first part of a line pointing toward the off-screen target, there is a certain amount of uncertainty about the angle. As a result, the shape emerging from a leg is not a line, but a cone. I call them *beams*. The intersection of the two beams is where the user would expect to find the off-screen target; this is the orbital. In reality, beams and orbitals have a fuzzy perimeter, but for the sake of simplicity, I illustrate them as solids. The size of the orbital depends on two factors: beam spread and intersection angle.

The spread of each beam depends on the length of the leg from which it emerges. In the same way that a rifle fires more accurately than a pistol, long legs resulting from deeper intrusion result in

thinner beams (Figure 30a) than the shorter legs of a wedge with shallow intrusion (Figure 30b). Note that the orbital of a wedge is infinite if the outside edges of the two beams diverge, as is the case in Figure 30b. In this case, a wedge provides users with an estimate of the minimum target distance, but not with an estimate for the maximum distance.

The angle under which the two beams intersect depends on the aperture separating the wedge legs. Decreasing the aperture of a wedge (Figure 30c) generally leads to a shallower angle, resulting in a longer orbital.

Rotating a wedge decreases the spread of one beam at the expense of increasing the spread of the other. This results in a skewed orbital (Figure 30d).

Using this framework, I believe it is possible to develop a predictive model of Wedge performance that relates Wedge length, base length, and leg lengths directly to orbital size and position. The next section describes an experiment that begins to provide the empirical support needed to create such a predictive model.

5.2 DATA COLLECTION EXPERIMENT

In this experiment, I collected data related to the perceptual completion of the triangular shape that forms the basis of the Wedge off-screen pointing technique. Appendix A.3 contains raw results and ancillary material from the experiment.

The experiment consisted of a task requiring the participants to perceptually complete a triangle shape, show in Figure 31. The participants were shown part of a triangle, containing three connected line segments and they were asked to click in the blank space at the location of the remaining vertex. The partial triangle was presented at a different random orientation for each trial.

Participants were shown all triangles resulting from the factorial combination of these independent variables:

leg lengths The length of the lines extending from the base of a triangle. Four conditions used identical leg lengths (4, 8, 16, 32

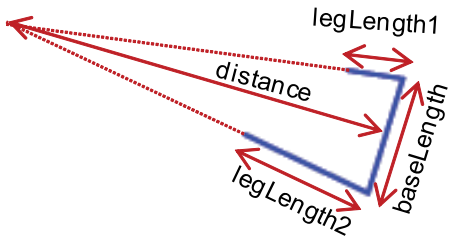


Figure 31: Triangle completion task: the blue lines were shown to the user and they were asked to click where they believed the remaining triangle vertex was located. All dashed red lines are drawn for illustration purposes only and were not shown to the user.

pixels) and six used combinations of those lengths (4/8, 4/16, 4/32, 8/16, 8/32, 16/32).

base length The base of the triangle. Four levels were used (25, 50, 100, 200 pixels).

distance The distance from the base to undisplayed point. Four levels were used (50, 100, 200, 300 pixels).

The set of trials was repeated 3 times for each participant, yielding 10 leg length combinations $\times$ 4 base lengths $\times$ 4 distances $\times$ 3 blocks = 480 trials per participant.

The experiment system recorded depth error and breadth error. Depth error is the difference from the selected point to the actual point of the complete triangle along the length-wise axis of the triangle. Depth error is a measurement of the misinterpretation of the overall size of the triangle. Breadth error was calculated as the distance along the width-wise axis of the triangle from the selected to the correct point. It is a measurement of misinterpreting which direction the triangle is pointing.

Twenty one participants recruited from undergraduate classes participated in the study and all had normal or corrected to normal vision. The participants received course credit for completing the study.

At the start of the session, an administrator provided a brief demonstration of the experiment and instructed the participant to follow the on-screen prompts for the rest of the experiment. The experiment application walked the participant through a training

phase before it began recording error data. The training phase consisted of 20 trials where the participant was supplied with feedback on their performance and the completed shape was briefly shown (0.5 seconds) at the end of each trial. No feedback was given after the training. The training phase was designed to instruct the participants on the task only, not to improve their performance with the technique. I wished to model the real life performance of perceptual shape completion. In real world situations the user is never shown the full shape or any feedback, therefore I designed the experimental phase of the experiment to mimic this minimal feedback environment.

5.3 EXPERIMENT RESULTS

I analyzed the recorded data to determine trends and relationships among the variables and determined the orbital size and shape for each condition. Initially I will consider unrotated wedges (i.e., identical leg lengths) and I will consider rotated wedges separately at the end of this section.

In Figure 32 I show all collected data points for each distance when the base length was 50 pixels and the both legs were 16 pixels. Overlaid onto this figure are the full triangle shapes for those conditions. From this figure you can clearly see the spreading out of the selected points as the distance increases. You can also note how the center of hits (i.e., the bias) separates further from the actual target as distance increases.

In general the distribution of errors for a given leg length, base length and distance can be roughly fit to a bivariate normal distribution oriented with one axis along the length of the triangle and the other across. The two variables of the bivariate distribution are, in this case, depth and breadth error, as described previously. In these terms the orbital is the portion of the error distribution likely to contain most positioning guesses from a user. Figure 33 illustrates the measurements of the orbital discussed in the next subsections.

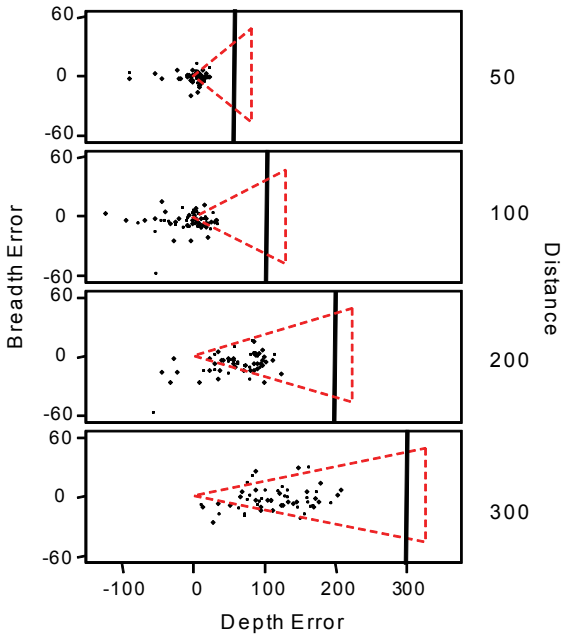


Figure 32: Scatterplot showing all data points from study for all distances, a base length of 50 pixels, and both legs at 16 pixels. Notice how the depth error is much more spread than breadth error.

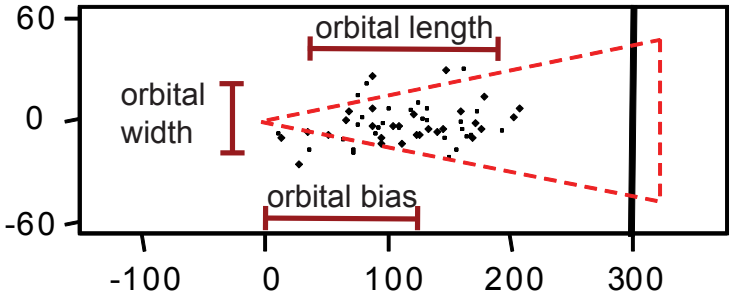


Figure 33: Close-up of Figure 32 showing the measures of orbital bias, width and length. The fourth measure, orbital offset, is not shown in this case because it is zero.

Orbital Bias

The distribution of depth error tends to be centered substantially closer to the edge of the screen than the actual target position. I define the *orbital bias* to be the mean value along the depth axis of the error distribution. From the results of the experiment, shown in Figure 34, it appears that bias is primarily determined by distance (i.e., wedge length) and to a less extent base and leg length. A simple linear regression ($r^2 = 0.625$) produced the following equation:

$$bias = -58.781 + 0.582\,dist - 0.002\,base - 0.375\,leg \tag{5.1}$$

Orbital length

Along with bias, the *orbital length* is an important element that can be modeled. It is effectively a measure of variation in the distribution of depth errors. I chose to define the length of the orbital as the standard deviation from the mean (i.e., bias) along the depth axis for each participant and condition. The experiment indicates (see Figure 35) orbital length is primarily determined by the distance to the target (although to a lesser extent than bias) and the leg and base length are also important factors. A linear regression ($r^2 = 0.135$) of these parameters yields:

$$length = 24.904 + 0.074\,dist - 0.059\,base - 0.217\,leg \tag{5.2}$$

Orbital width

While orbital length is a measurement of misinterpreting distance to an off-screen target, the *orbital width* measures interpretation of the direction to the target. I define orbital width as the standard deviation along the breadth axis of the error distribution for each participant and condition. See Figure 36 for a summary of the results.

Orbital width is much smaller than the length (length is on average almost 4.5 times as large: length mean = 28.21 pixels; width mean

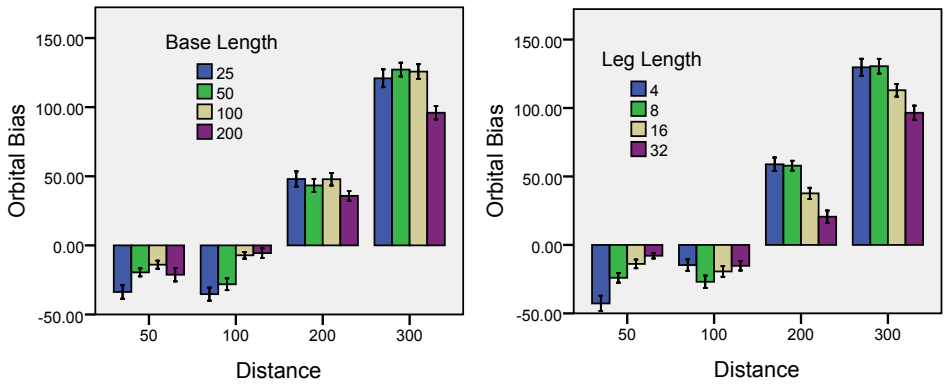


Figure 34: Orbital bias results for unrotated wedges. Error bars indicate ± 1 standard error.

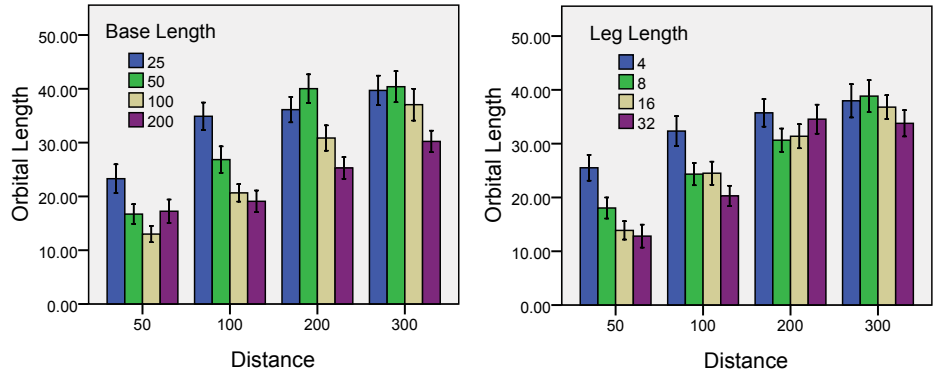


Figure 35: Orbital length results for unrotated wedges. Error bars indicate ± 1 standard error.

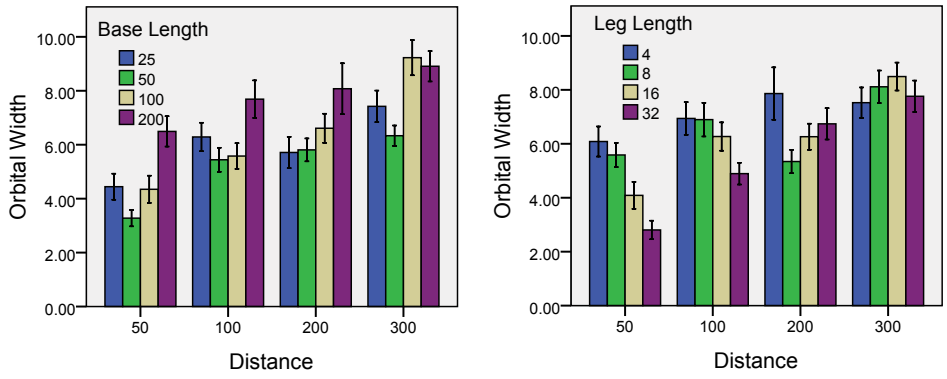


Figure 36: Orbital width results for unrotated wedges. Error bars indicate ± 1 standard error.

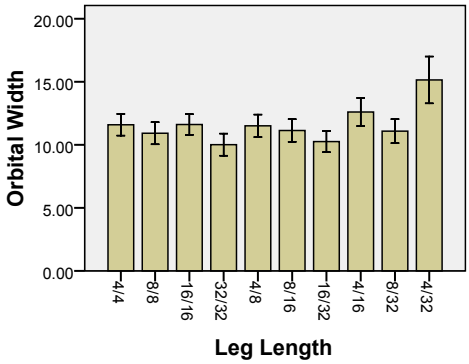


Figure 37: Wedge orbital width for rotated triangles. The leg length values listed are for the top and bottom leg lengths. For example 4/4 is a non-rotated wedge with both legs at 4 pixels and 4/8 has one leg of 4 pixels and the other of 8.

= 6.35). So much so that almost all of the misinterpretation error is in depth. For completeness and despite its poor fit ($r^2 = 0.080$) I include an equation relating orbital width to the wedge factors:

$$width = 4.010 + 0.012\,dist + 0.013\,base - 0.050\,leg \tag{5.3}$$

Equations 5.1-5.3 are clearly too simple to capture all of the complexities of the error distributions. From Figures 34-36 there does appear to be certain patterns and complex interactions between the factors. However, I was unable to mathematically capture those relationships and I therefore must leave a full investigation for future work.

Rotated wedges

When legs are not equal (i.e., the wedge is rotated) the triangle is usually interpreted very similarly to a non-rotated version. However, this is not true when the triangle has a large difference between leg lengths. As you can see in Figure 37, highly rotated triangles, like those with one 32 pixel leg and the other 4 pixels long, have a slightly increased orbital width. Participants appear to misinterpret Wedge direction slightly more when it is heavily rotated.

5.4 MINIMIZING ERRORS

Based on the experimental data, I provide a number of design implications for practitioners such that errors in interpreting Wedge are minimized.

Since large wedges are more intrusive into the main display area and they are more likely to overlap, a system designer should choose the smallest possible leg and base length (which together determine wedge size) that still results in a reasonably small orbital. Since an increased leg and base length will result in a smaller orbital, the exact level of compromise among the variables is a subjective matter. I encourage a system designer to take into account the Wedge characteristics outlined in this chapter along with the specific requirements of the intended system when implementing Wedge.

A base length of at least 50 pixels appears sufficient for close targets (50, 100 pixels from the screen edge) and a base length of at least 100 pixels for further distances. Increasing it to 200 pixels does improve bias at large distances but a 200 pixel base is probably too large to be practical for most systems.

A large base length can actually be detrimental to orbital width. As seen in Figure 36 large bases tend to increase orbital width.

Using an increased leg length (i.e., greater than 8 pixels) does not have much of an effect on orbital length but does decrease bias slightly. Because of this, along with the general desire to avoid wedges from heavily intruding into the display, I recommend varying leg lengths from roughly 8-16 pixels long.

What is clear from the results is that users are inconsistent in their interpretation of wedges. There is both a large amount of variability between participants and within a participant's own performance.

It appears that a system designer should strike a fine balance to limit this variability to reasonable levels, by eliminating situations that increase variability above the base line. For the most part this can be done by eliminating wedges with very small leg and base lengths, especially for distant objects, and avoiding heavily rotated objects as much as possible. However, rotating wedges, even heavily, may still be preferable to displaying overlapping wedges.

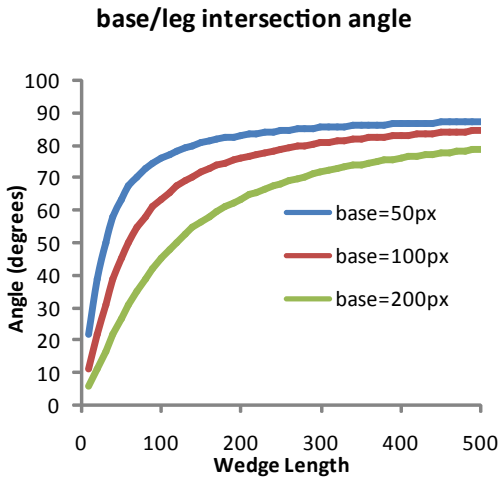


Figure 38: The relationship between distance from screen edge and angle of intersection between a wedge's leg and the screen edge.

5.5 CORRECTING ERRORS

Beyond the suggestions for minimizing interpretation error in the last section, there is the possibility of correcting some of the users' bias.

As discussed in Section 4.3 a simple bias correction was included in the original version of Wedge. That method was simply to draw a wedge for a given distance as if it was actually 25% further away. It is illustrated in Figure 20 back in Chapter 4. As seen in Equation 5.1, bias was actually closer to 50% but only significantly present for distant targets (further than 100 pixels from the screen edge).

Regardless, another reason why the automatic correction scheme from Section 4.3 did not work as well as intended is because as distance grows so does the angle of intersection between the wedge legs and the edge of the screen. The relationship is not linear and levels off fairly rapidly (increased base length will push off the leveling until a little later) meaning that after a certain distance, each wedge for larger distances are virtually indistinguishable from one another (i.e., all are very close to 90 degrees), as shown in Figure 38. This relationship is:

$$f(x, c) = \arctan(2x/c) \tag{5.4}$$

where $f()$ is a function that calculates the current angle of intersection (in radians) of the leg/base vertex for a wedge with x as the current wedge length and c as current base length.

A bias correction scheme that increases overall wedge length leads to a problem: For a given base length, a larger wedge length results in less distinguishability of leg/base vertex angles. The sections of lines in Figure 38 that have leveled off have very small changes in angle compared to the more vertical segments. The distinguishability can be defined at the slope of the curve at a given point, i.e., the derivative of Equation 5.4 with respect to x, which is:

$$f'(a,c) = \frac{2c}{c^2 + 4a^2} \tag{5.5}$$

Now, if the system automatically modifies the wedge length from a to b, what new base length, d would be required to maintain the same rate of change in intersection angle? To calculate, I equate two derivatives of the form of Equation 5.5, modifying the variables and solving for d:

$$\begin{aligned} f'(a,c) &= f'(b,d) \\ \frac{2c}{c^2 + 4a^2} &= \frac{2d}{d^2 + 4b^2} \\ d &= \frac{c^2 + 4a^2 - \sqrt{c^4 - 16b^2c^2 + 8a^2c^2 + 16a^4}}{2c} \end{aligned} \tag{5.6}$$

For small values of a, b, c this function does not produce meaningful results. For example, when c is 50 and $b = a * 1.25$ (like the Wedge length correction function used in Chapter 4) values of c are out of range when $a < 75$.

Given the complexity of Equation 5.6 system designers may choose to precalculate tables of overall Wedge lengths and base lengths for reasonable distances in their system.

I have not tested this bias correction scheme and it is possible that the improvements to distinguishability will be overshadowed by increased overlap caused by the larger wedges. In any event this discussion underscores the complexities involved in improving Wedge interpretability.

5.6 CONCLUSION

In this chapter I presented a model that relates characteristics of a wedge's size and shape to the distribution of interpretation errors, followed by a preliminary data collection experiment to provide empirical foundations for a full predictive model. I then made several recommendations based on the gathered data that will help implementers of the Wedge technique avoid problems.

Although the results presented in this chapter are preliminary, I believe they represent an excellent starting point for developing a full predictive model of user performance using Wedge.

6

SUMMARY AND FUTURE WORK

The display and navigation of large amounts of spatial data (especially maps) on small displays is a common problem for existing navigation and presentation techniques. In this thesis, I report on my thorough investigation of off-screen pointing techniques and describe the specifics of a new technique, called *Wedge*, for visualizing the location of off-screen points of interest. Wedge has specific advantages that make it more useful than traditional techniques.

I describe in detail the design and implementation of Wedge, as well as the results and analysis of an empirical evaluation intended to test the effectiveness of this technique in real world tasks. Furthermore, I include a preliminary theoretical analysis that hopefully serves as the start of a full predictive model of Wedge interpretability. The preliminary model provides insight that may be used to guide future design and could lead to a deeper understanding of off-screen pointing in general.

By dynamically avoiding clutter, the Wedge off-screen pointing technique improves on the *distinctness* and *even spread* of similar techniques, like Halo [2]. I show Wedge to be easier to interpret and to determine the *position* of off-screen points of interest. Wedge is also more *scalable* than Halo, making it suitable for a wider range of deployments.

The main contributions of this research are:

- The summary of existing techniques and a set of criteria for evaluating them.
- The introduction of a new technique, *Wedge*, that is superior in many respects to other techniques.
- The introduction of the concept of degrees of freedom for an off-screen pointing technique and the use of them in a dynamic layout algorithm.

- A thorough evaluation of Wedge that supports its improvements over Halo.
- An empirical investigation and preliminary model of user performance using Wedge.

6.1 FUTURE WORK

The most exciting prospect for future work in this area is the integration of off-screen pointing into a general Fitts' Law [18] model of selection in multi-scale environments. Fitts' Law is a classic model, used extensively in HCI, that can be used to predict the selection time of a target based on its distance and size. There are many variations of this model, such as [22, 23], that extend its usefulness beyond one dimensional pointing tasks. Of particular interest to this thesis is the finding by Guiard and Beaudouin-Lafon [24] that Fitts' Law still applies when navigating multi-scale worlds (such as map interfaces), where pointing at the target involves a series of pan and zoom operations as well as normal pointing. However, they show this is only true when the exact off-screen location of the target is known in advance and feedback is supplied throughout the user's navigation. When the target location is not known the user may revert to a simple visual search-like [38] behavior that is typically characterized by linearly scaling time (as opposed to the logarithmic scaling of Fitts' Law). An off-screen pointer provides the user with partial knowledge of the target's location, hypothetically resulting in search times characterized by a weighted average of linear and logarithmic functions.

That style of model was employed by Cockburn and colleagues [13, 15] in two related models that describe menu and hierarchical list item selection time. In Cockburn's models the weighted average increased influence of the logarithmic term as the user's familiarity with the menu/list contents increased with use.

In a similar vein, Cao et al. [10] investigated target selection time in a peephole pointing framework (where physical movement of the display causes the virtual world to move in the opposite direc-

tion, creating a portal-like effect). They came to the conclusion that completely directed movement (i.e., the target location is precisely known) results in a task completion time logarithmically related to the target constraints, and undirected (i.e., search-like behavior) navigation was related linearly.

There appears to be great possibility for creating a model of selection time in multi-scale worlds (especially maps) that incorporates the effectiveness of off-screen pointing techniques. With such a model, researchers could show the usefulness of specific off-screen pointers in real world map navigation tasks and more completely determine their benefits.

Beyond that, the research presented in this thesis opens up a number of other possibilities for future exploration:

- Completing a full predictive model of Wedge interpretability. This may lead to models for the interpretation of other off-screen pointing techniques.
- An optimized version of Wedge could be created using the model of interpretability. By selecting the smallest possible wedge sizes that still have good accuracy, overall performance should increase by improving the balance between accuracy and risk of overlap.
- Testing Wedge in a real world scenario might reveal difficulties with the technique not present in a lab setting.
- Chapter 3 lays out a good starting point for a taxonomy of off-screen visualization techniques that could lead to new techniques that address even more of the problems.
- Creating new interaction techniques, like Hop [31], that take full advantage of off-screen pointers by shortcutting common navigation tasks.
- Developing a more computationally efficient Wedge layout algorithm would also be a useful endeavour. It could be based on spring models, simulated annealing or possibly label placement algorithms from Geographical Information System (GIS) and augmented reality research.

6.2 FINAL WORDS

Off-screen pointing in general, and Wedge in particular, are simple techniques that have proved useful for improving the usability of map interfaces on devices with very small displays.

As more power and functionality come to our mobile devices, we will expect more from these small instruments and their tiny displays. The constrained physical form factor of these devices will become even more limiting as our expectations increase. Techniques like Wedge that make up for this limitation (in whatever small way) will only become increasingly important and I hope research in this area continues.

A

MATERIAL FROM EXPERIMENTS

A.1 HALO VS OVAL VS HALF ARC STUDY

Halo vs Oval vs Half Arc Pilot

Within-Subjects Factors

Measure:MEASURE_1

Viz	Dependent Variable
1	Compression
2	HalfArc
3	NoEffect

Descriptive Statistics

	Mean	Std. Deviation	N
Compression	334.8401	43.99794	5
HalfArc	223.9094	59.79769	5
NoEffect	162.8144	60.71734	5

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.958	34.090[a]	2.000	3.000	.009
	Wilks' Lambda	.042	34.090[a]	2.000	3.000	.009
	Hotelling's Trace	22.726	34.090[a]	2.000	3.000	.009
	Roy's Largest Root	22.726	34.090[a]	2.000	3.000	.009

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz

Pairwise Comparisons

Measure:MEASURE_1

		[a]			95% Confidence Interval for Difference[a]	
(I) Viz	(J) Viz	Mean Difference (I-J)	Std. Error	Sig.[a]	Lower Bound	Upper Bound
1	2	110.931*	14.782	.005	52.382	169.480
	3	172.026*	18.429	.002	99.032	245.019
2	1	-110.931*	14.782	.005	-169.480	-52.382
	3	61.095*	8.357	.006	27.996	94.194
3	1	-172.026*	18.429	.002	-245.019	-99.032
	2	-61.095*	8.357	.006	-94.194	-27.996

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

a. Adjustment for multiple comparisons: Bonferroni.

Raw statistics from SPSS for Halo vs Oval vs Half arc evaluation study. Above "No Effect" refers to Halo and "Compression" to Oval.

A.2 WEDGE EVALUATION STUDY

Locate Task – Means

Viz	Position	Density		Errpx	Tmems
Halo	Corner	Dense	Mean	69.3380	2421.2014
			Std. Deviation	19.80299	1355.93641
		Sparse	Mean	59.3819	2615.5625
			Std. Deviation	15.33213	1609.82159
		Total	Mean	64.3600	2518.3819
			Std. Deviation	18.28466	1481.02120
	Side	Dense	Mean	38.6545	2384.5544
			Std. Deviation	21.54293	1284.01634
		Sparse	Mean	31.4714	2219.1687
			Std. Deviation	17.68150	1024.21809
		Total	Mean	35.0630	2301.8615
			Std. Deviation	19.89919	1156.19925
	Total	Dense	Mean	53.9962	2402.8779
			Std. Deviation	25.70567	1311.26391
		Sparse	Mean	45.4267	2417.3656
			Std. Deviation	21.62152	1354.42672
		Total	Mean	49.7115	2410.1217
			Std. Deviation	24.05574	1328.37083
Wedge	Corner	Dense	Mean	44.3345	2270.9109
			Std. Deviation	22.34972	968.58626
		Sparse	Mean	40.7500	2216.2060
			Std. Deviation	20.27631	1013.74711
		Total	Mean	42.5422	2243.5584
			Std. Deviation	21.26414	984.80252
	Side	Dense	Mean	32.0757	2161.5309
			Std. Deviation	17.07284	784.59780
		Sparse	Mean	27.8187	2028.0692
			Std. Deviation	15.54502	731.88911
		Total	Mean	29.9472	2094.8000
			Std. Deviation	16.35251	756.33060
	Total	Dense	Mean	38.2051	2216.2209
			Std. Deviation	20.68875	876.90805
		Sparse	Mean	34.2843	2122.1376
			Std. Deviation	19.08362	882.97025
		Total	Mean	36.2447	2169.1792
			Std. Deviation	19.92999	878.13199

Raw statistics from SPSS for Wedge evaluation study.
Summary of means for Locate Task.

Locate Task - Error

Descriptive Statistics

	Mean	Std. Deviation	N
Errpx_mean.Halo.Corner.Dense	69.3380	19.80299	36
Errpx_mean.Halo.Corner.Sparse	59.3819	15.33213	36
Errpx_mean.Halo.Side.Dense	38.6545	21.54293	36
Errpx_mean.Halo.Side.Sparse	31.4714	17.68150	36
Errpx_mean.Wedges.Corner.Dense	44.3345	22.34972	36
Errpx_mean.Wedges.Corner.Sparse	40.7500	20.27631	36
Errpx_mean.Wedges.Side.Dense	32.0757	17.07284	36
Errpx_mean.Wedges.Side.Sparse	27.8187	15.54502	36

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.352	19.003[a]	1.000	35.000	.000
	Wilks' Lambda	.648	19.003[a]	1.000	35.000	.000
	Hotelling's Trace	.543	19.003[a]	1.000	35.000	.000
	Roy's Largest Root	.543	19.003[a]	1.000	35.000	.000
Position	Pillai's Trace	.795	1.358E2	1.000	35.000	.000
	Wilks' Lambda	.205	1.358E2	1.000	35.000	.000
	Hotelling's Trace	3.880	1.358E2	1.000	35.000	.000
	Roy's Largest Root	3.880	1.358E2	1.000	35.000	.000
Density	Pillai's Trace	.452	28.843[a]	1.000	35.000	.000
	Wilks' Lambda	.548	28.843[a]	1.000	35.000	.000
	Hotelling's Trace	.824	28.843[a]	1.000	35.000	.000
	Roy's Largest Root	.824	28.843[a]	1.000	35.000	.000
Viz * Position	Pillai's Trace	.538	40.745[a]	1.000	35.000	.000
	Wilks' Lambda	.462	40.745[a]	1.000	35.000	.000
	Hotelling's Trace	1.164	40.745[a]	1.000	35.000	.000
	Roy's Largest Root	1.164	40.745[a]	1.000	35.000	.000
Viz * Density	Pillai's Trace	.112	4.399[a]	1.000	35.000	.043
	Wilks' Lambda	.888	4.399[a]	1.000	35.000	.043
	Hotelling's Trace	.126	4.399[a]	1.000	35.000	.043
	Roy's Largest Root	.126	4.399[a]	1.000	35.000	.043
Position * Density	Pillai's Trace	.008	.284[a]	1.000	35.000	.598
	Wilks' Lambda	.992	.284[a]	1.000	35.000	.598
	Hotelling's Trace	.008	.284[a]	1.000	35.000	.598
	Roy's Largest Root	.008	.284[a]	1.000	35.000	.598
Viz * Position * Density	Pillai's Trace	.044	1.613[a]	1.000	35.000	.212
	Wilks' Lambda	.956	1.613[a]	1.000	35.000	.212
	Hotelling's Trace	.046	1.613[a]	1.000	35.000	.212
	Roy's Largest Root	.046	1.613[a]	1.000	35.000	.212

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz + Position + Density + Viz * Position + Viz * Density + Position * Density + Viz * Position * Density

Raw statistics from SPSS for Wedge evaluation study.
Locate Task, Error.

Locate Task - Time

Descriptive Statistics

	Mean	Std. Deviation	N
Tmems_mean.Halo.Corner.Dense	2421.2014	1355.93641	36
Tmems_mean.Halo.Corner.Sparse	2615.5625	1609.82159	36
Tmems_mean.Halo.Side.Dense	2384.5544	1284.01634	36
Tmems_mean.Halo.Side.Sparse	2219.1687	1024.21809	36
Tmems_mean.Wedges.Corner.Dense	2270.9109	968.58626	36
Tmems_mean.Wedges.Corner.Sparse	2216.2060	1013.74711	36
Tmems_mean.Wedges.Side.Dense	2161.5309	784.59780	36
Tmems_mean.Wedges.Side.Sparse	2028.0692	731.88911	36

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.084	3.222[a]	1.000	35.000	.081
	Wilks' Lambda	.916	3.222[a]	1.000	35.000	.081
	Hotelling's Trace	.092	3.222[a]	1.000	35.000	.081
	Roy's Largest Root	.092	3.222[a]	1.000	35.000	.081
Position	Pillai's Trace	.224	10.107[a]	1.000	35.000	.003
	Wilks' Lambda	.776	10.107[a]	1.000	35.000	.003
	Hotelling's Trace	.289	10.107[a]	1.000	35.000	.003
	Roy's Largest Root	.289	10.107[a]	1.000	35.000	.003
Density	Pillai's Trace	.018	.640[a]	1.000	35.000	.429
	Wilks' Lambda	.982	.640[a]	1.000	35.000	.429
	Hotelling's Trace	.018	.640[a]	1.000	35.000	.429
	Roy's Largest Root	.018	.640[a]	1.000	35.000	.429
Viz * Position	Pillai's Trace	.020	.717[a]	1.000	35.000	.403
	Wilks' Lambda	.980	.717[a]	1.000	35.000	.403
	Hotelling's Trace	.020	.717[a]	1.000	35.000	.403
	Roy's Largest Root	.020	.717[a]	1.000	35.000	.403
Viz * Density	Pillai's Trace	.045	1.636[a]	1.000	35.000	.209
	Wilks' Lambda	.955	1.636[a]	1.000	35.000	.209
	Hotelling's Trace	.047	1.636[a]	1.000	35.000	.209
	Roy's Largest Root	.047	1.636[a]	1.000	35.000	.209
Position * Density	Pillai's Trace	.115	4.540[a]	1.000	35.000	.040
	Wilks' Lambda	.885	4.540[a]	1.000	35.000	.040
	Hotelling's Trace	.130	4.540[a]	1.000	35.000	.040
	Roy's Largest Root	.130	4.540[a]	1.000	35.000	.040
Viz * Position * Density	Pillai's Trace	.096	3.696[a]	1.000	35.000	.063
	Wilks' Lambda	.904	3.696[a]	1.000	35.000	.063
	Hotelling's Trace	.106	3.696[a]	1.000	35.000	.063
	Roy's Largest Root	.106	3.696[a]	1.000	35.000	.063

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz + Position + Density + Viz * Position + Viz * Density + Position * Density + Viz * Position * Density

Raw statistics from SPSS for Wedge evaluation study.
Locate Task, Completion Time.

Avoid Task – Means

Viz	Position	Density		Time	ErrorRate
Halo	Corner	Dense	Mean	4342.8785	30.9028
			Std. Deviation	2440.43650	17.29113
		Sparse	Mean	4158.6528	37.1528
			Std. Deviation	2168.11711	18.29474
		Total	Mean	4250.7656	34.0278
			Std. Deviation	2293.85915	17.95218
	Side	Dense	Mean	4131.8958	31.5972
			Std. Deviation	2167.35839	17.80001
		Sparse	Mean	4023.7049	32.2917
			Std. Deviation	2064.14871	24.34407
		Total	Mean	4077.8003	31.9444
			Std. Deviation	2102.13180	21.17675
	Total	Dense	Mean	4237.3872	31.2500
			Std. Deviation	2294.09005	17.42691
		Sparse	Mean	4091.1788	34.7222
			Std. Deviation	2102.90964	21.52036
		Total	Mean	4164.2830	32.9861
			Std. Deviation	2194.09588	19.58993
Wedges	Corner	Dense	Mean	4085.7188	27.7778
			Std. Deviation	1824.51447	19.15890
		Sparse	Mean	4166.1875	35.7639
			Std. Deviation	1816.14033	24.84825
		Total	Mean	4125.9531	31.7708
			Std. Deviation	1807.92158	22.39387
	Side	Dense	Mean	4236.9688	30.9028
			Std. Deviation	2111.86787	17.29113
		Sparse	Mean	4318.5208	31.5972
			Std. Deviation	2167.63713	28.58042
		Total	Mean	4277.7448	31.2500
			Std. Deviation	2125.20751	23.45583
	Total	Dense	Mean	4161.3438	29.3403
			Std. Deviation	1960.96104	18.18815
		Sparse	Mean	4242.3542	33.6806
			Std. Deviation	1986.97616	26.67281
		Total	Mean	4201.8490	31.5104
			Std. Deviation	1967.51711	22.85218

Raw statistics from SPSS for Wedge evaluation study.
Summary of means for Avoid Task.

Avoid Task - Percent Correct

Descriptive Statistics

	Mean	Std. Deviation	N
CorrectYN_mean.Halo.Corner.Dense	.6910	.17291	36
CorrectYN_mean.Halo.Corner.Sparse	.6285	.18295	36
CorrectYN_mean.Halo.Side.Dense	.6840	.17800	36
CorrectYN_mean.Halo.Side.Sparse	.6771	.24344	36
CorrectYN_mean.Wedges.Corner.Dense	.7222	.19159	36
CorrectYN_mean.Wedges.Corner.Sparse	.6424	.24848	36
CorrectYN_mean.Wedges.Side.Dense	.6910	.17291	36
CorrectYN_mean.Wedges.Side.Sparse	.6840	.28580	36

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.015	.541[a]	1.000	35.000	.467
	Wilks' Lambda	.985	.541[a]	1.000	35.000	.467
	Hotelling's Trace	.015	.541[a]	1.000	35.000	.467
	Roy's Largest Root	.015	.541[a]	1.000	35.000	.467
Position	Pillai's Trace	.020	.699[a]	1.000	35.000	.409
	Wilks' Lambda	.980	.699[a]	1.000	35.000	.409
	Hotelling's Trace	.020	.699[a]	1.000	35.000	.409
	Roy's Largest Root	.020	.699[a]	1.000	35.000	.409
Density	Pillai's Trace	.095	3.654[a]	1.000	35.000	.064
	Wilks' Lambda	.905	3.654[a]	1.000	35.000	.064
	Hotelling's Trace	.104	3.654[a]	1.000	35.000	.064
	Roy's Largest Root	.104	3.654[a]	1.000	35.000	.064
Viz * Position	Pillai's Trace	.004	.157[a]	1.000	35.000	.694
	Wilks' Lambda	.996	.157[a]	1.000	35.000	.694
	Hotelling's Trace	.004	.157[a]	1.000	35.000	.694
	Roy's Largest Root	.004	.157[a]	1.000	35.000	.694
Viz * Density	Pillai's Trace	.002	.058[a]	1.000	35.000	.811
	Wilks' Lambda	.998	.058[a]	1.000	35.000	.811
	Hotelling's Trace	.002	.058[a]	1.000	35.000	.811
	Roy's Largest Root	.002	.058[a]	1.000	35.000	.811
Position * Density	Pillai's Trace	.112	4.422[a]	1.000	35.000	.043
	Wilks' Lambda	.888	4.422[a]	1.000	35.000	.043
	Hotelling's Trace	.126	4.422[a]	1.000	35.000	.043
	Roy's Largest Root	.126	4.422[a]	1.000	35.000	.043
Viz * Position * Density	Pillai's Trace	.001	.050[a]	1.000	35.000	.825
	Wilks' Lambda	.999	.050[a]	1.000	35.000	.825
	Hotelling's Trace	.001	.050[a]	1.000	35.000	.825
	Roy's Largest Root	.001	.050[a]	1.000	35.000	.825

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz + Position + Density + Viz * Position + Viz * Density + Position * Density + Viz * Position * Density

Raw statistics from SPSS for Wedge evaluation study.
Avoid Task, Percent Correct.

Avoid Task - Time

Descriptive Statistics

	Mean	Std. Deviation	N
Time_mean.Halo.Corner.Dense	4342.8785	2440.43650	36
Time_mean.Halo.Corner.Sparse	4158.6528	2168.11711	36
Time_mean.Halo.Side.Dense	4131.8958	2167.35839	36
Time_mean.Halo.Side.Sparse	4023.7049	2064.14871	36
Time_mean.Wedges.Corner.Dense	4085.7188	1824.51447	36
Time_mean.Wedges.Corner.Sparse	4166.1875	1816.14033	36
Time_mean.Wedges.Side.Dense	4236.9688	2111.86787	36
Time_mean.Wedges.Side.Sparse	4318.5208	2167.63713	36

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.001	.037[a]	1.000	35.000	.848
	Wilks' Lambda	.999	.037[a]	1.000	35.000	.848
	Hotelling's Trace	.001	.037[a]	1.000	35.000	.848
	Roy's Largest Root	.001	.037[a]	1.000	35.000	.848
Position	Pillai's Trace	.000	.007[a]	1.000	35.000	.932
	Wilks' Lambda	1.000	.007[a]	1.000	35.000	.932
	Hotelling's Trace	.000	.007[a]	1.000	35.000	.932
	Roy's Largest Root	.000	.007[a]	1.000	35.000	.932
Density	Pillai's Trace	.002	.085[a]	1.000	35.000	.772
	Wilks' Lambda	.998	.085[a]	1.000	35.000	.772
	Hotelling's Trace	.002	.085[a]	1.000	35.000	.772
	Roy's Largest Root	.002	.085[a]	1.000	35.000	.772
Viz * Position	Pillai's Trace	.027	.988[a]	1.000	35.000	.327
	Wilks' Lambda	.973	.988[a]	1.000	35.000	.327
	Hotelling's Trace	.028	.988[a]	1.000	35.000	.327
	Roy's Largest Root	.028	.988[a]	1.000	35.000	.327
Viz * Density	Pillai's Trace	.035	1.284[a]	1.000	35.000	.265
	Wilks' Lambda	.965	1.284[a]	1.000	35.000	.265
	Hotelling's Trace	.037	1.284[a]	1.000	35.000	.265
	Roy's Largest Root	.037	1.284[a]	1.000	35.000	.265
Position * Density	Pillai's Trace	.002	.063[a]	1.000	35.000	.804
	Wilks' Lambda	.998	.063[a]	1.000	35.000	.804
	Hotelling's Trace	.002	.063[a]	1.000	35.000	.804
	Roy's Largest Root	.002	.063[a]	1.000	35.000	.804
Viz * Position * Density	Pillai's Trace	.001	.026[a]	1.000	35.000	.873
	Wilks' Lambda	.999	.026[a]	1.000	35.000	.873
	Hotelling's Trace	.001	.026[a]	1.000	35.000	.873
	Roy's Largest Root	.001	.026[a]	1.000	35.000	.873

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz + Position + Density + Viz * Position + Viz * Density + Position * Density + Viz * Position * Density

Raw statistics from SPSS for Wedge evaluation study.
Avoid Task, Completion Time.

Closest Task – Means

Viz	Position	Density		Time	ErrorRate
Halo	Corner	Dense	Mean	4711.9479	49.3056
			Std. Deviation	3626.60539	15.21995
		Sparse	Mean	4816.0729	60.0694
			Std. Deviation	4382.54914	20.44580
		Total	Mean	4764.0104	54.6875
			Std. Deviation	3994.29220	18.69858
	Side	Dense	Mean	3603.8819	34.3750
			Std. Deviation	2375.51120	21.83031
		Sparse	Mean	4122.2917	27.0833
			Std. Deviation	3240.29371	17.80349
		Total	Mean	3863.0868	30.7292
			Std. Deviation	2832.97227	20.11603
	Total	Dense	Mean	4157.9149	41.8403
			Std. Deviation	3094.60409	20.14032
		Sparse	Mean	4469.1823	43.5764
			Std. Deviation	3842.64872	25.26210
		Total	Mean	4313.5486	42.7083
			Std. Deviation	3480.02002	22.78184
Wedges	Corner	Dense	Mean	5067.2569	44.0972
			Std. Deviation	3017.45733	13.85064
		Sparse	Mean	5532.5590	50.6944
			Std. Deviation	4158.05535	23.13478
		Total	Mean	5299.9080	47.3958
			Std. Deviation	3614.72666	19.22091
	Side	Dense	Mean	4674.7569	29.8611
			Std. Deviation	2910.22321	17.74768
		Sparse	Mean	4465.6667	20.8333
			Std. Deviation	2957.46909	15.52648
		Total	Mean	4570.2118	25.3472
			Std. Deviation	2915.10811	17.16893
	Total	Dense	Mean	4871.0069	36.9792
			Std. Deviation	2950.00284	17.35573
		Sparse	Mean	4999.1128	35.7639
			Std. Deviation	3622.60185	24.67264
		Total	Mean	4935.0599	36.3715
			Std. Deviation	3292.52154	21.26430

Raw statistics from SPSS for Wedge evaluation study.
Summary of means for Closest Task.

Closest Task - Percent Correct

Descriptive Statistics

	Mean	Std. Deviation	N
CorrectYN_mean.Halo.Corner.Dense	.5069	.15220	36
CorrectYN_mean.Halo.Corner.Sparse	.3993	.20446	36
CorrectYN_mean.Halo.Side.Dense	.6563	.21830	36
CorrectYN_mean.Halo.Side.Sparse	.7292	.17803	36
CorrectYN_mean.Wedges.Corner.Dense	.5590	.13851	36
CorrectYN_mean.Wedges.Corner.Sparse	.4931	.23135	36
CorrectYN_mean.Wedges.Side.Dense	.7014	.17748	36
CorrectYN_mean.Wedges.Side.Sparse	.7917	.15526	36

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.120	4.793[a]	1.000	35.000	.035
	Wilks' Lambda	.880	4.793[a]	1.000	35.000	.035
	Hotelling's Trace	.137	4.793[a]	1.000	35.000	.035
	Roy's Largest Root	.137	4.793[a]	1.000	35.000	.035
Position	Pillai's Trace	.854	2.042E2	1.000	35.000	.000
	Wilks' Lambda	.146	2.042E2	1.000	35.000	.000
	Hotelling's Trace	5.835	2.042E2	1.000	35.000	.000
	Roy's Largest Root	5.835	2.042E2	1.000	35.000	.000
Density	Pillai's Trace	.001	.019[a]	1.000	35.000	.890
	Wilks' Lambda	.999	.019[a]	1.000	35.000	.890
	Hotelling's Trace	.001	.019[a]	1.000	35.000	.890
	Roy's Largest Root	.001	.019[a]	1.000	35.000	.890
Viz * Position	Pillai's Trace	.009	.312[a]	1.000	35.000	.580
	Wilks' Lambda	.991	.312[a]	1.000	35.000	.580
	Hotelling's Trace	.009	.312[a]	1.000	35.000	.580
	Roy's Largest Root	.009	.312[a]	1.000	35.000	.580
Viz * Density	Pillai's Trace	.016	.570[a]	1.000	35.000	.455
	Wilks' Lambda	.984	.570[a]	1.000	35.000	.455
	Hotelling's Trace	.016	.570[a]	1.000	35.000	.455
	Roy's Largest Root	.016	.570[a]	1.000	35.000	.455
Position * Density	Pillai's Trace	.567	45.821[a]	1.000	35.000	.000
	Wilks' Lambda	.433	45.821[a]	1.000	35.000	.000
	Hotelling's Trace	1.309	45.821[a]	1.000	35.000	.000
	Roy's Largest Root	1.309	45.821[a]	1.000	35.000	.000
Viz * Position * Density	Pillai's Trace	.006	.225[a]	1.000	35.000	.638
	Wilks' Lambda	.994	.225[a]	1.000	35.000	.638
	Hotelling's Trace	.006	.225[a]	1.000	35.000	.638
	Roy's Largest Root	.006	.225[a]	1.000	35.000	.638

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz + Position + Density + Viz * Position + Viz * Density + Position * Density + Viz * Position * Density

Raw statistics from SPSS for Wedge evaluation study.
Closest Task, Percent Correct.

Closest Task - Time

Descriptive Statistics

	Mean	Std. Deviation	N
Time_mean.Halo.Corner.Dense	4711.9479	3626.60539	36
Time_mean.Halo.Corner.Sparse	4816.0729	4382.54914	36
Time_mean.Halo.Side.Dense	3603.8819	2375.51120	36
Time_mean.Halo.Side.Sparse	4122.2917	3240.29371	36
Time_mean.Wedges.Corner.Dense	5067.2569	3017.45733	36
Time_mean.Wedges.Corner.Sparse	5532.5590	4158.05535	36
Time_mean.Wedges.Side.Dense	4674.7569	2910.22321	36
Time_mean.Wedges.Side.Sparse	4465.6667	2957.46909	36

Multivariate Tests[b]

Effect		Value	F	Hypothesis df	Error df	Sig.
Viz	Pillai's Trace	.175	7.399[a]	1.000	35.000	.010
	Wilks' Lambda	.825	7.399[a]	1.000	35.000	.010
	Hotelling's Trace	.211	7.399[a]	1.000	35.000	.010
	Roy's Largest Root	.211	7.399[a]	1.000	35.000	.010
Position	Pillai's Trace	.289	14.231[a]	1.000	35.000	.001
	Wilks' Lambda	.711	14.231[a]	1.000	35.000	.001
	Hotelling's Trace	.407	14.231[a]	1.000	35.000	.001
	Roy's Largest Root	.407	14.231[a]	1.000	35.000	.001
Density	Pillai's Trace	.048	1.764[a]	1.000	35.000	.193
	Wilks' Lambda	.952	1.764[a]	1.000	35.000	.193
	Hotelling's Trace	.050	1.764[a]	1.000	35.000	.193
	Roy's Largest Root	.050	1.764[a]	1.000	35.000	.193
Viz * Position	Pillai's Trace	.009	.304[a]	1.000	35.000	.585
	Wilks' Lambda	.991	.304[a]	1.000	35.000	.585
	Hotelling's Trace	.009	.304[a]	1.000	35.000	.585
	Roy's Largest Root	.009	.304[a]	1.000	35.000	.585
Viz * Density	Pillai's Trace	.035	1.282[a]	1.000	35.000	.265
	Wilks' Lambda	.965	1.282[a]	1.000	35.000	.265
	Hotelling's Trace	.037	1.282[a]	1.000	35.000	.265
	Roy's Largest Root	.037	1.282[a]	1.000	35.000	.265
Position * Density	Pillai's Trace	.016	.556[a]	1.000	35.000	.461
	Wilks' Lambda	.984	.556[a]	1.000	35.000	.461
	Hotelling's Trace	.016	.556[a]	1.000	35.000	.461
	Roy's Largest Root	.016	.556[a]	1.000	35.000	.461
Viz * Position * Density	Pillai's Trace	.087	3.332[a]	1.000	35.000	.076
	Wilks' Lambda	.913	3.332[a]	1.000	35.000	.076
	Hotelling's Trace	.095	3.332[a]	1.000	35.000	.076
	Roy's Largest Root	.095	3.332[a]	1.000	35.000	.076

a. Exact statistic

b. Design: Intercept
Within Subjects Design: Viz + Position + Density + Viz * Position + Viz * Density + Position * Density + Viz * Position * Density

Raw statistics from SPSS for Wedge evaluation study – Closest Task, Completion Time.

Chi-Square Test

Locate

	Observed N	Expected N	Residual
Wedge	25	17.5	7.5
Halo	10	17.5	-7.5
Total	35		

Avoid

	Observed N	Expected N	Residual
Wedge	21	17.0	4.0
Halo	13	17.0	-4.0
Total	34		

Closest

	Observed N	Expected N	Residual
Wedge	18	16.0	2.0
Halo	14	16.0	-2.0
Total	32		

Test Statistics

	Locate	Avoid	Closest
Chi-Square	6.429[a]	1.882[b]	.500[c]
Df	1	1	1
Asymp. Sig.	.011	.170	.480

a. 0 cells (.0%) have expected frequencies less than 5. The minimum expected cell frequency is 17.5.

b. 0 cells (.0%) have expected frequencies less than 5. The minimum expected cell frequency is 17.0.

c. 0 cells (.0%) have expected frequencies less than 5. The minimum expected cell frequency is 16.0.

Raw statistics from SPSS for Wedge evaluation study.
User preferences for each task.

_____

Off-screen Target Study

Age: **under 21** **21-25** **26-30** **over 30** (circle one)

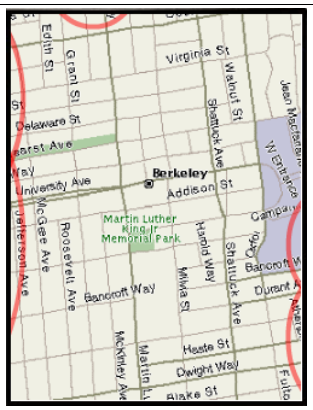
Sex: **Male** **Female** (circle one)

Which is your dominant hand (the hand you write with)? **Left** **Right** (circle one)

Do you have any visual impairments (including colorblindness)? **Yes** **No** (circle one)

If so, what? _____

The following questions correspond to the "Locate" portion of the task. Where you had to click off-screen where you thought each item of located.



During this task, which technique did you prefer? (circle)

Rings Wedges No preference

Please write down any comments you may have regarding the Locate task:

_____

_____

_____

_____

_____

_____

_____

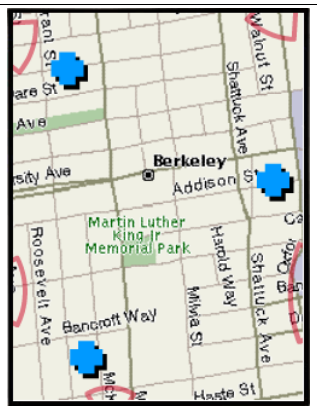
_____

_____

Questionnaire from Wedge evaluation study. Page 1 of 2.

_____

The following questions correspond to the "Avoid" portion of the task. Where you were an ambulance driver needing to drop a patient off at the hospital (cross) that is farthest away from any accidents or constructions zones (flash icons or rings/wedges).

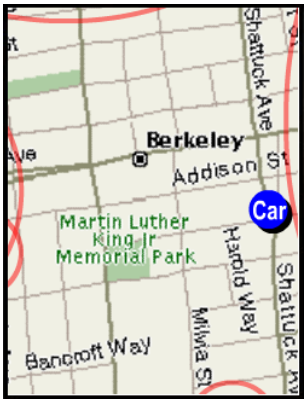


During this task, which technique did you prefer? (circle)

Rings **Wedges** **No preference**

Please write down any comments you may have regarding the Avoid task:

The following questions correspond to the "Closest" portion of the task. Where you were asked to click the ring/wedge representing the off-screen location closest to the car.



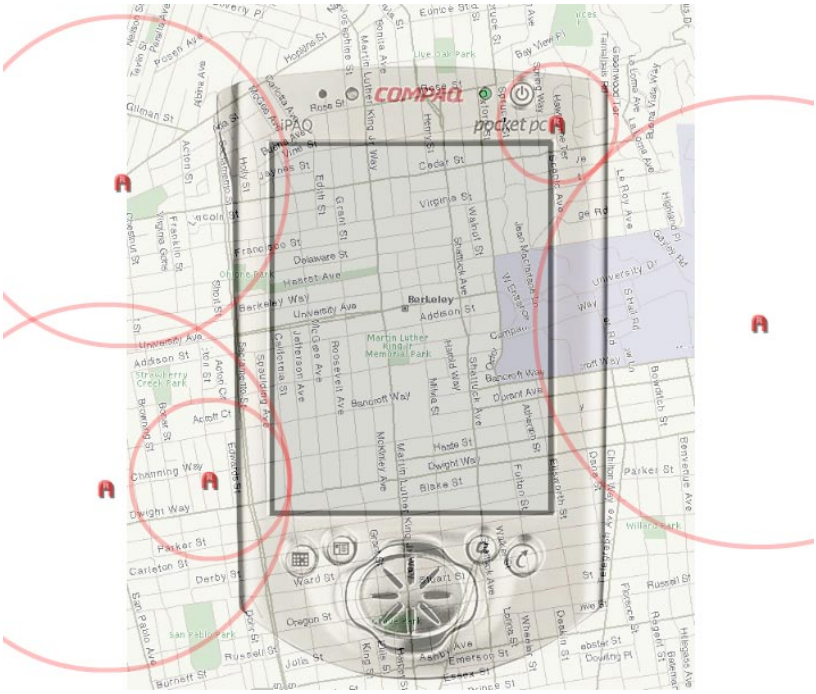
During this task, which technique did you prefer? (circle)

Rings **Wedges** **No preference**

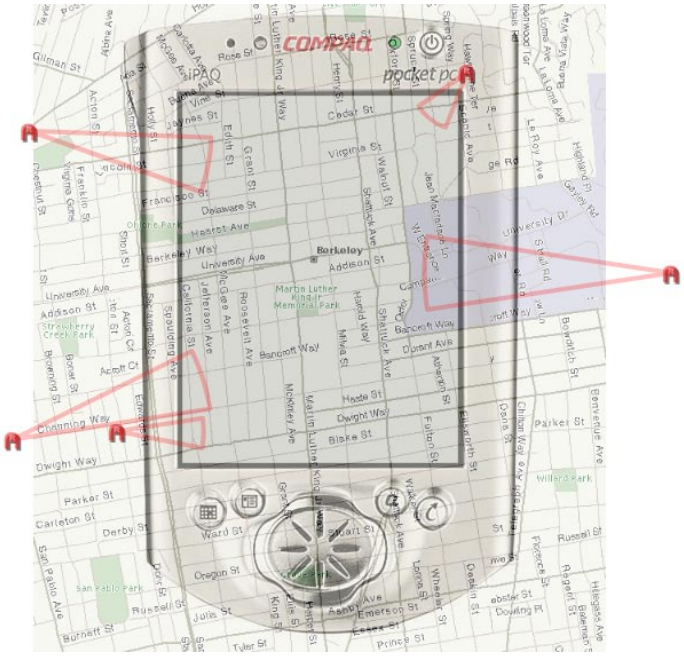
Please write down any comments you may have regarding the Closest task:

Questionnaire from Wedge evaluation study. Page 2 of 2.

Rings



Training material given during Wedge evaluation study. Page 1 of 2.



Training material given during Wedge evaluation study. Page 2 of 2.

Wedges Study – Oct 2007
Page 1/1

Participant Information

Date / Time: _____

Participant Number: _____ Participant Name: _____

Informed Consent Agreement
Please read this consent agreement carefully before you decide to participate in the study.

Purpose of the research study: The purpose of the study is to evaluate features of a PDA-based map application.

What you will do in the study: You will interact with a simulated PDA on a desktop personal computer using a mapping application.

Time required: The study will require about 30-60 minutes of your time.

Risks: There are no anticipated risks in this study.

Benefits: You will receive a 2% mark in COMP1260 in exchange for your participation.

Confidentiality: The information that you give in the study will be handled confidentially. It will be viewable only by researchers working on this project, which may involve faculty members and graduate research assistants.

Voluntary participation: Your participation in the study is completely voluntary.

Right to withdraw from the study: You have the right to withdraw from the study at any time without penalty.

How to withdraw from the study: If you want to withdraw from the study, please inform the experimenter and leave the room. There is no penalty for withdrawing. You will still receive full credit for the study. If you would like to withdraw after your materials have been submitted, please contact one of the researchers listed below.

If you have questions about the study, contact:
Sean Gustafson
Department of Computer Science, E2-445 EITC
University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada
Phone: +1 (204) 474-8313

Faculty Advisor: Dr. Pourang Irani
Department of Computer Science, E2-580 EITC
University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada
Phone: +1 (204) 474-8995

Agreement:
I agree to participate in the research study described above.

Signature: _____ Date: _____

Revision Date: Oct 8, 2007

Consent form from the Wedge evaluation study.

A.3 WEDGE MODEL STUDY

orbitalBias orbitalLength orbitalWidth * distance * baseLength

distance	baseLength		orbitalBias	orbitalLength	orbitalWidth
50	25	Mean	-33.7135	23.2950	4.4440
		Std. Deviation	44.78907	24.46630	4.42403
	50	Mean	-19.5554	16.7130	3.2775
		Std. Deviation	26.44677	16.79084	2.78338
	100	Mean	-13.9651	12.9970	4.3473
		Std. Deviation	26.60330	13.90458	4.62890
	200	Mean	-21.2361	17.2430	6.4935
		Std. Deviation	44.22616	19.97530	5.14446
	Total	Mean	-22.1175	17.5620	4.6406
		Std. Deviation	37.18195	19.45754	4.47130
100	25	Mean	-35.2843	34.8942	6.2876
		Std. Deviation	43.06919	23.38810	4.84078
	50	Mean	-28.1571	26.8409	5.4422
		Std. Deviation	38.97869	22.54281	4.08806
	100	Mean	-7.1720	20.6557	5.5799
		Std. Deviation	22.30528	14.84928	4.37008
	200	Mean	-5.6108	19.0899	7.6893
		Std. Deviation	31.51443	18.43276	6.37531
	Total	Mean	-19.0560	25.3702	6.2498
		Std. Deviation	37.05266	20.95221	5.05413
200	25	Mean	48.0012	36.1285	5.7130
		Std. Deviation	50.25033	21.65167	5.31920
	50	Mean	43.3601	40.0275	5.8082
		Std. Deviation	43.59795	24.36577	3.88851
	100	Mean	47.8518	30.8537	6.6107
		Std. Deviation	41.39767	21.55264	4.94224
	200	Mean	35.8287	25.2884	8.0758
		Std. Deviation	31.45966	18.53330	8.64321
	Total	Mean	43.7605	33.0745	6.5519
		Std. Deviation	42.31902	22.23358	6.01793
300	25	Mean	120.8669	39.7005	7.4226
		Std. Deviation	59.18107	25.16720	5.35822
	50	Mean	127.1839	40.3892	6.3335
		Std. Deviation	45.63397	26.48162	3.47755
	100	Mean	125.7222	37.0517	9.2272
		Std. Deviation	49.17359	27.12730	5.98861
	200	Mean	95.9331	30.2258	8.9091
		Std. Deviation	44.14485	18.31042	5.16436
	Total	Mean	117.4265	36.8418	7.9731
		Std. Deviation	51.24032	24.74392	5.19300
Total	25	Mean	24.9676	33.5045	5.9668
		Std. Deviation	81.62583	24.39032	5.09248
	50	Mean	30.7078	30.9926	5.2154
		Std. Deviation	73.57277	24.79234	3.76281
	100	Mean	38.1092	25.3895	6.4413
		Std. Deviation	66.82022	22.03780	5.31192
	200	Mean	26.2287	22.9618	7.7919
		Std. Deviation	59.32308	19.43598	6.51941
	Total	Mean	30.0033	28.2121	6.3538
		Std. Deviation	70.92421	23.12826	5.34060

Means from SPSS for Wedge modeling study. Page 1 of 2.

orbitalBias orbitalLength orbitalWidth * distance * legLength

distance	legLength		orbitalBias	orbitalLength	orbitalWidth
50	4	Mean	-42.7245	25.5124	6.0840
		Std. Deviation	51.43170	21.91876	5.10870
	8	Mean	-23.9951	18.0403	5.5855
		Std. Deviation	32.54654	17.92235	4.11026
	16	Mean	-13.8563	13.8778	4.0878
		Std. Deviation	28.97599	15.59363	4.58192
	32	Mean	-7.8943	12.8175	2.8049
		Std. Deviation	18.13058	19.55870	3.15812
	Total	Mean	-22.1175	17.5620	4.6406
		Std. Deviation	37.18195	19.45754	4.47130
100	4	Mean	-14.7091	32.3419	6.9402
		Std. Deviation	38.49392	25.40955	5.55096
	8	Mean	-26.9111	24.3366	6.8962
		Std. Deviation	41.46443	19.01387	5.71924
	16	Mean	-19.3666	24.5081	6.2691
		Std. Deviation	35.44178	19.69123	4.80909
	32	Mean	-15.2374	20.2941	4.8935
		Std. Deviation	31.42607	17.31560	3.69901
	Total	Mean	-19.0560	25.3702	6.2498
		Std. Deviation	37.05266	20.95221	5.05413
200	4	Mean	58.8812	35.7245	7.8625
		Std. Deviation	45.26874	23.45969	8.89303
	8	Mean	57.8970	30.6459	5.3427
		Std. Deviation	32.70544	19.84202	3.97191
	16	Mean	37.6536	31.3765	6.2627
		Std. Deviation	37.40216	20.51842	4.43199
	32	Mean	20.6100	34.5511	6.7397
		Std. Deviation	41.19126	24.73713	5.32237
	Total	Mean	43.7605	33.0745	6.5519
		Std. Deviation	42.31902	22.23358	6.01793
300	4	Mean	129.7223	37.9615	7.5244
		Std. Deviation	57.28039	28.33482	5.17738
	8	Mean	130.5292	38.8360	8.1135
		Std. Deviation	49.23681	27.27479	5.52805
	16	Mean	112.9655	36.7893	8.4933
		Std. Deviation	42.37457	20.24746	4.74042
	32	Mean	96.4892	33.7806	7.7613
		Std. Deviation	48.01792	22.35307	5.33521
	Total	Mean	117.4265	36.8418	7.9731
		Std. Deviation	51.24032	24.74392	5.19300
Total	4	Mean	32.7925	32.8851	7.1028
		Std. Deviation	82.85781	25.22817	6.38690
	8	Mean	34.3800	27.9647	6.4845
		Std. Deviation	76.20518	22.59310	5.00075
	16	Mean	29.3491	26.6379	6.2782
		Std. Deviation	64.37623	20.87173	4.87844
	32	Mean	23.4919	25.3608	5.5499
		Std. Deviation	57.26350	23.00686	4.84877
	Total	Mean	30.0033	28.2121	6.3538
		Std. Deviation	70.92421	23.12826	5.34060

Means from SPSS for Wedge modeling study. Page 2 of 2.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.788[a]	.622	.621	43.64892
2	.788[b]	.622	.621	43.66500
3	.790[c]	.625	.624	43.49486

a. Predictors: (Constant), distance

b. Predictors: (Constant), distance, baseLength

c. Predictors: (Constant), distance, baseLength, legLength

ANOVA[d]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	4198801.744	1	4198801.744	2203.831	.000[a]
	Residual	2556816.292	1342	1905.228		
	Total	6755618.035	1343			
2	Regression	4198824.658	2	2099412.329	1101.110	.000[b]
	Residual	2556793.378	1341	1906.632		
	Total	6755618.035	1343			
3	Regression	4220601.669	3	1406867.223	743.665	.000[c]
	Residual	2535016.367	1340	1891.803		
	Total	6755618.035	1343			

a. Predictors: (Constant), distance

b. Predictors: (Constant), distance, baseLength

c. Predictors: (Constant), distance, baseLength, legLength

d. Dependent Variable: orbitalBias

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	-64.594	2.341		-27.598	.000
	distance	.582	.012	.788	46.945	.000
2	(Constant)	-64.412	2.874		-22.415	.000
	distance	.582	.012	.788	46.928	.000
	baseLength	-.002	.018	-.002	-.110	.913
3	(Constant)	-58.781	3.309		-17.766	.000
	distance	.582	.012	.788	47.111	.000
	baseLength	-.002	.018	-.002	-.110	.912
	legLength	-.375	.111	-.057	-3.393	.001

a. Dependent Variable: orbitalBias

Excluded Variables[c]

Model						Collinearity Statistics
		Beta In	t	Sig.	Partial Correlation	Tolerance
1	baseLength	-.002[a]	-.110	.913	-.003	1.000
	legLength	-.057[a]	-3.394	.001	-.092	1.000
2	legLength	-.057[b]	-3.393	.001	-.092	1.000

a. Predictors in the Model: (Constant), distance

b. Predictors in the Model: (Constant), distance, baseLength

c. Dependent Variable: orbitalBias

Bias regression results from SPSS for Wedge modeling study.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.309[a]	.096	.095	22.00377
2	.353[b]	.125	.124	21.65248
3	.367[c]	.135	.133	21.53508

a. Predictors: (Constant), distance

b. Predictors: (Constant), distance, baseLength

c. Predictors: (Constant), distance, baseLength, legLength

ANOVA[d]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	68642.295	1	68642.295	141.774	.000[a]
	Residual	649750.652	1342	484.166		
	Total	718392.947	1343			
2	Regression	89691.905	2	44845.953	95.655	.000[b]
	Residual	628701.041	1341	468.830		
	Total	718392.947	1343			
3	Regression	96954.974	3	32318.325	69.688	.000[c]
	Residual	621437.973	1340	463.760		
	Total	718392.947	1343			

a. Predictors: (Constant), distance

b. Predictors: (Constant), distance, baseLength

c. Predictors: (Constant), distance, baseLength, legLength

d. Dependent Variable: orbitalLength

Coefficients[a]

		Unstandardized Coefficients		Standardized Coefficients		
Model		B	Std. Error	Beta	t	Sig.
1	(Constant)	16.117	1.180		13.660	.000
	distance	.074	.006	.309	11.907	.000
2	(Constant)	21.653	1.425		15.195	.000
	distance	.074	.006	.309	12.100	.000
	baseLength	-.059	.009	-.171	-6.701	.000
3	(Constant)	24.904	1.638		15.202	.000
	distance	.074	.006	.309	12.166	.000
	baseLength	-.059	.009	-.171	-6.737	.000
	legLength	-.217	.055	-.101	-3.957	.000

a. Dependent Variable: orbitalLength

Excluded Variables[c]

						Collinearity Statistics
Model		Beta In	t	Sig.	Partial Correlation	Tolerance
1	baseLength	-.171[a]	-6.701	.000	-.180	1.000
	legLength	-.101[a]	-3.894	.000	-.106	1.000
2	legLength	-.101[b]	-3.957	.000	-.107	1.000

a. Predictors in the Model: (Constant), distance

b. Predictors in the Model: (Constant), distance, baseLength

c. Dependent Variable: orbitalLength

Orbital length regression results from SPSS for Wedge modeling study.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.209[a]	.044	.043	5.22415
2	.264[b]	.070	.068	5.15519
3	.282[c]	.080	.078	5.12934

a. Predictors: (Constant), distance

b. Predictors: (Constant), distance, baseLength

c. Predictors: (Constant), distance, baseLength, legLength

ANOVA[d]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	1679.482	1	1679.482	61.538	.000[a]
	Residual	36625.570	1342	27.292		
	Total	38305.052	1343			
2	Regression	2666.599	2	1333.300	50.169	.000[b]
	Residual	35638.453	1341	26.576		
	Total	38305.052	1343			
3	Regression	3049.527	3	1016.509	38.636	.000[c]
	Residual	35255.525	1340	26.310		
	Total	38305.052	1343			

a. Predictors: (Constant), distance

b. Predictors: (Constant), distance, baseLength

c. Predictors: (Constant), distance, baseLength, legLength

d. Dependent Variable: orbitalWidth

Coefficients[a]

		Unstandardized Coefficients		Standardized Coefficients		
Model		B	Std. Error	Beta	t	Sig.
1	(Constant)	4.462	.280		15.928	.000
	distance	.012	.001	.209	7.845	.000
2	(Constant)	3.263	.339		9.618	.000
	distance	.012	.001	.209	7.950	.000
	baseLength	.013	.002	.161	6.095	.000
3	(Constant)	4.010	.390		10.276	.000
	distance	.012	.001	.209	7.990	.000
	baseLength	.013	.002	.161	6.125	.000
	legLength	-.050	.013	-.100	-3.815	.000

a. Dependent Variable: orbitalWidth

Excluded Variables[c]

						Collinearity Statistics
Model		Beta In	t	Sig.	Partial Correlation	Tolerance
1	baseLength	.161[a]	6.095	.000	.164	1.000
	legLength	-.100[a]	-3.764	.000	-.102	1.000
2	legLength	-.100[b]	-3.815	.000	-.104	1.000

a. Predictors in the Model: (Constant), distance

b. Predictors in the Model: (Constant), distance, baseLength

c. Dependent Variable: orbitalWidth

Orbital width regression results from SPSS for Wedge modeling study.

Wedge Modeling Study – Oct 9-10 / 2008
Page 1/2

Participant Information

Date / Time: _____

Participant Number: _____ Participant Name: _____

Informed Consent Agreement
Please read this consent agreement carefully before you decide to participate in the study.

Purpose of the research study: The purpose of the study is to gather exploratory data regarding people's ability to perceive partially obscured shapes.

What you will do in the study: You will interact with an experimental application running on a standard desktop computer. You will be shown a portion of a large triangle and you will be asked to click on the screen where you believe is the opposing point of the triangle. You will perform this task repeatedly with several short breaks.

Time required: The study will require about 30-60 minutes of your time.

Risks: There are no anticipated risks in this study.

Benefits: You will receive a 2% participation mark in COMP3020 (up to a maximum of 5% per term) in exchange for your participation.

Confidentiality: The information that you give in the study will be handled confidentially. It will be viewable only by researchers working on this project, which may involve faculty members and graduate research assistants.

Voluntary participation: Your participation in the study is completely voluntary.

Right to withdraw from the study: You have the right to withdraw from the study at any time without penalty.

How to withdraw from the study: If you want to withdraw from the study, please inform the experimenter and leave the room. There is no penalty for withdrawing. You will still receive full credit for the study. If you would like to withdraw after your materials have been submitted, please contact one of the researchers listed below.

If you have questions about the study, contact:
Sean Gustafson
Department of Computer Science, E2-445 EITC
University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada
Phone: +1 (204) 474-8313

Faculty Advisor: Dr. Pourang Irani
Department of Computer Science, E2-580 EITC
University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada
Phone: +1 (204) 474-8995

Agreement:
I agree to participate in the research study described above.

Signature: _____ Date: _____

(flip over)

Revision Date: Oct 9, 2008

Consent form from the Wedge modeling study. Page 1 of 2.

Instructions

1) Read and sign the consent form. Please print your full name at the space provided at the top of the form. This is how we know who to credit COMP3020 marks.

2) Login to the computer if it is not already logged on.
Username: e2450
Password: e2450

3) Run: Start → My Computer

4) Navigate to F:\experiment

5) Run the application: ModelApp

6) Enter participant number into the experiment application. Your number is found on top of your consent form.

7) **Wait until after the administrator has completed the demonstration.** Then click "Start Experiment".

8) Perform the experiment by following the prompts on your screen. There is a progress indicator in the right side of the screen that shows how many trials you have completed so far.

9) The program will inform you when you have completed. That's it, you can leave.

Thank you very much for participating.

Revision Date: Oct 9, 2008

Consent form from the Wedge modeling study. Page 2 of 2.

BIBLIOGRAPHY

[1] Roland Arsenault, Colin Ware, Matthew Plumlee, Stephen Martin, Louis L. Whitcomb, David Wiley, Tom Gross, and Ata Bilgili. A system for visualizing time varying oceanographic 3D data. In *Proceedings of MTS/IEEE Conference on Oceans (OCEANS '04. MTS/IEEE TECHNO-OCEAN '04)*, volume 2, pages 743–747, 2004. (Cited on page 6.)

[2] Patrick Baudisch and Ruth Rosenholtz. Halo: A technique for visualizing off-screen objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*, pages 481–488, 2003. (Cited on pages 10, 14, 15, 22, 25, 26, 38, 42, and 67.)

[3] Benjamin B. Bederson and James D. Hollan. Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '94)*, pages 17–26, 1994. (Cited on page 5.)

[4] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535–546, 2004. (Cited on page 5.)

[5] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, 2002. (Cited on page 100.)

[6] Stefano Burigat and Luca Chittaro. Navigation in 3D virtual environments: Effects of user experience and location-pointing navigation aids. *International Journal of Human-Computer Studies*, 65(11):945–958, 2007. (Cited on page 11.)

[7] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. Visualizing locations of off-screen objects on mobile devices: A comparative evaluation of three approaches. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '06)*, pages 239–246, 2006. (Cited on pages iii, 9, and 10.)

[8] Stefano Burigat, Luca Chittaro, and Edoardo Parlato. Map, diagram, and web page navigation on mobile devices: The effectiveness of zoomable user interfaces with overviews. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '08)*, pages 147–156, 2008. (Cited on page 6.)

[9] Thorsten Büring, Jens Gerken, and Harald Reiterer. Usability of overview-supported zooming on small screens with regard to individual differences in spatial ability. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '06)*, pages 233–240, 2006. (Cited on pages 6 and 19.)

[10] Xiang Cao, Jacky Jie Li, and Ravin Balakrishnan. Peephole pointing: Modeling acquisition of dynamically revealed targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 1699–1708, 2008. (Cited on page 68.)

[11] Luca Chittaro. Visualizing information on mobile devices. *IEEE Computer*, 39(3):34–39, 2006. (Cited on page 6.)

[12] Luca Chittaro and Stefano Burigat. 3D location-pointing as a navigational aid in virtual environments. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '04)*, pages 267–274, 2004. (Cited on pages iii and 11.)

[13] Andy Cockburn and Carl Gutwin. A predictive model of human performance with scrolling and hierarchical lists. *Human-Computer Interaction*, 23(4), 2008. (Cited on pages 52 and 68.)

[14] Andy Cockburn and Joshua Savage. Comparing speed-dependent automatic zooming with traditional scroll, pan and zoom methods. In *Proceedings of the British Computer Society Conference on Human Computer Interaction (HCI '03)*, pages 87–102, 2003. (Cited on page 5.)

[15] Andy Cockburn, Carl Gutwin, and Saul Greenberg. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pages 627–636, 2007. (Cited on page 68.)

[16] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 2009. Accepted, to appear March 2009. Preprint available at `http://www.cosc.canterbury.ac.nz/andrew.cockburn/papers/fc.pdf`. (Cited on pages 15, 16, and 23.)

[17] James Elder and Steven Zucker. The effect of contour closure on the rapid discrimination of two-dimensional shapes. *Vision Research*, 33(7):981–991, 1993. (Cited on page 13.)

[18] Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954. (Cited on page 68.)

[19] George W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*, pages 16–23, 1986. (Cited on pages 15 and 17.)

[20] Brian Gaines. Modeling and forecasting the information sciences. *Information Sciences*, 3(22):3–22, 1991. (Cited on page 52.)

[21] Google. Google Maps. `http://maps.google.com/`. (Cited on pages 5 and 6.)

[22] Tovi Grossman and Ravin Balakrishnan. Pointing at trivariate targets in 3D environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*, pages 447–454, 2004. (Cited on page 68.)

[23] Tovi Grossman, Nicholas Kong, and Ravin Balakrishnan. Modeling pointing at targets of arbitrary shapes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pages 463–472, 2007. (Cited on page 68.)

[24] Yves Guiard and Michel Beaudouin-Lafon. Target acquisition in multiscale electronic worlds. *International Journal of Human-Computer Studies*, 61(6):875–905, 2004. (Cited on page 68.)

[25] Sean Gustafson and Pourang Irani. Comparing visualizations for tracking off-screen moving targets. In *Extended Abstracts on Human Factors in Computing Systems (CHI '07)*, pages 2399–2404, 2007. (Cited on pages 9 and 38.)

[26] Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. Wedge: Clutter-free visualization of off-screen locations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 787–796, 2008. (Cited on page 41.)

[27] Carl Gutwin. Improving focus targeting in interactive fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*, pages 267–274, 2002. (Cited on pages 18 and 21.)

[28] Carl Gutwin and Chris Fedak. Interacting with big interfaces on small screens: A comparison of fisheye, zoom, and panning techniques. In *Proceedings of the 2004 Conference on Graphics Interface (GI '04)*, pages 145–152, 2004. (Cited on page 18.)

[29] Catherine Q. Howe and Dale Purves. The Poggendorff illusion. In *Perceiving Geometry: Geometrical Illusions Explained by Natural Scene Statistics*, pages 85–96. Springer, 2005. (Cited on pages 14 and 15.)

[30] Takeo Igarashi and Ken Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '00)*, pages 139–148, 2000. (Cited on page 5.)

[31] Pourang Irani, Carl Gutwin, and Xing Dong Yang. Improving selection of off-screen targets with hopping. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, pages 299–308, 2006. (Cited on pages iii, 12, 32, and 69.)

[32] Pourang Irani, Carl Gutwin, Grant Partridge, and Mahtab Nezhadasl. Techniques for interacting with off-screen targets. In *Proceedings of the IFIP International Conference on Human-Computer Interaction (INTERACT '07)*, pages 234–249, 2007. (Cited on page 12.)

[33] Mikkel R. Jakobsen and Kasper Hornbæk. Evaluating a fisheye view of source code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, pages 377–386, 2006. (Cited on pages iii, 17, and 18.)

[34] Paul Janecek and Pearl Pu. A framework for designing fisheye views to support multiple semantic contexts. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI '02)*, pages 51–58, 2002. (Cited on page 17.)

[35] Susanne Jul and George W. Furnas. Critical zones in desert fog: Aids to multiscale navigation. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '98)*, pages 97–106, 1998. (Cited on page 7.)

[36] Microsoft Live Labs. Seadragon. `http://livelabs.com/seadragon/`. (Cited on page 5.)

[37] Heidi Lam and Patrick Baudisch. Summary Thumbnails: Readable overviews for small screen web browsers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*, pages 681–690, 2005. (Cited on page 5.)

[38] Eric Lee and James McGregor. Minimizing user search time in menu retrieval systems. *Human Factors*, 27(2):157–162, 1985. (Cited on page 68.)

[39] Ying K. Leung and Mark D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994. (Cited on pages iii and 16.)

[40] Tim Marsh and Peter Wright. Using cinematography conventions to inform guidelines for the design and evaluation of virtual off-screen space. In *Proceedings of the AAAI Spring Symposium Series on Smart Graphics*, pages 123–127, 2000. (Cited on page 14.)

[41] L. I. Medvedev and I. I. Shoshina. Quantitative assessment of the effects of hemispheric asymmetry on the distortion of visual perception of the Jastrow version of the Poggendorff figure. *Human Physiology*, 30(5):505–510, 2004. (Cited on page 15.)

[42] Miguel A. Nacenta, Samer Sallam, Bernard Champoux, Sriram Subramanian, and Carl Gutwin. Perspective Cursor: Perspective-based interaction for multi-display environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, pages 289–298, 2006. (Cited on page 10.)

[43] Miguel A. Nacenta, Regan L. Mandryk, and Carl Gutwin. Targeting across displayless space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 777–786, 2008. (Cited on page 10.)

[44] Dmitry Nekrasovski, Adam Bodnar, Joanna McGrenere, François Guimbretière, and Tamara Munzner. An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, pages 11–20, 2006. (Cited on pages 6 and 10.)

[45] Emmanuel Pietriga and Caroline Appert. Sigma Lenses: Focus-context transitions combining space, time and translucence. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 1343–1352, 2008. (Cited on page 6.)

[46] Catherine Plaisant, David Carr, and Ben Shneiderman. Image-browser taxonomy and guidelines for designers. *IEEE Software*, 12(2):21–32, 1995. (Cited on page 6.)

[47] John Predebon. Decrement of the Müller-Lyer and Poggendorff illusions: The effects of inspection and practice. *Psychological Research*, 70(5):384–394, 2006. (Cited on page 15.)

[48] Daniel C. Robbins, Edward Cutrell, Raman Sarin, and Eric Horvitz. ZoneZoom: Map navigation for smartphones with recursive view segmentation. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '04)*, pages 231–234, 2004. (Cited on page 4.)

[49] George Robertson, Jock D. Mackinlay, and Stuart Card. The Perspective Wall: Detail and context smoothly integrated. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '91)*, pages 173–179, 1991. (Cited on page 16.)

[50] Allison Sekuler and Richard Murray. Amodal completion: A case study in grouping. In Thomas Shipley and Philip Kellman, editors, *Fragments to Objects – Segmentation and Grouping in Vision*, pages 265–267. Elsevier, 2001. (Cited on pages 13 and 14.)

[51] Allison Sekuler and Stephen Palmer. Perception of partly occluded objects: A microgenetic analysis. *Journal of Experimental Psychology: General*, 121(1):95–111, 1992. (Cited on page 13.)

[52] Allison Sekuler, Stephen Palmer, and Carol Flynn. Local and global processes in visual completion. *Psychological Science*, 5(5): 260–267, 1994. (Cited on page 13.)

[53] Robert Spence and Mark Apperley. Data base navigation: An office environment for the professional. *Behavior and Information Technology*, 1(1):43–54, 1982. (Cited on pages 16 and 24.)

[54] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufman, second edition, 2004. (Cited on pages 13 and 16.)

[55] Colin Ware and Marlon Lewis. The DragMag image magnifier. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*, pages 407–408, 1995. (Cited on page 6.)

[56] Edward O. Wilson. *Consilience: The Unity of Knowledge*. Knopf, 1998. (Cited on page iv.)

[57] Wesley Woodson and Donald Conover. *Human Engineering Guide for Equipment Designers*. University of California Press, 1964. (Cited on page 16.)

[58] Polle T. Zellweger, Jock D. Mackinlay, Lance Good, Mark Stefik, and Patrick Baudisch. City Lights: Contextual views in minimal space. In *Extended Abstracts on Human Factors in Computing Systems (CHI '03)*, pages 838–839, 2003. (Cited on pages iii, 9, 10, and 38.)

COLOPHON

This thesis was typeset with the `pdflatex` LaTeX 2_ε interpreter using Hermann Zapf's *Palatino* type face for text and math and *Euler* for chapter numbers. The listings were set in *Bera Mono*.

The typographic style of the thesis was based on André Miede's wonderful `classicthesis` LaTeX style available from CTAN. My modifications were limited to those required to satisfy the constraints imposed by my university, mainly 12pt font on letter-size paper with extra leading. Miede's original style was inspired by Robert Bringhurst's classic *The Elements of Typographic Style* [5]. I hope my naïve, yet carefully considered changes are consistent with Miede's original intentions.

Final Version as of December 19, 2008 at 12:30.