Lodestones and Leylines: Designing Locomotion in a Personal File System

Susanne Jul
Electrical Engineering and Computer Science
University of Michigan
sjul@acm.org

ABSTRACT
We report on an exercise intended to help articulate constraints on locomotion and their implications for design. The assumption that locomotion is in service of navigation leads us to consider both cognitive and mechanical constraints. The exercise is to design support for accessing files in a hierarchical file system in the course of ordinary computer-based work. Characterizing the design situation in terms of four possible sources of locomotional constraints (the navigator, the task, the environment and the circumstances) leads to an abstract design of a dynamic locomotional mechanism that provides rapid accurate access to a small set of files. Applying this abstract design in Microsoft Windows™ and Jazz yields two quite different specific designs. The exercise results in a broad organization of the factors that constrain locomotion into three categories: Locomotional mechanism, Navigational resources and Navigational effect.

KEYWORDS
Locomotion, Navigation, Information Navigation, Hierarchical File System, File Access, Lodestone, Leyline, Multiscale, Jazz, Pad++, Windows, Zooming User Interface, Space-Scale Diagram, Design Framework.

INTRODUCTION
It is a fairly safe bet that you have some experience in using a computer to accomplish your daily tasks and that you interact regularly with a file system—probably hierarchical—to access your files. Stop for a moment and consider what kinds of navigational aids would be most useful to a user, such as yourself, to find and get to the files you need. A common response is, in some form or another, "a map," that is an overview of all the files in the system. We have come to believe that this is the wrong answer. This paper describes the reasoning behind this conclusion and offers the beginnings of a framework for identifying the salient navigational characteristics of a design situation and using them in generating useful designs.

By *navigation*, we mean the task of getting from one location to another. This encompasses the cognitive activity of deciding how to get there as well as the cognitive and mechanical activities involved in getting there. Navigation is fundamental to much human activity, including most human-computer interaction. However, it is generally incidental to such activity, that is, a means to an end rather than the end itself. As such, it takes extra time, consumes cognitive resources such as memory and attention, and disrupts the flow of the primary activity. Our goal is to develop designs that minimize navigational overhead.

The concept of *locomotion* or "getting there" is at the heart of navigational activity. The locomotional structure and mechanisms of a space (what locations are and how one moves between them) determine what navigation is necessary and possible. In the physical world, locomotion is constrained by the laws of physics and human anatomy. In an electronic world, all aspects of locomotion must be designed explicitly and may even be subject to dynamic modification.

Most prior work on navigation in electronic spaces has sought to apply navigational techniques that are successful in the physical world. Such approaches may preclude the development of novel navigational techniques only possible in electronic worlds, and encourage the transfer of techniques that reflect adaptations to special properties of the physical world. We rely heavily on research on navigation and spatial thinking done in the physical world, but seek to understand the inherent navigational possibilities and requirements of a design situation. Thus, we avoid the a priori assumptions implicit in any given navigational technique.

In this paper, we report on an exercise intended to help articulate the factors that constrain locomotion and explore their implications for design. We assume that locomotion is in service of navigation. The exercise is to design support for accessing files in a hierarchical file system in the course of ordinary computer-based work. This *file access task* is a commonly occurring task that we ourselves perform frequently and for which commercial applications provide numerous designs for comparison. The design is limited to support for locating and opening files (file system access). Support for creation and placement of files (file system modification) remains for an expanded exercise.

We first characterize the design situation in terms of four possible sources of locomotional constraints: the navigator,

the (superordinate) task, the environment and the circumstances under which the task is performed. We then draw a set of locomotional implications from this characterization and use them to develop an *abstract design* (a description of the essential characteristics of a design). This abstract design is applied in two different user interface environments, Microsoft Windows™ and Jazz (the Java-based successor to Pad++, a multiscale zooming environment). The resulting Windows design reproduces existing features, but uses them slightly differently than does the existing interface. The Jazz design introduces novel features. The Jazz design has been implemented and preliminary user tests have been highly encouraging.

As a result of this exercise, we have identified three categories of locomotional properties that may be constrained by navigational needs: Locomotional mechanisms, Navigational resources (both cognitive and mechanical), and Navigational effects (e.g., learning about the space). The categorization, along with the four sources of constraints, provides the beginnings of a framework that describes both what questions designers need to ask during the design process and what they should do with the answers. The similarity of the Windows design to existing designs that have evolved over years of use and the apparent utility of the Jazz design lead us to believe that this is a promising approach to helping designers produce effective designs rapidly.

RELATED WORK

Prior work on locomotional design typically regards locomotion as being in service of either a specific interaction need, such as pointing and selecting, or a particular type of environment. In contrast, we view locomotion as being in service of navigation, which, in turn is in service of some superordinate task, and which takes place in some particular environment.

We divide navigation into three sub-tasks [15]. *Locomotion* is the task of moving from one location to another. *Wayfinding* is the task of determining how to get from one particular location to another. *Spatial knowledge acquisition* is the task of learning about spatial relationships between locations. Locomotion has both cognitive and mechanical components, while wayfinding and spatial knowledge acquisition are cognitive tasks. We further divide locomotion into *steering*, controlling movement, and *traversal*, actually moving.

Task-Oriented Locomotion

Work on input devices—steering controls in navigational terms—typically focuses on the mechanical problems of pointing, selecting and free-hand drawing [1]. This work rarely, if ever, considers the cognitive affects of devices.

Steering in 3D has attracted considerable attention. Two specific techniques reduce cognitive load by simplifying steering. In Point of Interest navigation [16], the user indicates a point on an object on which they would like the view focused. The system then computes and moves the viewpoint along a "nice" path to that view. In Path Drawing [11], the user draws a path on the 2D display device. The system then computes a corresponding path in the 3D world and moves an avatar along that path.

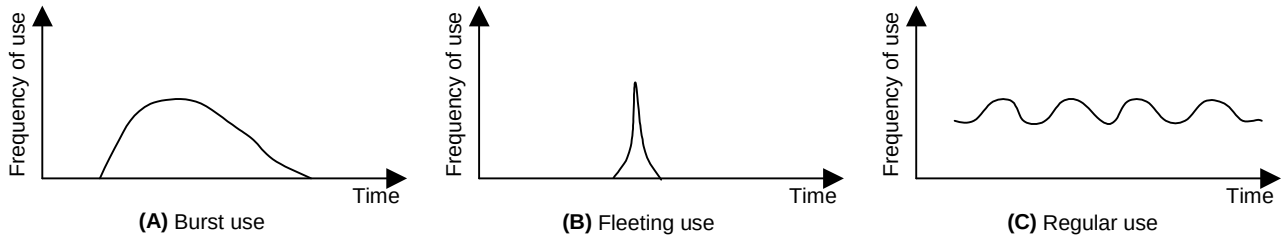
Both these techniques rely on implicit traversal constraints: Point of Interest navigation by pre-defined notions of "nice" paths, Path Drawing by pre-defined laws of physics (gravity, impenetrable surfaces) combined with object placement in the space. Both reduce the cognitive demands of steering, but neither contributes to reducing the overhead of wayfinding or spatial knowledge acquisition. Both techniques are limited to locomotion within a single view.

Another approach to steering in 3D bases locomotion on tools for wayfinding and spatial knowledge acquisition. Overview maps, such as that provided by a map window [3], provide an overview of the contents of the world. The user controls movement in a separate detail view by selecting or manipulating a rectangle corresponding to the detail view on the overview map. In a related fashion, Worlds-in-Miniature [20] allow users to hold a miniature view of a 3D virtual world and, fantasy-like, "step into" the World-in-Miniature, reentering the virtual world at a different location. These techniques simplify the mechanics of locomotion, but pre-suppose the need for wayfinding and spatial knowledge acquisition tools. Both techniques rely on the destination location being visible in an overview.

Environment-Oriented Locomotion

Interface metaphors generally embody a traversal mechanism. For example, most WIMP interfaces (those based on windows, icons, menus and pointers) use one-way visually-discontinuous hyperlinks that jump from icons to windows. (For our purposes, we define a hyperlink as a traversal mechanism that, once initiated, moves from a pre-defined source location to a pre-defined destination location with no further user intervention.) Interfaces with concepts of continuous space commonly employ flythrough or walkthrough mechanisms that emulate locomotion in the physical world. Such mechanisms inherit the cognitive support for navigation (or lack thereof) of the underlying metaphor.

In a different approach, the Pad environment [21] was developed from the principles of spatial cognition. The interface metaphor is of a conceptually infinite two-dimensional surface. The surface can be viewed at an infinite range of magnifications. Objects have position and extent on the surface, and can appear differently depending on the magnification (scale) of the view. Locomotion is by panning (moving across the surface) and zooming (changing the scale of the view). Both space and scale dimensions are conceptually continuous and infinite. This metaphor supports standard cognitive mechanisms of spatial knowledge (both acquisition and application) [13, 25], but wayfinding and steering have proven seriously difficult [14].



(A) Burst use **(B)** Fleeting use **(C)** Regular use

Figure 1 File access patterns.

CHARACTERIZATION OF THE DESIGN SITUATION

The present exercise is not concerned with how to gather information about a particular design situation or whether the information used is generally accurate. The focus is on identifying factors that constrain locomotion and the process of using knowledge of these factors in generating designs. Consequently, the characterization of the hypothetical design situation is derived from our personal experience. Had this been a real design, a significant part of the designer's task would have been to collect and validate the pertinent information about their design situation.

We presuppose that navigation is a context-dependent task [17] in which a particular user is trying to accomplish a particular task in a particular environment under particular circumstances. This assumption yields four possible sources of constraints on navigation and, consequently, on locomotion: the navigator, the superordinate task, the environment in which the navigation takes place and the circumstances of the navigational activity. For instance, cats choose quite different routes than humans, a mail carrier might need quite different transportation when working and when on vacation, ways across cities are quite different from ways through mountains, and some people always walk on the sunny side of the street. So we characterize our design situation in terms of these four elements.

Design Situation

Our basic assumption is that the user is a "normal" user using a "normal" desktop personal computer system—a la late twentieth century. The system uses a hierarchical file storage system. It is a single user system and we are designing support for the regular user. Our immediate goal is to support inter-file navigation with no concern for intra-file navigation.

The User

The user has "normal" physiology (vision, eye-hand coordination, etc.) as well as "normal" cognitive skills and resources (memory, attention, reasoning, etc.). They are at least somewhat familiar with the interface provided and may be expert users. As this is their own system, they have organized at least part of the file system themselves. Thus, they have some understanding of the way it is organized and some memory of where things are. This knowledge may be incomplete and it may be inaccurate, in particular, recall of exact names and locations of files may be faulty [23].

The Task

The file access task is undertaken for multiple purposes. We do not speculate on these purposes, but assume that they give rise to the following tasks: (1) finding a specific file in order to edit, display or copy its contents, (2) starting an application to create content, or (3) modifying the file system by deleting unnecessary files, reorganizing the structure or saving content to a new location. For the present exercise, we concentrate on the first two types of tasks.

Our experience suggests three patterns of access to a particular file: *burst use*, *fleeting use* and *regular use*. In burst use (Figure 1A), the user uses a file intensively for a period of time and only rarely after that. This pattern is typical of document preparation. In fleeting use (Figure 1B), a file is accessed a few times in a short period of time, then not accessed for a long while. This typically represents information "look-up" in the superordinate task. In regular use (Figure 1C), a file is accessed regularly on a continuing basis. This is typical of files that represent regularly used applications. These patterns may be apparent within single as well as across multiple sessions; e.g., a file may be used once an hour for one day, while another is used once a month every month. Use of one file frequently overlaps use of another. Our experience suggests that the majority of file accesses are part of a burst or regular use pattern.

In our experience, the user only accesses a small number of the files in the environment manually. The remaining files are accessed, if at all, by software and not by the user directly. Those files that are accessed manually tend to cluster in groups or subhierarchies. These subhierarchies contain related information and their hierarchical structures are significant to the user.

Note that, while these observations are based on introspection, they are borne out by studies conducted by Barreau and Nardi [2]. They studied use of information rather than files, and report categories of working, ephemeral and archival information. The first two are analogous to our categories of burst/regular and fleeting use, respectively. Nardi et al. [18] confirm our assumption that users typically only access a small number of the overall files. The behavior patterns described match those evinced by "knowledge work" as described by Williamson [27]. (This is in contrast to "procedural work," e. g., data entry, where few files are used repeatedly.)

The Environment

We assume that the environment constitutes a personal information space and has a relatively stable structure. This structure is, conceptually, a collection of hierarchical tree structures although a common root node may be provided that converts the forest into a single tree. Changes to the structures and their contents are mostly incremental. There is a single, medium-sized (10"-20"), pixellated, 2D display device. Input is based on use of a mouse and the interaction metaphor favors direct manipulation. We do not assume any particular interaction metaphor at this point.

The Circumstances

We assume that the user carries out their superordinate task under comfortable conditions. That is, while they might like to complete the task rapidly, there is no critical urgency to complete it or any aspect of it. The user is able to devote most, if not all, their cognitive and physical resources to the task. If any external aids (documents, other people, etc.) are present, these pertain to the superordinate rather than the file access task.

Locomotional Implications

The locomotional structure offered by the environment is that of the file hierarchy: Locations are the nodes of the trees and routes follow the tree structure. While some of these locations and routes may be more important than others in terms of the semantic interpretation of the tree, none are distinguished in terms of locomotion.

The significant objects for the task, in contrast, are files, that is, leaf nodes in the tree structures. Internal nodes are not significant to the task. The patterns of use of files encourages us to distinguish between files that are part of an active or incipient burst or regular use pattern and those that have not been accessed manually for some period of time. This indicates two variants of the file access task: *repeated file access*, in which the target file is already part of a pattern, and *initial file access*, in which it is not. In the initial access task, all leaf nodes are potential target locations. However, in the equally important and more frequent (in our design situation) repeated file access task, target locations represent a small subset—identifiable by patterns of use—of all leaf nodes.

The routes suggested by the task also differ from those offered by the environment. For the initial access task, the problem is selecting among a large set of possible destinations—a wayfinding problem. For the repeated access task, the primary problem is not selecting the destination, but getting there—a locomotional problem. Both are facilitated by short routes, ideally single steps, but longer routes interfere directly with the repeated access task. Traversing the tree structure, as required by the environment, is not beneficial in the repeated access task, while it may facilitate wayfinding decisions in the initial access task [7, 8].

The environment requires knowledge of the organization of the tree structures for efficient navigation [7, 8]. This knowledge can either be precise memory of locations and routes, or it can be knowledge of the semantics of the structure. The navigator has this knowledge, but it may be partial or inaccurate. The circumstances are such that the navigator is able to direct their attention to applying this knowledge, however, at the expense of the superordinate task. This knowledge is not inherently needed for the task, so need only be acquired if necessary for future navigation.

We conclude that the two variants of the file access task differ navigationally. An informed design for the initial access task requires further examination from a wayfinding perspective. The repeated access task, however, is amenable to a locomotional design approach. The characterization indicates that the most useful navigational aid for repeated file access is a locomotional mechanism that provides rapid accurate access to small set of leaf nodes. The set of leaf nodes is determined dynamically based on access history, with files that have been used recently and extensively (in terms of frequency and duration) being candidates. The characterization indicates that map-like navigational aids, which treat all nodes in the file system trees equally, are not appropriate for the repeated access task.

DESIGNS: LODESTONES AND LEYLINES

The following designs develop locomotional support for the repeated access task. This is not to belittle the initial access task, but acknowledges that, in our design situation, the repeated access task is the more common variant of the file access task. Also, design for the initial access task requires additional analysis from a wayfinding perspective.

We first develop an abstract design that describes the essential characteristics of a final design. We then apply this to two different interface environments that each adds different locomotional constraints.

Abstract Design

The shape of the abstract design is straightforward at this point. The design provides a mechanism that monitors the time, frequency and duration of use of each file that is accessed manually. A formula of the form

$$\frac{\text{duration} * \text{frequency}}{c * \text{time since last use}}$$

where c is some appropriate constant, is used to compute the likelihood of a file being accessed again. The locomotional structure of the space is augmented so that there is a central location, which can be reached in one step from any location. Further augmentation provides short paths from this central location to those files that have a high likelihood of being accessed. These files are called *lodestones* (since they "attract" navigational attention). We define short paths as being of length one for regular and burst use files and two for fleeting use files. This new locomotional structure is updated as files are used.

Thresholds for including and categorizing lodestones would need to be determined empirically and, probably, be under user control to some extent. Similarly, the user should be able to override the dynamic classification, for instance, making a file a permanent lodestone so that it serves a reminding or nagging function.

In the applications of the abstract design, our interest is in the design of the dynamic locomotional structure. We recognize that determining when a file is actually in use and classifying usage patterns of a file are non-trivial problems. Our specific designs merely suggest naïve solutions.

Windows Design

We presume that the reader is familiar with the Microsoft Windows™ interface and the desktop metaphor upon which it is based, so do not explain the metaphor here.

Windows Environment Characteristics

Windows maps the tree structure of the file system onto a system of icons and windows. Open windows represent active files (leaf nodes representing documents or applications) as well as internal nodes. Icons represent both leaf and internal nodes. Clicking on an icon opens the corresponding window. There is a single special desktop "window" that is always available that provides a workspace in which to manipulate windows. Conceptually, this desktop is not part of the file system, but usually contains icons that represent the roots of the file system trees.

There are three ways of accessing a particular file. First, one can follow the file system tree structure by starting at the top and clicking through icons and windows until the desired file is reached. Second, one can use the "Explorer" interface, a map-like mechanism that displays a tree diagram in one part of a window and the contents of a single (selected) internal node in another. The diagram is manipulated until the icon representing the desired file is in view. The tree structure, in this case, is traversed mentally and not always mechanically, offering a sense of locomotion across the structure. Third, shortcuts can be created that provide direct access to a particular location in the tree. Shortcuts are represented as icons and can be stored within a tree structure or on the desktop.

Locomotional Implications

The Windows environment defines three types of locations: icons, windows, and the special directly-accessible desktop. Traversal is via hyperlinks from icons to windows. Routes either follow the file system structure or are one-step shortcuts that cut across it. Like the basic environment, efficient navigation in a Windows environment requires knowledge of the organization of the tree structures. However, this knowledge can be less accurate due to a reliance on recognition rather than recall for wayfinding. If the appropriate shortcuts are available, no such knowledge is needed.

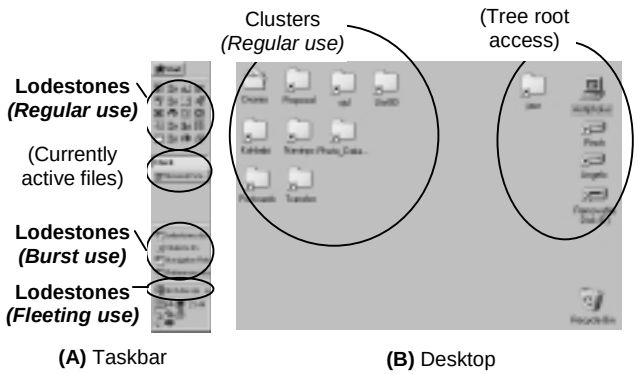


Figure 2 Windows design. **(A)** Dynamic locomotion structure for repeated file access task. **(B)** Conjectured locomotional support for initial access task.

This suggests that the target locations for the file access task in a Windows environment are windows representing open files. Icons serve as locomotional intermediaries, but are not themselves target locations. Shortcuts match the kinds of routes suggested by the task.

Specific design

We base our specific design on use of the desktop and shortcuts. Specifically, we use the taskbar, an iconic menu on the desktop, as our central location. We use a file being open in a running application and the corresponding window being visible on the screen as our metric for file usage. The design for the locomotional structure is illustrated in Figure 2A. There are three separate areas for the three types of lodestones. Shortcuts to regular and burst use lodestones are all shown directly. Only the most likely of the fleeting use lodestones is shown, along with an arrow indicator for a flyout menu displaying the rest. We elect to have regular use lodestones appear as icons only, whereas burst and fleeting use lodestones have text labels alongside their icons. The exact configuration of inclusion and placement of text labels should be under user control.

For completeness, we present a conjectured locomotional design for the initial access task. This design is based on intuition and the recognition in our characterization of the file access task that files accessed manually tend be clustered together in the file system. In this design (Figure 2B), we place shortcuts directly on the desktop, allowing them to be reached in one step from most "File Save As" dialogs. On the right, there is permanent access to the roots of the trees in the file system. On the left, there are shortcuts to regularly used clusters of files. We suspect, based on our own experience, that they correspond to the lowest common ancestors in the file tree of burst and fleeting use lodestones and can be identified using a relatively simple algorithm. (Note that this algorithm does not consider the route used to access a file.) Accordingly, they are managed dynamically like lodestones.

Implementation and Testing

The Windows design has not been implemented, but it is possible to approximate the suggested design using existing

features. Maintaining the locomotional structure to reflect current file use requires constant effort and attention. Despite this effort, the author finds this approach quite productive and rarely uses the standard navigational tools outside of localized clusters.

Jazz Design

Jazz [5, 12] is the Java-based successor to Pad++ [4], which was based on Pad [21]. The interface metaphor is the same as that of Pad, described earlier. Note that the pan and zoom model of locomotion moves the viewpoint relative to the surface and, thus, to objects on the surface, whereas the Windows model has a fixed viewpoint and moves objects relative to the desktop.

In the following discussion, we rely on the space-scale notation developed by Furnas and Bederson [9] for description of interactions. This notation is summarized in the box "Understanding Space-Scale Diagrams."

Jazz Environment Characteristics

Jazz is an application framework designed to support the development of multiscale applications using zooming user interfaces. It does not provide a metaphor for or means of interacting with files or the underlying file system, so our design for its use as a desktop is both speculative and suggestive. We presume that the user would organize files on the surface, much as they would on a physical desktop, using spatial proximity to indicate semantic relationships and scale to fit the desired objects into the (conceptually) allocated space. This implies that the layout of objects on the surface is entirely under user control. Navigational aids are not at liberty to alter the layout.

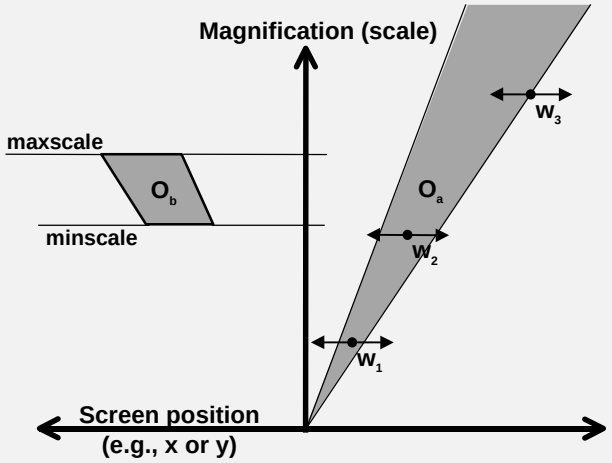
Bederson and Hollan [4] suggest two techniques for interacting with files. First, they suggest a zooming directory browser. This is analogous to the icon-window means of traversing the file system in Windows. Rather than clicking on an icon to open a window, in the directory browser, the user zooms in on the object representing an internal node. As this object is magnified, the details of its contents become visible and zoom-in can continue on the next level until the desired file is reached. Second, they suggest that zooming in on a particular file automatically starts the appropriate application.

The theoretical space-scale origin (x = 0, y = 0, magnification = 0) is a special location in Jazz. At this point, objects are infinitely small and are all rendered (theoretically) at the same screen location. As the view zooms out (magnification decreases), all points on the surface converge visually. This convergent property is evident in the classic V shapes of space-scale diagrams.

Locomotional Implications

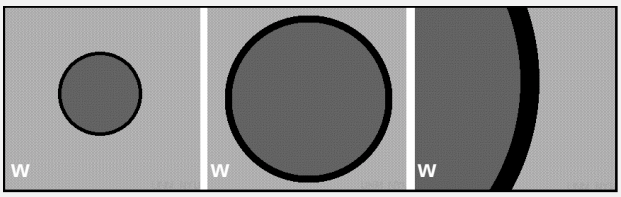
In Jazz, locations are views in space-scale, i. e., a point on the surface and a magnification at which it is being viewed. By convention, being at a view means that the viewpoint is centered on that point in space and the scale of the view is set to that magnification in scale. Routes are trajectories

UNDERSTANDING SPACE-SCALE DIAGRAMS



Space-scale diagrams were developed as a tool for understanding multiscale spaces [9]. They show the apparent change in size and position of an object relative to the magnification of the view. In the sample diagram above, the horizontal axis indicates location in screen-space (e.g., x-coordinate) and the vertical axis indicates degree of magnification (the scale-coordinate). Note that zooming "in" and "out" correspond to moving "up" and "down," respectively, in the diagram.

In the simple case, an object only grows in size as it is magnified. Such geometrically-scaling objects, like $\mathbf{O}_a$ in the sample diagram, have a V shape in a space-scale diagram, indicating that the object appears to be infinitely small at infinitely small scales, and grows larger as the view is magnified. In practice, an object typically has a minimum magnification at which it is rendered, its *minscale*, or automatically disappears when it is smaller than one pixel. Objects also have a maximum effective magnification, the *maxscale;* e.g., when they fill the view uniformly they are often culled by the rendering system. These limits are shown schematically for object $\mathbf{O}_b$ in the sample diagram.



A particular view of the world is defined by the position in space and scale of a window with a given width. This is represented in a space-scale diagram by a horizontal line whose midpoint represents of the center of the window. (Note that we assume uniform magnification across any particular view.) Since the width of the window is unaffected by the magnification of the view, a line representing a particular window will have the same width throughout the diagram. In the sample diagram, $\mathbf{w}_1$ is a view in which $\mathbf{O}_a$ fills the middle third of the window, as shown in the first of the screen-shots above. $\mathbf{w}_2$ has zoomed in on (the now magnified) $\mathbf{O}_a$, as shown in the second screen-shot. $\mathbf{w}_3$ has zoomed in further and panned right almost half a window width, as shown in the third screen-shot.

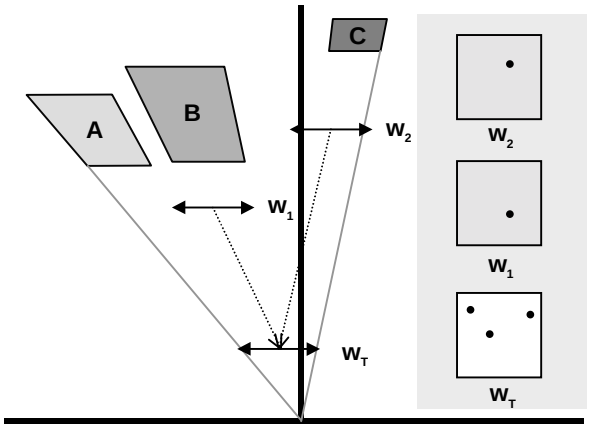


Figure 3 Zoom-out is constrained to move toward, but not past, the Top of the [Lodestone] World (w_T). In the space-scale diagram, this is shown as dotted arrows. In the schematized views on the right, lodestone locations are indicated by dots. Zoom-out is possible from w_1 and w_2, but not w_T. Gray lines in the space-scale diagram show the defining boundaries of the Top of the [Lodestone] World.

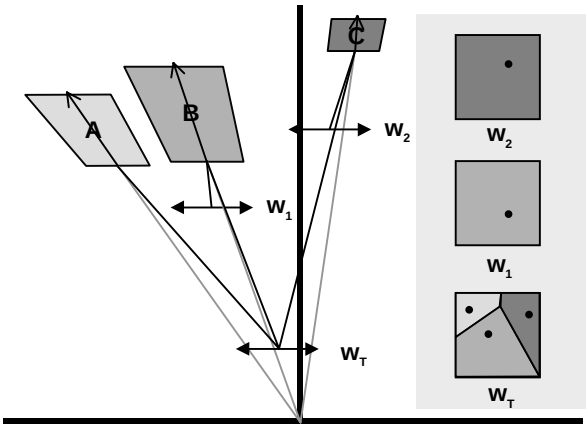


Figure 4 Zoom-in is constrained to follow a leyline (black arrows) to a lodestone contained in the current view. If there are multiple lodestones in the view, the leyline that leads to the lodestone whose center (gray lines) is closest to the mouse location is followed. In w_T, clicking anywhere in the light gray area leads to **A**, medium gray to **B** and dark gray to **C**. Clicking anywhere in w_2 leads to **B**, anywhere in w_1 to **C**.

through space-scale, i. e., paths from view to view. (It is possible to program movement that jumps between views with no visual continuity, but this is contradictory to the basic philosophy of the environment.)

If there is a view in which all objects on the surface are visible at sufficient detail to be recognized and that view can be found from anywhere in the space, navigation in Jazz requires no prior knowledge [14]. In our experience, this only occurs in small specially-designed worlds. If no such view exists, then the navigator must know the layout of objects on the surface and in scale. This layout may or may not correspond to the file system structure.

The mismatch between the concepts of location in the file access task in Jazz and the Jazz environment itself is immediately apparent. Locations in the task are views of objects, while locations in the environment are points in space-scale. This suggests that locomotion should be relative to objects on the surface rather than to the surface itself.

Specific Design

We could employ a specific design in Jazz that is analogous to the Windows design by providing a menu of "shortcuts" that would change the current view to focus on the desired objects. However, we wish to respect the spatial nature of the environment and support locomotion that retains the sense of spatial movement.

File usage statistics can no longer be based on the opening and closing of windows. Instead, we adapt Bederson and Hollan's idea and consider a file as being in use when the object representing it is visible on the screen and larger than some minimum size. A daemon mechanism runs periodically to collect statistics. Another possibility would be to track the "opening" and "closing" of individual files by collecting information at the conclusion of each pan or zoom.

The environment offers only one special location, the space-scale origin, but this is not useful for practical purposes. However, the fact that the set of lodestones is finite suggests another: the view that is centered on the bounding box of all lodestones whose magnification is the largest at which that bounding box will fit within the window. This view is called the *Top of the* [Lodestone] *World* and serves as the special location called for by the abstract design. Since, by definition, all lodestones are within the window rectangle and can thus be reached by zooming in, zooming out beyond this view serves no purpose. So zoom-out is constrain in two ways (Figure 3): Zooming out always moves towards the Top of the World, and zooming is not permitted past this view. We call space-scale trajectories that constrain locomotion *leylines* (after the mythical lines in Celtic tradition that can be followed to nodes of power and, in some cases, yield access to the fairie realms).

Lodestones can be reached from the Top of the World by zooming in. To ensure direct routes to lodestones, zoom-in is constrained to follow the shortest path from the current view to the center of a lodestone (Figure 4). Such a zoom-in leyline ends at the lodestone's maxscale, if any, so zoom-in is only permitted if there is a lodestone in the view.

In the present implementation, the user presses a mouse button to zoom in. The system selects the lodestone whose center is closest to the mouse location and begins to follow the leyline toward that lodestone. Only lodestones that are contained in the current view are considered. If, during the zoom, the user moves the mouse closer to another lodestone, the system switches to the leyline leading to that

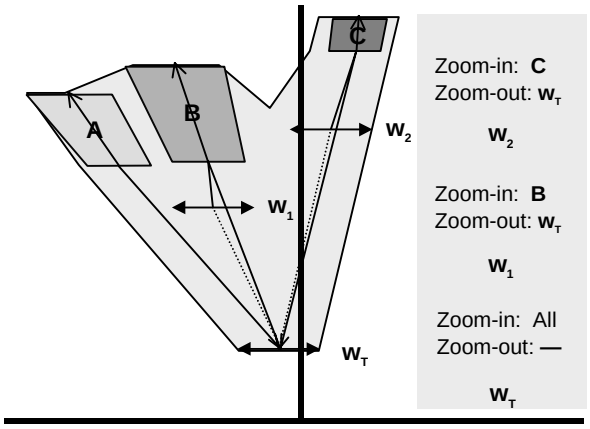


Figure 5 Locomotion constrained by lodestones and leylines. E.g., $\mathbf{w_T}$ offers 3 options, $\mathbf{w_2}$ and $\mathbf{w_1}$ two each. Consequently, interaction is restricted to the gray area.

lodestone. An alternate steering control treats leylines as hyperlinks and automatically follows a leyline to its destination once it has been selected.

Leylines represent an aberrant zoom behavior in Jazz. In spite of the use of zoom "in" and "out" to describe following leylines, leyline trajectories are generally combined zoom and pan movements. "In" and "out" merely denote the direction of the zoom component. This is evident in Figure 3 - Figure 5, where leylines do not intersect the space-scale origin. Zoom-in leylines are in two parts: the first is a pan-zoom that centers the view on the lodestone, with the lodestone nearly filling the view in at least one dimension, the second a pure zoom used once the lodestone is centered in the view. This two-part behavior is particularly evident in Figure 4 when following the leyline from view $\mathbf{w_1}$ to lodestone **B**.

Constraining locomotion to leylines reduces the number of locomotional options in each view greatly (Figure 5), and reduces the number of navigational decisions the user must make accordingly. It also simplifies steering significantly and, incidentally, limits movement to a finite volume of space-scale (Figure 5). These latter two properties serendipitously address two major problems with the default navigation mechanisms in both Pad++ and Jazz: the general difficulty of steering and the near certainty of getting lost in space-scale [14].

The Windows design segregated the three types of lodestones so that burst and regular use files were more easily accessible than all but one of the fleeting use files. In the Jazz design, lodestones are weighted according to their likelihood of access. When selecting among possible leylines, lodestones with a higher weight (heavily used burst files or regular files) have a stronger pull. Thus, the distance calculation of how "far" a lodestone is from the mouse location is weighted with the likelihood of that lodestone being the target location.

For completeness, we conjecture a design for the initial access task. As for the Windows design, these ideas are not

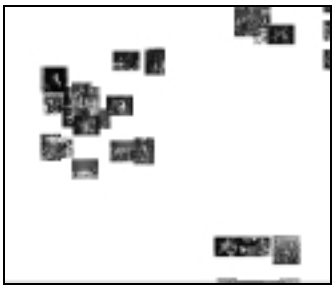


Figure 6 Example Jazz world used in user tests.

based on analysis, but rather on intuition and experience with the environment. In Jazz, the initial access task is a two-part problem: the file must first be located in the file system and then placed on the surface. For the first part, interaction with the underlying file system, we suggest a dialog window containing a secondary surface with a directory browser as designed by Bederson and Hollan [4]. Clusters act as the lodestones of this surface. They are identified as in the Windows design. While placing the new objects on the primary surface, spatial groups of existing lodestones (identified using a spatial clustering algorithm) act as lodestones. A keyboard modifier allows the user to move off leylines temporarily during this time. A similar "zooming dialog" interface is used for finding and launching applications, much in the manner intimated by Perlin and Meyer [22].

Implementation and Testing

The Jazz design is fully implemented. A small formative study comparing lodestone and leyline locomotion to the Pad++ model has been completed. Both models use separate mouse buttons for zoom-in and zoom-out. In the Pad++ model, the center of the zoom follows the mouse. Panning was permitted in both cases, using a keyboard-modified mouse action, but few subjects used this capability. All subjects ended by favoring lodestones and leylines, although some had slight difficulty in giving up the manual control afforded by the Pad++ model. All subjects spontaneously reported feeling less lost or confused and more confident about their actions when using lodestones and leylines. Two subjects attributed this, in part, to the ease of returning to the Top of the World. One subject had to be persuaded to return to the Pad++ model to navigate a second layout (Figure 6).

RESULTS OF THE EXERCISE

In addition to the two specific designs, the exercise yielded a broad organization of the factors that affect locomotion. These fall into three categories: Locomotional mechanism, Navigational resources and Navigational effect.

In the exercise, task and environment were the primary sources of constraints on locomotional mechanisms, determining locations, routes, and time spent on traversal. The Jazz environment raised issues surrounding the accuracy in following routes and reaching locations. In a different design situation, the circumstances might play a

more important role, e. g., if the machine were networked and file access dependent on network connectivity. A different user population might be confused by a dynamic locomotion structure and benefit more from a slow but fixed structure.

Task and environment were also the primary sources of constraints on navigational resources in the exercise. The environment dictated knowledge of the locomotional structure, whereas the task indicated that devoting cognitive resources to applying such knowledge would be counter-productive. Interestingly, knowledge of the navigator did not add constraints, but was needed to ensure that the precondition of the environmental constraint was satisfied. Other resources that might be constrained include the presence of navigational tools (substituting for actual knowledge), external assistance (e. g., from others), time allotted for task, etc.

The acquisition of spatial knowledge was the only constraint on navigational outcome in the exercise. The task did not inherently require spatial knowledge, but, if the task were to be repeated in the basic environment (as assumed), acquiring spatial knowledge during task performance would facilitate future interactions. Other navigational outcomes might include enjoyment, distance traversed, exposure to certain types of intermediate locations (e. g., sites of historical interest), etc. Many of these are dictated by individual preferences and longer-term goals.

FUTURE WORK

We would like to pursue three lines of research further. First, further evaluation of the proposed designs is needed. This includes implementing and testing the Windows design and a more formal study of the Jazz locomotional mechanisms. Also, the present implementation of the Jazz design uses fixed rates of locomotion. We would like to experiment with varying rates of motion based on the distance to be traversed and the number of locomotional options in view.

Second, we intend to develop a more formal framework of the factors that affect locomotion and their implications for design. This entails revisiting the psychological literature on navigation and spatial cognition, further design studies (for example, varying the task rather than the environment, as in the present exercise) to elucidate the framework itself and design studies to validate the framework.

Finally, we would like to investigate the computational implications of view-dependent interaction. For example, in the present exercise, thumbnail views of objects were used as wayfinding aids. This may require a different minscale setting for the object than that used in the primary view camera. However, the present Jazz implementation assumes a one-to-one association between object and minscale. Thumbnails imply a one-to-one association between an object-camera pair and the minscale value.

SUMMARY

We have described a design exercise aimed at exposing the factors that affect locomotion and explore their implications for design. Characterizing our design situation in terms of four possible sources of locomotional constraints (the navigator, the task, the environment and the circumstances) led us to conclude that appropriate locomotional support for the repeated file access task is a dynamic locomotion mechanism that gives rapid accurate access to a small set of files. Using this mechanism neither requires nor incurs overhead for spatial knowledge acquisition. From these conclusions, we developed an abstract design that describes the essential characteristics of a final design, which, applied to the Windows and Jazz environments, yielded different specific designs.

Although the characterization of the design situation was based on introspection rather than analysis, it is noteworthy that the resulting designs reflect known ideas as well as new ones. The design goal of maintaining a dynamic locomotion structure is consistent with the theory of cost-structuring of information spaces [24] and empirical evidence that people expend considerable effort to reduce wayfinding in the course of accomplishing their tasks in electronic worlds [26]. Basing lodestone designations on usage patterns is reminiscent of combining concepts of history mechanisms [6, 19] with the notion of read wear and edit wear [10]. Shortcuts, recognized as fundamental in our design, were a late addition to desktop interfaces. (Although the concept of aliases already existed in UNIX systems, it first appeared in the generally well-designed Macintosh interface in System 7, 1991, seven years after its initial release in 1984.) Leyline locomotion is a new concept in Jazz.

The categorization resulting from the exercise may be incomplete and much work remains in determining what factors constrain locomotion and how. Nonetheless, along with the four sources of constraints, it suggests a framework that describes both what questions designers need to ask during the design process and what they should do with the answers. The similarity of the Windows design to designs that have evolved over years of use and the apparent utility of the Jazz design lead us to believe that this is a promising approach to helping designers produce effective designs rapidly.

ACKNOWLEDGEMENTS

The author thanks George W. Furnas for unflagging intellectual support, and Rudy Darken, Barry Peterson and the MOVES department of the Naval Postgraduate School for intellectual support and shelter in the sun.

REFERENCES

1. Baber, C. (1997). *Beyond the Desktop: Designing and Using Interaction Devices.* San Diego, CA: Academic Press.

2. Barreau, D., Nardi, B. A. (1995). Finding and Reminding: File Organization from the Desktop. *SIGCHI Bulletin*, Vol.27, No.3 (July).

3. Beard, D. V., Walker, J. Q. II. (1990.) Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces. *Behaviour & Information Technology*, vol. 9, no. 6, 451-466.

4. Bederson, B. B., Hollan, J. D. (1994). Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of ACM UIST'94*, New York, NY: ACM Press, 17-26.

5. Bederson, B. B., McAlister, B. (1999). *Jazz: A Java Zooming Toolkit*, CS-TR-4015, UMIACS-TR-99-24, (May).

6. Foss, C. L. (1989). Tools for Reading and Browsing Hypertext. *Information Processing & Management*, 25 (4), 407-418.

7. Furnas, G. W. (1997). Effective View-Navigation. *Human Factors in Computing Systems CHI '97 Conference Proceedings*, New York, NY: ACM Press, 367-374.

8. Furnas, G. W. (1995). Effectively View-Navigable Structures. Paper presented at the 1995 Human Computer Interaction Consortium Workshop (HCIC95), Snow Mountain Ranch, Colorado, Feb. 17, 1995. Manuscript available at `http://http2.si.umich.edu/~furnas/POSTSCRIPTS/EVN<HCIC95.workshop.paper.ps`

9. Furnas, G. W., Bederson, B. B. (1995). Space-Scale Diagrams: Understanding Multiscale Interfaces. *Human Factors in Computing Systems CHI '95 Conference Proceedings*, vol. 1, New York, NY: ACM Press, 234-241.

10. Hill, W. C., Hollan, J. D., Wroblewski, D., McCandless, T. (1992). Edit Wear and Read Wear. *ACM Conference on Human Factors in Computing Systems - CHI '92*, p. 3 – 9. New York: ACM.

11. Igarashi, T., Kadobayashi, R., Mase, K., Tanaka, H. (1998). Path Drawing for 3D Walkthrough. *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology, UIST 98*, 173-174.

12. http://www.cs.umd.edu/hcil/jazz

13. Hirtle, S. C., Jonides, J. (1985). Evidence of Hierarchies in Cognitive Maps. *Memory & Cognition*, 13(3), 208-271.

14. Jul, S., Furnas, G. W. (1998). Critical Zones in Desert Fog: Aids to Multiscale Navigation. *ACM Symposium on User Interface Software and Technology, UIST 98*, 97-107.

15. Jul, S., Furnas, G. W. (1997). Navigation in Electronic Worlds. *SIGCHI Bulletin*, 29, 4 (Oct), 44-49.

16. Mackinlay, J. D., Card, S. K., Robertson, G. G. (1990). Rapid Controlled Movement Through a Virtual 3D Workspace. *SIGGRAPH '90 Conference Proceedings*, in *Computer Graphics* 24 (4, Aug.), 171-176.

17. Nardi, B. (1996). Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition. In Nardi, B. (Ed.) *Context and Consciousness*, 69-102.

18. Nardi, B. Anderson, K., Erickson, T. (1995). Filing and Finding Computer Files. *East-West International Conference on Human-Computer Interaction: Proceedings of the EWHCI'95*, 162-179. Intl. Centre for Scientific & Technical Information.

19. Parunak, H. V. D. (1989). Hypermedia Topologies and User Navigation. *Hypertext '89 Proceedings*, ACM, 43-50.

20. Pausch, R., Burnette, T., Brockway, D., Weiblen, M. E. (1995). Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures. *ACM SIGGRAPH `95 Conference Proceedings, Computer Graphics*, (July).

21. Perlin, K., Fox, D. (1993). An Alternative Approach to the Computer Interface. *Proceedings of the ACM SIGGRAPH 93 Conference*, New York, NY: ACM Press, 57-64.

22. Perlin, K., Meyer, J. (1999). Nested User Interface Components. *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, 11-18.

23. Pezdek, K., Evans, G. W. (1979). Visual and Verbal Memory for Objects and Their Spatial Locations. *Journal of Experimental Psychology: Human Learning and Memory*, 5(4), 360-373.

24. Russell, D. M., Stefik, M. J., Pirolli, P., Card, S. K. (1993). Cost Structure of Sensemaking. *Conference Proceedings on Human Factors in Computing Systems 1993*. New York, NY: ACM, 269-276.

25. Tolman, E. C. (1948). Cognitive Maps in Rats and Men. *Psychological Review*, 55, 189-208. Reprinted in Downs, R. M., Stea, D. (Eds.) *Image and Environment; Cognitive Mapping and Spatial Behavior*. (1973).

26. Watts, J. (1994). Navigation in the Computer Medium: A Cognitive Analysis. *Proceedings of the Human Factors and Ergonomics Society 1994*, Human Factors and Ergonomics Society, Inc., Santa Monica, CA, USA, 310-314.

27. Williamson, A. (1998). Moneypenny: Lessons from the Messy Desk. *Interacting with Computers*, v.9 n.3, 241-267.