Nonvisual Interaction Techniques at the Keyboard Surface

Rushil Khurana, Duncan McIsaac, Elliot Lockerman
Carnegie Mellon University,
Pittsburgh, PA, USA
rushil@cmu.edu

Jennifer Mankoff
University of Washington
Seattle, WA, USA
jmankoff@acm.org

ABSTRACT
Web user interfaces today leverage many common GUI design patterns, including navigation bars and menus (hierarchical structure), tabular content presentation, and scrolling. These visual-spatial cues enhance the interaction experience of sighted users. However, the linear nature of screen translation tools currently available to blind users make it difficult to understand or navigate these structures. We introduce Spatial Region Interaction Techniques (SPRITEs) for nonvisual access: a novel method for navigating two-dimensional structures using the keyboard surface. SPRITEs 1) preserve spatial layout, 2) enable bimanual interaction, and 3) improve the end user experience. We used a series of design probes to explore different methods for keyboard surface interaction. Our evaluation of SPRITEs shows that three times as many participants were able to complete spatial tasks with SPRITEs than with their preferred current technology.

Author Keywords
Accessibility; interaction techniques; keyboard

INTRODUCTION

Although assistive technology has done much to improve access to computers, the tools available to blind users are still lacking [2,20,27]. GUIs proactively layout and layer on-screen content, provide structure and present visual cues that enhance interaction. In contrast, screen readers (and braille displays) produce mostly linear, ephemeral streams of text. The two-dimensional visual structures that websites rely on to convey information are easily lost in translation, because presenting spatial and visual cues are challenging to represent in a linear format [*e.g.,* 3,20,22,23,27]. The inaccessibility of spatial layout information has prevented visually impaired individuals from equal access to information and services as a sighted individual [20].

In this paper, we argue that it is possible to enhance existing nonvisual technologies with *spatial input and interaction techniques* by leveraging a device that blind users use every day—the keyboard. The keyboard is not normally thought of as a pointer or mapped directly onto interactive elements on the screen. However, it could be used for spatial interaction. There are about 90 tangibly distinct keys on most keyboards, laid out (approximately) in a two-dimensional (2D) grid. In addition, unlike other tangible interfaces, a keyboard is essentially free (since every desktop and laptop already has one).

We call our approach *SPRITEs* (*Spatially Region Interaction Techniques).* SPRITEs build on past work exploring spatial interaction techniques such as gesturing, using keyboards, for sighted users [*e.g.,* 32,36]. For example, GestKeyboard uses the keyboard surface to enable touch-screen like gestures on an ordinary keyboard [36] while Taylor *et al.'s* mechanical keyboard [32] and Ramos *et al.'s Fingers* technique [28] sense motion over the keys using IR sensors. However, neither applies these concepts to nonvisual interaction, nor to providing spatial information in a tactile fashion.

SPRITEs also build on past work in representing structural and spatial information in nonvisual interfaces. Automatic extraction of structure was first pioneered in systems such as Mercator [22] and EmacsSpeak [28], and today's screen readers also support access to the document hierarchy. However, most commercial screen readers limit this to simple page elements (such as headers, links, lists *etc.*), and, using these features requires command memorization [2,3]. Also, GUIs naturally take advantage of Gestalt psychology principles [18,34] such as grouping similar items (proximity), or placing items in familiar locations in a consistent fashion (similarity). While Gestalt psychology also applies to auditory and tactile grouping of elements [4,7], the setup of the modern day screen reader typically does not support these principles, making tasks like search more difficult [2,3].

SPRITEs is a suit of techniques that provide quick, usable, and rich access to web GUIs, thus reproducing the perceptual benefits of a spatial layout for sighted users for a non-sighted individual. We demonstrate the value of SPRITEs for non-visual access to web content conveyed implicitly to sighted users *via* spatial layout. Examples of interface elements with this property include menus, tables and maps. Our validation shows that access to SPRITEs in addition to a screen reader more than triples task completion rates for spatial tasks.

CHI 2018, April 21–26, 2018, Montréal, QC, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-56206/18/04…$15.00
https://doi.org/10.1145/3173574.3173585

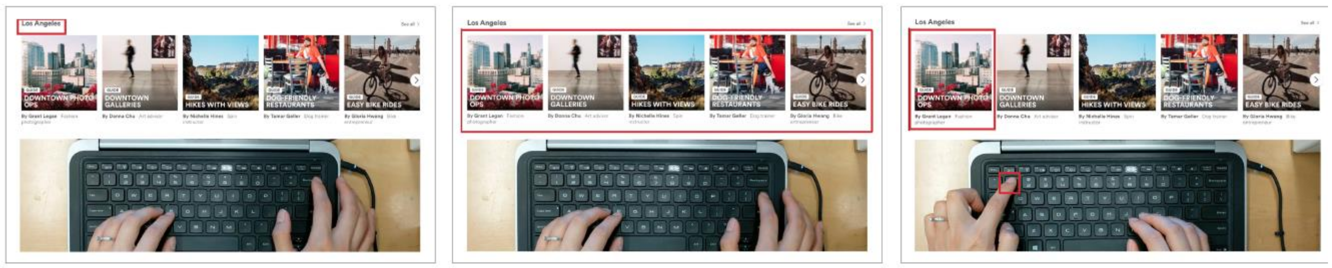

Figure 1: (left) AirBnb (https://www.airbnb.com/s/places), with the initial selection chosen by SPRITEs highlighted when the user presses the topmost key in the rightmost keyboard key. (**middle**) When the user presses '\' key, SPRITEs reads out "menu" and double pressing '\' activates the menu on the numeric row. **(right)** Pressing the '1' key on the numeric row reads out the first element in the menu.

Example Usage Scenario

In this section, we walk through an end-to-end example of SPRITEs. Jane is interested in traveling over the holidays, so she opens a web browser and loads the AirBnb 'Places' page (See **Figure 1**), and presses 'ctrl+w' to enable SPRITEs mode. The SPRITEs server, running in the background, automatically analyzes the page and extracts hierarchical information about structure, which is mapped onto the column of keys at the rightmost side of the keyboard. All similar web elements are mapped to the same location on the keyboard for consistency (*e.g.*, lists or any grouped items on a page always mapped to numeric row of keyboard) that helps the user build a mental model of the interface.

Jane positions her right hand at right edge of her laptop to find the rightmost column and presses the topmost key (corresponding to the red box in **Figure 1 (left)**). SPRITEs outputs "Header". When she presses the key again, it outputs "Los Angeles" (in audio). She presses the next key ('\'), and hears "menu".

When she presses '\' again, she is indicating to SPRITEs that she wants to explore the contents of the menu. SPRITEs informs her "menu activated on number row", meaning that the contents of the menu are now associated with the keys of the number row of her keyboard. Note that the very top row of keys is not remapped because it controls speakers and other important functions. Jane can now move her left hand to the top left corner of the laptop keyboard, and find and press '1' to hear the first menu item. SPRITEs responds "Guide: Downtown photo ops, By Grant Legan, Fashion Photographer".

Jane can continue to explore, interrupting the spoken output with another key press at any time if she is not interested. If at any point she wants to hear what else is on the web page, she can use her right hand to press the next key in the right hand column ('return') and she will hear "Header: Havana". She can change her mind and go back to the menu, which is still mapped to the number keys, or move on and map something else by double pressing. An audio/video description of this example can be found the in the accompanying video.

Overview of Contributions

We used iterative design to identify hardware and interactive requirements for SPRITEs. Our implementation of SPRITEs supports properly marked HTML pages, such as most web pages in Wikipedia, many blogs and other accessible content without extensive scripting.

We implement SPRITEs for page content browsing (the scenario described above), tabular browsing and searching, menus and map searching. SPRITEs is intended to be used in conjunction with a screen reader, and thus does not need to support everything on a web page.

We validate SPRITEs in a study with 10 blind users. We compare SPRITEs to each participant's preferred assistive technology (screen reader, screen magnification software). SPRITES outperformed the screen reader. Task completion rates were equal or better than the participant's own technology for every single task. Also, three times as many participants completing difficult spatial or hierarchical tasks (involving menus, tables and maps) with SPRITES as with their own technology, despite years of experience with their own screen readers (mean=17, mode=15). The difference in task completion success was significant, and very large (M=7.7 tasks completed on average with SPRITES and only 4.3 with the participant's own technology). Participants preferred SPRITEs because of its ease of use and ease of understanding the content.

Our work opens up opportunities to improve existing assistive technologies and enable better nonvisual access to the web, particularly by conveying visual-spatial cues.

RELATED WORK

One of the reasons GUIs have been so successful is that they replace memorization of commands (*e.g.,* to be used at the command line) with the much easier task of recognizing commands (*e.g.,* visible in a menu or toolbar) [6]. Icons, menus, and so on provide cues about the action they can perform. Non-speech audio, Braille displays, and vibro-tactile displays can support iconic information, quantities, or context using rhythmic pulses [5,29,33]). These techniques can help overcome the linear and ephemeral nature of audio by adding context (such as indicating nearby content [22]) or using loud tones for available form fields, and muffled ones for grayed-out portions [11,22]. However, all of these techniques are limited by the linear (or potentially hierarchical) ordering imposed by screen reading.

An alternative is needed to support exploration and interaction with two-dimensional information. Braille displays can support exploration of on-screen content using a small number of braille characters in each dimension [*e.g.,* 26], but more importantly, they and other tactile approaches allow interaction techniques such as edge projection [*e.g.,* 10,16]. For example, *Access Overlays* enhance the nonvisual touch screen experience by preserving spatial layout during search [16]. An example *Access Overlay* is *Edge Projection,* which uses horizontal and vertical edge menus (easy to find due to the physical cue of the tablet edge) to map each on-screen point of interest (POI) to a corresponding "edge proxy." Edge projection is also commercially available in some Braille displays [10].

Tangible nonvisual interaction techniques have also been a focus of recent work. Studies show that tangible interaction techniques can provide significant benefits to users [2], however such approaches require setup, maintenance, and potentially transport of additional devices. For example, FingerReader is a finger-worn device to assist reading printed text and perform functions such as non-linear text skimming [31]. The work shows the benefits of tangible interaction techniques and its viability with an 18 month long user study, however requires a wearable device. Similarly, passive tangible devices [*e.g.,* 17] are easier to transport but still may be hard to keep track of. Other audio-tangible techniques are application specific, such as customized for maps [35].

To summarize, there appears to be an untapped opportunity to improve traditional desktop/laptop computing. New interaction techniques for spatial and tangible interaction have shown great benefit, but either cannot work without a touchscreen, lack deployability, or lack the portability of a laptop.

ITERATIVE PROTOTYPING

To explore the design space for SPRITEs, we built two experience prototypes, inspired by past examples of spatial interaction at the keyboard surface. One prototype explored interaction techniques and hardware for sensing finger position over the keyboard using a Wii, and the other explored options for additional contextual feedback using a hand-worn vibro-tactile device. Our intent in designing these prototypes was to explore interaction parameters for SPRITEs at the physical (hardware) level. As we will describe at the end, our tests ultimately showed that SPRITEs could function effectively with the simple low-fidelity solution provided by an unmodified keyboard. However, the prototypes helped us to formulate interaction requirements that influenced the final design. Both prototypes were implemented and studied in parallel.

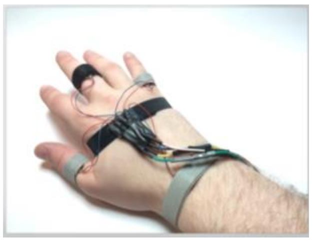

Figure 2: A Velcro band with the four tactors attached at different positions on the hand.

Tactile Contextual Information

Our first prototype explored the potential for tactile feedback. Gestalt psychology can be used in auditory and tactile grouping of elements [4,7]. Non-speech audio, Braille displays, and vibro-tactile displays can potentially convey context information such quantities, as well as identity, using rhythmic pulses [5,30,33]). We developed a hand-worn vibro-tactile device, consisting of four tactors as shown in Figure 2. Our implementation used vibration motors mounted on the hand with velcro, and controlled by an Arduino microcontroller.

Past work on tactile displays for the blind suggested the promise of this approach [*e.g.,* 15,19]. Contextual feedback could indicate direction (*i.e.* to find a specific item the user has searched for, essentially a 'tactile tutorial'); indicate the location of search results on a page; provide virtual landmarks to aid situational awareness; or allow the user to explore a map or trace a route with turn-by-turn directions.

To indicate contextual information, we implemented support for using varying combinations of vibrations and rests [5]. We created auditory perceptual groups [13], and tested perceptual success for a variety of note lengths, ranging from 500ms to 1000ms. The patterns consisted of four notes, each varying in length from 1/8th to a full note length. However, we observed that short note lengths ran together, an effect likely related to the spin-up and spin-down time of the motors, whereas longer note lengths were easily distinguishable.

We next explored whether patterns could easily be distinguished from each other. However, the notes were perceived as individual elements, and did not contribute to a unique whole. Also, it was very difficult to keep track of individual notes when presented in longer sequences (> 2). When the information was presented slowly enough that the notes were individually perceived, the user would be required to keep a track of all notes, which makes it unsuitable for practical use. Moreover, using the device for more than several minutes was somewhat uncomfortable, creating sensations such as itches, tickles, and illusory vibrations. This led us to conclude that tactile feedback, at least on the hand, would not be a suitable option for contextual information.

Fingers Prototype

Our *Fingers* prototype was based on an existing Wii-based finger tracking system [29]. *Fingers* uses a glove with an

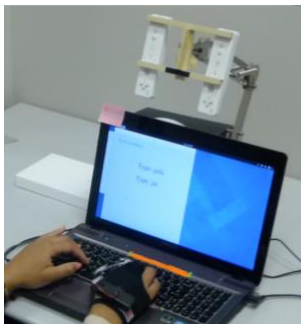

Figure 3: Fingers prototype. The hand-wearing glove with infrared LEDs is being tracked by two Wii remotes.

infrared light emitter on the pointer finger and two Wii-motes to track its movement across a keyboard as shown in **Figure 3**.

For testing purposes, we built a custom eCommerce application with different web elements on different pages. The website followed common eCommerce website design patterns including a sidebar menu containing categories, a search bar, a grid product view spanning multiple pages, and individual product pages.

We added a semi-transparent interaction layer to each webpage. This interaction layer was broken up into *boxes*, corresponding to a specific area of the screen containing information about a category or product. When mapping boxes to keyboard keys, we aligned grouped elements on the interface, such as a side menu bar and a product grid, with horizontal rows on the keyboard.

Touching a keyboard key (without pressing it) plays audio describing the associated box. Pressing a keyboard key invokes the action of the associated box (*e.g.,* click a link). Thus, *Fingers* provides immediate feedback while exploring a webpage. A user can skim their fingers over the keyboard (without pressing keys) to listen to the associated content, similar to how a visual system can see what is on screen without pressing anything.

Formative Study with Blind Participants

To test our interface, we recruited 5 visually impaired participants (all female) *via* word of mouth. Participants' ages ranged from 24 to 61 (mean= 44.2, S.D. = 17.36). The tasks were designed to emulate common shopping use cases (using our eCommerce application). Participants completed two blocks of four tasks: one block using *Fingers*, and one using a screen reader.

Results

Participants struggled to position their hands correctly over the keyboard, and were often confused about the position of their hands. In addition, we expected *Fingers*' ability to sense both hovering and pressing to be an advantage, as described above. However, due to the need to repeatedly determine hand position with respect to the keyboard edges, and the F and J keys (which have tactile marks on most keyboards), multiple touch interactions were activated as

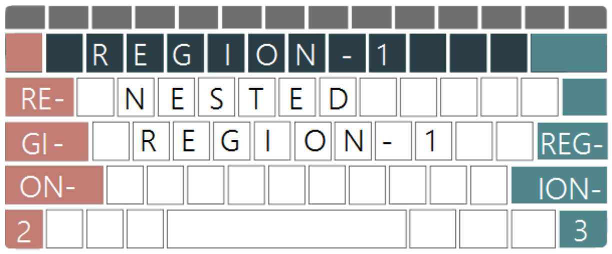

Figure 4: Keyboard layout showing different regions that we used in our implementation of SPRITEs

participants' fingers brushed along the keys. In addition to the disadvantages of reacting to finger position, participants complained that the glove-like device they had to wear muffled their sense of touch.

Although the system was not successful, there were three key design takeaways: (1) the edge of the keyboard and the F, J keys are of critical importance to orient and position the hands on the keyboard; (2) the system should facilitate easier understanding of the page structure; (3) *Fingers* was overly sensitive and reactive and impeded the sense of touch. Based on these observations, we redesigned the interaction techniques to address (1) and (2). Although a different sensing approach is possible, we decided to replace *Fingers* with a standard keyboard based on (3).

Discussion

Our prototypes explored a variety of rich implementation options for nonvisual spatial interaction at the keyboard surface. The *tactile contextual information* prototype failed partly due to overlapping notes, phantom vibrations, and the need to keep a track of longer notes. It revealed the need for a system that can facilitate easier understanding of the content, which was reaffirmed in *Fingers* study as well. *Fingers* additionally showed the critical importance of designing interaction techniques that could leverage natural keyboard landmarks (*e.g.,* the edges of the keyboard). It also demonstrated that providing early feedback (*e.g.,* during hovering) could be overwhelming.

Both probes additionally ran into human factors issues due to wearing something on the hand. Between illusory and uncomfortable sensations (caused by the tactile *contextual information* probe) and a muffled sense of touch (noted in *Fingers*), this indicates that a wearable solution is not ideal.

Based on these facts, and lack of inexpensive access to custom hardware for end users, we focused on a much simpler, software-only solution. As an additional benefit, a software approach can be more easily and widely deployed due to the lack of custom hardware.

SPRITES DESIGN

SPRITEs, which is deployable on commodity hardware, provides holistic support for a specific domain – web browsing. Thus, we designed the interaction techniques available in SPRITEs to function together as a cohesive

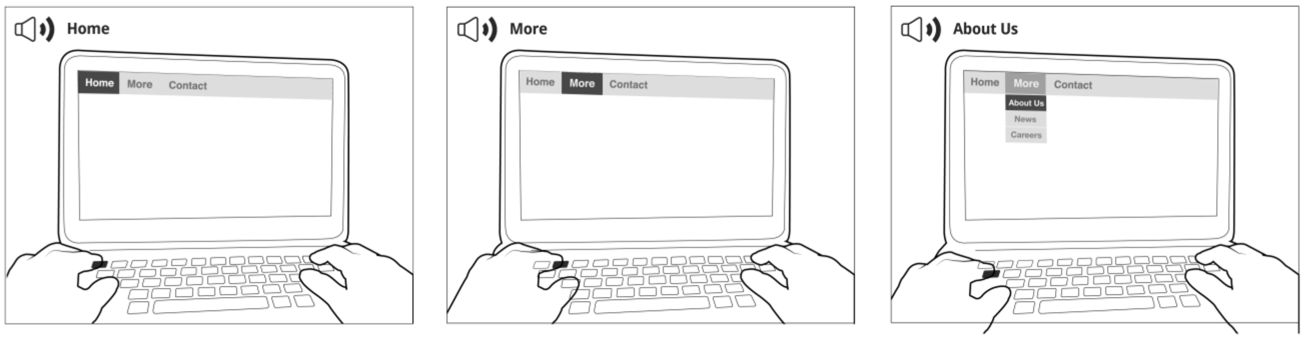

Figure 5: A menu bar (top of screen) is mapped onto the numeric row of keys. **(left)** The user selects the first menu **(middle)** The user selects the second menu **(right)** The user has opened a sub-menu, which is now mapped onto the next row.

whole, in concert with each other and a screen reader. The focus of SPRITEs is to improve:

- *Exploration* (browsing) of a web page and its spatial elements.
  - Scrolling and browsing the whole webpage.
  - *Interaction with ordered groups of items*, such as lists and menus. These may be hierarchical.
  - *Interaction with 2D elements*, such as tables.
- *Search within 2D elements*, such as tables and maps.

SPRITEs is a backend service accessible *via* a keyboard mode. A simple key combination can be used to switch between the default keyboard behavior, and SPRITEs. When SPRITEs is active, it creates and maintains a mapping between web page element and keyboard keys. Pressing a key once reads out any information associated with its corresponding web element. A double press invokes the action associated with that element.

The SPRITEs mappings are designed to leverage the edges and corners of the keyboard as much as possible so that it is easier for the user to find keys. The topmost row of function keys (above Region 1 in **Figure 4**) is reserved for browser level controls such as closing a tab, which are independent of a particular webpage, and should be available to the user at all times, and hence is not used in our techniques.

Exploring a web page

When a user arrives on a new webpage, the rightmost column of keys in the keyboard (Region 3 in **Figure 4**) is used for scrolling through elements within that page. We chose the rightmost column because it allows the user to hold onto the edge of the keyboard. This enables users to keep a track of which key was last pressed, and move up and down in the column easily.

The HTML elements in a webpage (header, div, *etc.*) are parsed and mapped to each key in Region 3 in **Figure 4**. Pressing a key once speaks the type of the object. Pressing it twice invokes that object's associated action. For example, if it is a text block, the content may be spoken, or if it is a table, it projects the table widget onto other parts of the keyboard (described later).

A webpage can contain more elements than the number of keys available in one column, so the first and the last key in Region 3 are used to scroll back and forth between the elements on the page. This loosely mimics the idea of scrolling to browse content. The state of the "scrollbar" is saved if a user switches to another task (using another SPRITEs technique or the default keyboard) and back again. We use this moving back and forth feature in all SPRITEs discussed below. The ability to scroll through a large amount of content using the same set of keys makes SPRITEs scalable to any amount of content on a webpage.

Having a consistent mapping on the right side of the keyboard allows the user to easily find and learn the location of the elements on a page. The user can quickly focus on and interact with any of them (activated *via* double key press).

Interaction with Ordered Groups of Items: Grouped items that a user might encounter include menu items (which may be hierarchical) and search results. Grouped items can leverage the proximity and alignment of keys within a row on a keyboard to convey grouping, and from one row to the next to convey hierarchy. When a grouped item is activated (such as double pressing on a menu bar when scrolling), SPRITEs project the group of items onto the numeric row of the keyboard (Region 1, **Figure 4**). If there are more elements in a menu or a sub-menu than the number of keys in that row of the keyboard, the first and last key of the row are reserved for scrolling.

Suppose a user searches a map. The list of results is projected onto the numeric row of the keyboard. When a key is pressed once, it reads out the title of the search result (*e.g.,* restaurant name). A double press reads out the contents of the item (*e.g.,* address).

If the contents are themselves a sub-group (*e.g.,* a menu item that opens a sub-menu), the contents are mapped onto the next row (Nested Region 1, **Figure 4**), thus making the hierarchical association between the two groups tangible. The mapping in a row is retained until the interaction finishes, or the user selects a new group or sub-group. Thus the user can easily continue exploring the original group, or

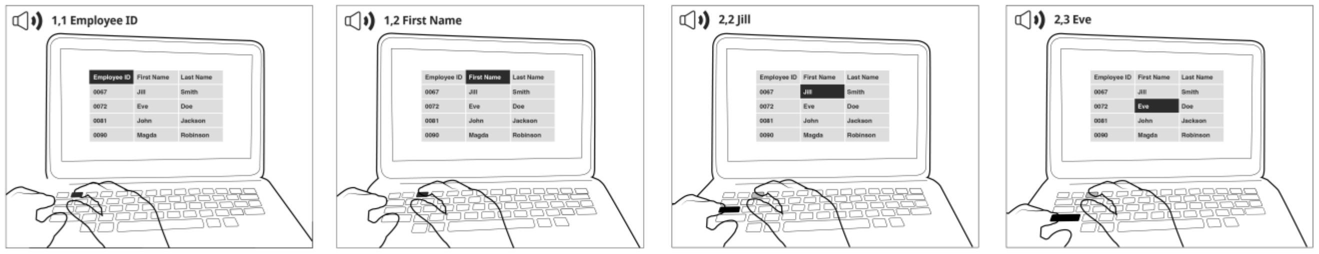

Figure 6: A table (shown on screen). Columns are mapped to the number row of the keyboard and rows to the leftmost column of keys, and (1) By default the top left cell is selected. **(2)** The right hand presses the '2' key, selecting the second column **(3)** The left hand selects the next row **(4)** The left hand selects the third row. **In each case, the position of the cell and its content are read out aloud.**

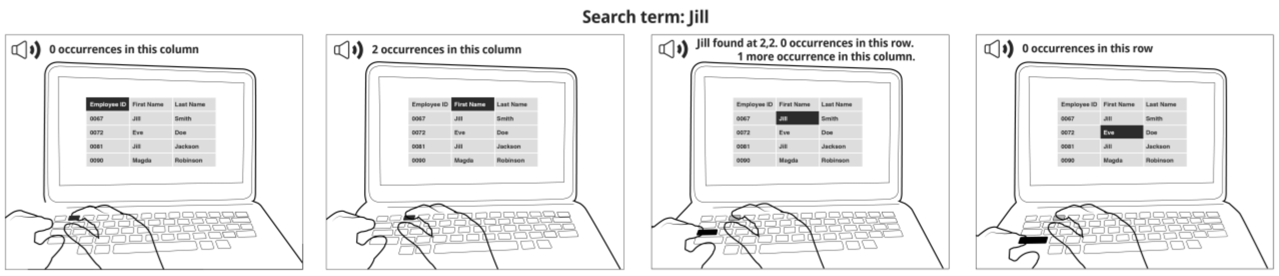
Search term: Jill
0 occurrences in this column
2 occurrences in this column
Jill found at 2,2. 0 occurrences in this row. 1 more occurrence in this column.
0 occurrences in this row

Figure 7: A user is searching a table (shown on screen) for the word 'Jill'. Columns are mapped to the number row of the keyboard and rows to the leftmost column of keys. (1) By default the top left cell is selected. **(2)** The right hand presses the '2' key, selecting the second column **(3)** The left hand selects the next row **(4)** The left hand selects the third row. In each case, the number of occurrences of the search query in the respective column or row are read aloud. When the query is found, the position and content of the cell are read out aloud.

switch back and forth. **Figure 5** shows a person accessing a menu using SPRITEs.

Interaction with 2D elements: Although a keyboard is not exactly a perfectly square grid, the letter keys do form a rhomboid grid. Thus, with a small perceptual adjustment, the keys can be thought of as specifying 2D location. However, based on our formative work, finding elements in the middle of a table may be difficult. Thus SPRITEs instead maps *table indexes* (row and column numbers). As illustrated in **Figure 6**, column numbers are mapped to Region 1, the number row of the keyboard (so that moving the right hand right to left moves across the columns right to left) and row numbers are mapped to Region 2, the leftmost column of the keyboard (so that moving the left hand top to bottom moves through rows top to bottom). When there are more rows in the table than the number of keys in **Region 2**, or more columns than the number of keys in **Region 1**, the end keys are reserved for scrolling.

By default, row 1 and column 1 of the table are selected. Pressing a column or row key once changes the selection to the corresponding column or row of the table. Changing the column selection does not change the row selection (and *vice versa*). Pressing a key a second time (*i.e.* after it is selected) reads out the currently selected row and column along with the contents of the corresponding cell.

One advantage of this design is that the user can intuitively and directly access any of the internal cells by pressing the corresponding row and column key.

Enabling Search of 2D elements with Edge Projection

When two-dimensional structures such as tables and maps are searched, most web browsers mark the location of search results in some visually salient way (such as map markers or highlighting). Past work has shown that for nonvisual interaction, edge projection carries similar benefits [16]. However, this has not been tested outside of touchscreen. Because SPRITEs provides spatial information about tables and maps, it too can support edge projection, by mapping search results to column and row indexes.

Search of Tables: When a table has been searched, pressing a column or row index key tells a user whether any search results are in that particular row/column, and if so, how many. For example, suppose the user in **Figure 7** wants to know which rows contain matches to the search term 'Jill'. She presses the key for column one, and is told "0 occurrences of Jill in this column". When she presses the key for column two (which has two cells that contain the search query) SPRITEs says "2 occurrences of Jill in this column" She can continue exploring the column, press keys for each row to identify the exact locations of the search results. For example, after pressing the key for the second row, which matches the search result at (2,2), she hears "Jill found at 2,2. 0 occurrences in this row. 1 more occurrence in this column." Pressing that key a second time plays the whole contents of the cell. A user can easily keep track of multiple rows or columns that matched with multiple fingers and switch between them to hear and compare search results.

Experimental Map Search: As described earlier, the standard map search returns and ordered group that is mapped to Region 1. We also implemented a map search that gives 2D information using edge projection.

The part of the map containing search results is divided evenly into X columns and Y rows, where X is the number of keys available in Region 1, and Y is the sum of keys available in Region 2 and 3. Search results are assigned to columns and rows based on which column they fall into. Odd numbered street addresses map to Region 2, and even to Region 3. Unlike table search, both Region 2 (the left side of the keyboard) and Region 3 (the right side) are used. This helps the user to understand which search results are on the same side of the street as each other. Further, if the user knows a building, this helps them locate other buildings with respect to it.

Note that in contrast to the other techniques, which are fully operational, this implementation is a prototype that was sufficient only for our study, and is based on a fake GPS location. The purpose of designing Edge Projection on Maps in such a manner was to explore the use of auxiliary data (*e.g.,* GPS) in conjunction with SPRITEs. While, our example is rather elementary and warrants further exploration, it was good enough to user test our idea as shown in the user study results.

Implementation

The goal of our implementation was to support the study described next (study tasks are listed in Table 2), and to demonstrate the viability and deployability of SPRITEs. Thus, we did not tackle the problem of inaccessible web pages or dynamic web content support, nor do we include a technical validation of the breadth of web pages supported by our implementation. However, our implementation is fairly general and functional for accessible web pages where content is available in the DOM (*i.e.* web pages that do not use AJAX and other styles of dynamic content). In particular, we can support websites such as Wikipedia (which we used in our study).

The only interactions described above that are not fully supported are both map search variations. We use the Google Maps API to send queries and parse the response received in JSON format, using a fake GPS location. Additionally, for map search results presented as an ordered group, our implementation does not support nested information retrieval (*e.g.,* reviews). For the map search results presented using edge projection, assumptions made about orientation were highly task specific and not general.

SPRITEs is implemented using Python in Firefox. When a page is loaded, it retrieves the DOM and simplifies it by removing tags that do not contain structural information. Tags such as headers, text blocks; tables; and lists are kept. Once the DOM is simplified, each element of the webpage is automatically associated with a particular interaction approach based on the type of its encapsulating HTML tag. We store this information in a dynamically created dictionary like data structure that also keeps a record of keyboard key-to-content mapping and current selections. These mappings may be nested depending on the structure of the page element. SPRITEs uses a keyboard hook to monitor keystrokes. It listens for mode switch requests and once it is invoked, looks up key presses in the data table to decide how to respond. The keyboard hook also allows the framework to control what function receives the keystroke and turns off the default keyboard functionality while SPRITEs is active. A python wrapper for a text-to-speech engine provides the audio output.

Since our primary goal was to test the value of SPRITEs for end users, we did not validate the generality of our framework in terms of its robustness within Wikipedia or across other similar websites. Instead, we evaluated the benefits of our approach for end users over a traditional screen reader (or other preferred assistive technology), as described in the next section.

VALIDATION OF SPRITES

We conducted a study comparing the performance of SPRITEs to each participant's preferred accessibility tool. A secondary goal was to explore how SPRITEs impacted participants' understanding of webpage organization and spatial layout. We recruited ten visually impaired participants for the study, including three low vision participants who used screen magnifiers, *via* word of mouth. Participants' years of experience with their assistive technology ranged from 6 to 32 (mean= 17.2, S.D. = 8.83). Participant details are shown in Table 1.

Method

To compare the participants' own setup to SPRITEs, we used a counterbalanced, within subject design. Participants completed one block of eight tasks with their own assistive technology (**PAT** condition), and one block of eight tasks using SPRITEs. The eight tasks are shown in Table 2 and were designed to include both target acquisition and questions about structure and organization. They were also

ID	Age	M/F	Impairment	Preferred Tech (#years)
P1	24	F	Low Vision	ZoomText (6)
P2	46	F	Blind	JAWS (15)
P3	29	F	Low Vision	MAGic (8)
P4	61	F	Blind	JAWS (20)
P5	61	F	Blind	JAWS (15)
P6	42	M	Blind	JAWS (14)
P7	69	M	Blind	JAWS (25)
P8	54	M	Blind	JAWS (32)
P9	59	M	Blind	JAWS (28)
P10	52	M	Blind	ZoomText Magnifier/Reader (9)

Table 1: Summary of participant data that took part in our evaluation. #years is years of experience with that technology.

Task	Description	PAT	SP
W1	**Webpage Search**: Find a specific section header.	10	10
W2	**Webpage #**: Count the # of headings in a page.	10	10
M3	**Menu Interact**: Given the name of a menu, find all the items in that menu. Next, find all the items that have a sub-menu and list one of the sub-menus	3	10
T4	**Table Interact**: Given a target cell, find the value in the cell.	3	9
T5	**Table Search**: Given a value, find all the occurrences of that value in a given column/row.	3	10
W6	**Webpage relocate**: First, find a particular target within a page. Next, start from the top of the page again and relocate the same target.	10	10
N7	**Navigation Search**: Find a specific point among multiple points on the map.	2	9
N8	**Navigate**: For a given map in street view, find two specific points on the left side of the road and two points on the right side of the road.	2	9

Table 2: Tasks and the number of participants who completed them. SP = SPRITEs. Tasks are shown in the order participants were asked to complete them.

selected to include tasks that require understanding of both spatial and more linear or hierarchical components.

Tasks were always assigned in the same order. To vary the content between conditions, three different Wikipedia pages with similar structure and content organization were used. The Wiki pages were randomly assigned to each technique block. For each task, we recorded the start and end time (when the participant thought they were done) and key press events. From this we calculated the time to complete the task. If a participant failed to start, or complete a task, we discarded their time data.

We also recorded subjective ratings and general feedback from the participants. Because the webpage relocate task (W6) has two distinct steps, we recorded times for each step as W6a and W6b.

After each block, participants rated the condition (**PAT, SP**) on a seven point Likert scale. Participants rated their experience with each task separately, because condition performance varied by task. At the end of the session, the participants ranked the techniques for different kinds of tasks (searching, and interacting/browsing) and provided qualitative feedback. All questionnaires were administered verbally and the participant feedback was recorded using a combination of recording audio, and pen and paper. Each participant session took approximately 2 hours to complete.

We used a laptop running Windows 10 with a screen size of 12.5 inches. The size of the physical keyboard was 11.5 x 28.5 cm with each of the A to Z being 1.5 x 1.5 cm. A time stamped activity log was collected for each of the tasks.

Results

The most striking impact of SPRITEs can be seen in the task completion numbers shown in Table 2**.** In five of the 8 tasks, participant ***task completion rates were three times higher*** using SPRITEs than using their own technology of choice. Additionally, every single participant completed more tasks with SPRITEs than with their own technology (no participant completed all tasks in the **PAT** condition, eight out of 10 did in the **SP condition**). This is despite the fact that participants had never used SPRITEs before. The difference between task completion rates for the **SP** (M=7.7, SD=0.7) and **PAT** (M=4.3, SD=1.82) was highly significant in a paired samples t-test (t=5.67, df=9, p=.0003).

The three tasks for which completion rates were high with both devices (10/10 for both) were the three tasks with the least spatial nature (W1 & W2 searching and counting headers within a page, and W6, searching and then finding the same item again). We discuss speed differences below, but could not conduct a statistical analysis due to the low overall task completion rates.

W1-2; W6a & b: All participants were able to complete the webpage search and count tasks (W1, W2) in both conditions. Finding the same element twice (W6a&b) was also an area where screen readers were fast and effective. The average time taken in the **PAT** condition was lower than SPRITEs by an average of 29 seconds in these tasks. This result was not surprising, since screen readers are optimized for navigating page hierarchy, and our relatively experienced users had no trouble using a shortcut key to jump sections/headings.

It is worth noting that when using SPRITEs, participants have a much faster average completion time in W6b than W6a. In contrast, times in the **PAT** condition stay about the same. W6b is an exact repeat of the search task W6a. In W6b, participants already had an idea of how far down the page to scroll. We observed that participants were confidently able to skip past certain content (scroll down) without listening to it, and hence were able to relocate the element faster.

M3: Only three out of ten participants (P1, P3, P8) completed M3, finding the structure of a menu, in the **PAT** condition. Out of these three participants, two used a screen magnifier to visually inspect the menu. One participant was able to complete the task using only a screen reader. Multiple participants thought they had succeeded in the **PAT** condition, even though they did not correctly understand the hierarchy of the menu widget. For example, they had difficulty differentiating the sub-menu items from the main menu, or missed a sub-menu. While the screen reader did inform the user about the beginning and end of a sub-menu, the participants were unable to keep track of the entire menu structure with just the linear audio output of their assistive technology. Using SPRITEs, average time was as fast as the two screen magnification users and screen reader user who completed the task.

T4-5: Only three participants (P3, P8, P10) successfully completed T4-5, finding the value of a table cell, and searching for a value in a table, in the **PAT** condition. Both are known to be difficult tasks with a screen reader, and we observed that participants struggled to understand row and column spans, the organization of table content and its structure. Participants were frustrated with their experience

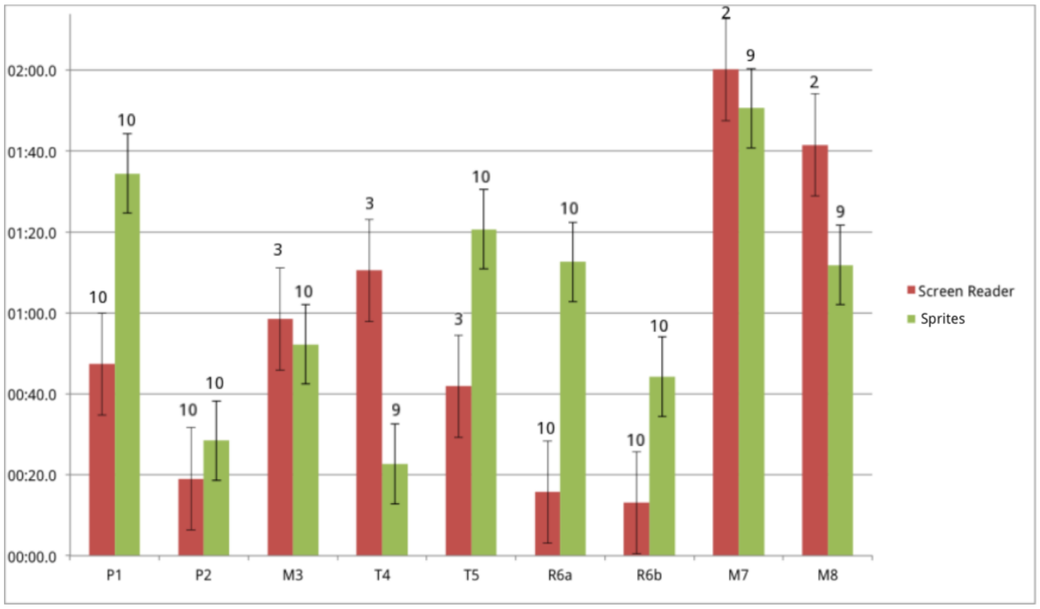

Figure 8: Average task completion times for all of the tasks in the study. Error bars show standard error. Column labels show how many participants completed task. Averages only include completed tasks. For example, the **PAT** average for T5 only includes three numbers.

with tables in the **PAT** condition. P2 said, "*The table is not nicely laid out. You are counting in your head, it's as if somebody took a table out and put it in lines underneath it which is horrible for locating something in a particular column.*" This lack of structural information severely lowered task completion rates in the **PAT** condition.

Using SPRITEs, all but one participant was able to complete both tasks. The participants were able to understand the row and column spans for tables, and were able to browse through the table structure easily. This is also reflected in the task timings, as the participants were much quicker in using SPRITEs when completing T4. Participants were much slower in T5 (searching the table, with search results projected out to the edge of the table) as shown in Figure 8. Our observations suggest this is because some of the participants got confused due to the change in how the table was described during searching. However, it is important to emphasize that despite the slow speed, far more participants were able to complete the task using SPRITEs than in the **PAT** condition, and participants rated it as the superior interaction method. With experience, we believe that participants would become as fast with table search (T5) as table exploration (T4).

N7-8: Only two participants in the **PAT** condition were able to complete the navigation tasks: N7 (P3, P10) and N8 (P1, P3). Most participants did not even attempt the navigation task (N8) after the screen reader failed to work seamlessly with maps in N7. This was not unexpected given that this is known to be a difficult domain for screen readers.

Qualitatively, participants preferred **SP** (M=4.68, SD=1.2) to **PAT** (M=4.2, SD=2.05) for most tasks, on the 7-point likert scale we used. The high rating for the screen reader despite low task completion rates may reflect the participants' comfort and familiarity with it. Participant P10 said that even though he can understand the benefits of SPRITEs, he would rate his setup (screen magnifier + screen reader) slightly higher due to the learning curve associated with any new assistive technology. Our participants showed varying degrees of skepticism towards adopting a new technique.

Participants appreciated that SPRITEs enables them to understand the layout and structure of the webpage elements. A common complaint across participants was their inability to remember all possible screen reader shortcuts despite years of experience. This was evident in our observation during the user study tasks as well. Participants did not use some shortcuts that would have allowed them to accomplish tasks in tables. Similarly, we observed that while screen readers do inform the users about the structure of web widgets such as menus, this did not scale to whole pages. In the **PAT** condition**,** participants struggled to remember the whole layout.

DISCUSSION

Our results demonstrate that SPRITEs is particularly valuable for tasks that have traditionally been difficult to support—namely interaction with web page elements that have a natural 2D structure, such as menus, tables and maps. This shows up most dramatically in task completion rates, which were triple those of participants more familiar technology for spatial tasks.

In contrast to task completion rates, participants were almost a minute faster using their own technology (**PAT** condition) for non-spatial tasks. This is not surprising, given that SPRITEs was an entirely new technology. Since only 2 or 3 participants out of 10 completed many tasks, the task times of others were not included in our analysis of task completion times. However, at best our study shows that a

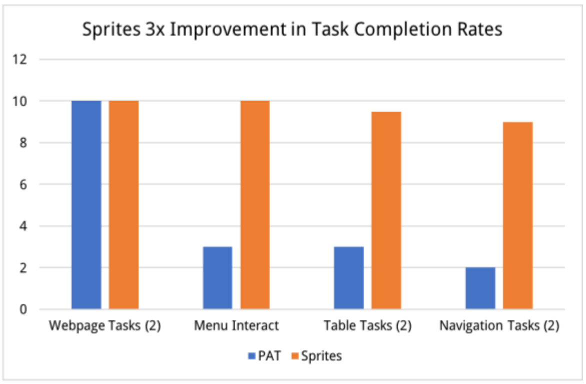

Figure 9: Graph showing task completion rates for different kinds of tasks in our user study

few expert screen reader/magnification users are faster than many novice SPRITEs users.

In fact, our study shows that for difficult spatial or hierarchical tasks, SPRITEs is three times better than the screen reader on the task completion metric as shown in Figure 9. Overall, it is equal or better *on every task we studied*, and the difference in task completion rates is highly significance, despite a small sample.

What is more surprising is that SPRITEs was faster in four out of five spatial tasks. This despite the fact that it was essentially only being compared to the two or three most capable users in each of these tasks (those who completed the task in the **PAT** condition), two of whom (in most cases) used screen magnification. In addition, participants had much less opportunity to learn SPRITEs than they had spent learning to use their preferred technology. Thus, SPRITEs is very competitive in terms of speed.

Our observations of SPRITEs use, as well as participant performance on W6b, indicate that participants were successfully developing a mental model of the correspondence between keyboard keys and spatial or hierarchical structure of the document. This is a promising sign that SPRITEs may be able to help improve a user's mental model of the interface.

SPRITEs also supports non-linear interactions, such as random access to interface elements. For example a menu item can be directly selected if the user knows its location. The consistent location of menu items (which is the same each time the user visits a page) also supports proprioception. Further, SPRITEs takes advantage multi-finger interactions in supporting random access. For example, a user can explore a table, 'leaving' fingers at important rows and columns to easily return to them.

FUTURE WORK

The success of GUIs is much studied, and the best direct manipulation interfaces [14] support the brain's ability to forage for information [9,24,25], recognize rather than recall information [6], form mental models [12], and make connections between related items [4,7]. It is a sort of holy grail for nonvisual interfaces to provide the same support.

In future work, we would like to expand SPRITEs to include more context cues. Context can help to overcome the linear and ephemeral nature of audio (such as by indicating nearby content [23], or using loud tones for available form fields, and muffled ones for grayed out portions [22]). This work could explore approaches for summarizing text moving during scrolling, highlight the presence of search terms during a scroll, or indicate which locations are available for interaction.

We also plan to explore a wider array of interaction techniques in SPRITEs. For example, pressing and holding a key, with appropriate audio feedback, could accomplish scrolling quickly. Radio buttons, checkboxes, forms, and other types of special interaction cases might benefit from new interaction designs using SPRITEs. It would also be valuable to move beyond the multi-finger input currently supported to take advantage of the benefits of true bimanual interaction [21]. Finally, adding support for things like drag and drop would be valuable.

SPRITEs could also be extended to work with other input devices, such as repurposing a track pad as a gestural input device, such as swiping between page elements [1] or adding support for a force feedback mouse to provide interaction cues [*e.g.,* 19].

CONCLUSION

Loss of spatial layout is one of the biggest challenges of nonvisual accessible technology. The impediment of not being able to understand layout deters visually impaired individuals from being fully empowered in today's digital world.

In this paper, we present SPRITEs, an inexpensive paradigm and a suite of interaction techniques that can provide non-visual access to graphical interfaces while preserving spatial layout and improve the end user experience. Our study shows that for spatial tasks SPRITEs task completion rates are triple that of participants' preferred access technology.

ACKNOWLEDGEMENTS

We thank our participants for their efforts and Dr. Mayank Goel for his valuable input into making this work successful. The contents of this paper were developed under NIDILRR grant number 90DP5004-01-00. NIDILRR is a Center within the Administration for Community Living (ACL), Department of Health and Human Services (HHS). The contents of this paper do not necessarily represent the policy of NIDILRR, ACL, HHS, and you should not assume endorsement by the Federal Government.

REFERENCES

1. Faisal Ahmed, Muhammad Asiful Islam, Yevgen Borodin, and I. V. Ramakrishnan. 2010. Assistive web browsing with touch interfaces. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility* (ASSETS '10). ACM, New

York, NY, USA, 235-236. DOI=http://dx.doi.org/10.1145/1878803.1878848

2. Yevgen, Borodin, Jeffrey P. Bigham, Glenn Dausch, and I. V. Ramakrishnan. More than meets the eye: a survey of screen-reader browsing strategies. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, p. 13. ACM, 2010.
3. Mark S. Baldwin, Gillian R. Hayes, Oliver L. Haimson, Jennifer Mankoff, Scott E. Hudson: The Tangible Desktop: A Multimodal Approach to Nonvisual Computing. TACCESS 10(3): 9:1-9:28 (2017)
4. Elvira Brattico & Fiorella Sassanelli. 2000. Perception and musical preferences in Wishart's work. *Journal of New Music Research*, *29*(2), 107-119. DOI= http://dx.doi.org/10.1076/jnmr.29.2.107.3099
5. Stephen Brewster and Lorna M. Brown. 2004. Tactons: structured tactile messages for non-visual information display. In *Proceedings of the fifth conference on Australasian user interface - Volume 28* (AUIC '04), A. Cockburn (Ed.), Vol. 28. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 15-23.
6. Stuart K. Card, Allen Newell, and Thomas P. Moran. 1983. *The Psychology of Human-Computer Interaction*. L. Erlbaum Assoc. Inc., Hillsdale, NJ, USA
7. Dempsey Chang, Keith V. Nesbitt, and Kevin Wilkins. 2007. The gestalt principles of similarity and proximity apply to both the haptic and visual grouping of elements. In *Proceedings of the eight Australasian conference on User interface - Volume 64* (AUIC '07), Wayne Piekarski and Beryl Plimmer (Eds.), Vol. 64. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 79-86.
8. Hsinchun Chen, Andrea L. Houston, Robin R. Sewell, and Bruce R. Schatz. 1998. Internet browsing and searching: User evaluation of category map and concept space techniques. *Journal of the American Society for Information Science, Special Issue on AI Techniques for Emerging Information Systems Applications*.
9. Ed H. Chi, Peter Pirolli, Kim Chen, and James Pitkow. 2001. Using information scent to model user information needs and actions and the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '01). ACM, New York, NY, USA, 490-497. DOI=http://dx.doi.org/10.1145/365024.365325
10. Alistair, Edwards, ed. Extraordinary Human-Computer Interaction: Interfaces for Users with Disabilities. Vol. 7. CUP Archive, 1995.
11. W. Keith Edwards and Elizabeth D. Mynatt. 1994. An architecture for transforming graphical interfaces. In *Proceedings of the 7th annual ACM symposium on User interface software and technology* (UIST '94). ACM, New York, NY, USA, 39-47. DOI=http://dx.doi.org/10.1145/192426.192443
12. Gerhard Fischer. 1991. The Importance of Models in Making Complex Systems Comprehensible in D. Ackerman, & M. Tauber (Eds.), *Mental Models and Human Computer Communication: Proceedings of the 8th Interdisciplinary Workshop on Informatics and Psychology (Schaerding, Austria), Volume 2*, Elsevier Science, Amsterdam, pp. 3-36.
13. Goldstein, E.B. 1989. *Sensation and perception*. Wadsworth Pub. Co.
14. Edwin L. Hutchins , James D. Hollan & Donald A. Norman. 1985. Direct manipulation interfaces. *Human–Computer Interaction*, *1*(4), 311-338. DOI=http://dx.doi.org/10.1207/s15327051hci0104_2
15. Chandrika Jayant, Christine Acuario, William Johnson, Janet Hollier, and Richard Ladner. 2010. V-braille: haptic braille perception using a touch-screen and vibration on mobile phones. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility* (ASSETS '10). ACM, New York, NY, USA, 295-296. DOI=http://dx.doi.org/10.1145/1878803.1878878
16. Shaun K. Kane, Meredith Ringel Morris, Annuska Z. Perkins, Daniel Wigdor, Richard E. Ladner, and Jacob O. Wobbrock. 2011. Access overlays: improving non-visual access to large touch screens for blind users. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 273-282. DOI=http://dx.doi.org/10.1145/2047196.2047232
17. Shaun K. Kane, Meredith Ringel Morris, and Jacob O. Wobbrock. “Touchplates: low-cost tactile overlays for visually impaired touch screen users.” In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, p. 22. ACM, 2013.
18. Wolfgang Köhler. 1970. *Gestalt psychology: An introduction to new concepts in modern psychology.* WW Norton & Company.
19. Ravi Kuber, Wai Yu, and Graham McAllister. 2007. Towards developing assistive haptic feedback for visually impaired internet users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 1525-1534. DOI=http://dx.doi.org/10.1145/1240624.1240854
20. Jonathan Lazar, Aaron Allen, Jason Kleinman & Chris Malarkey. 2007. What frustrates screen reader users on the web: A study of 100 blind users. *International Journal of human-computer interaction*, *22*(3), 247-269. DOI=http://dx.doi.org/10.1080/10447310709336964
21. Andrea Leganchuk, Shumin Zhai, and William Buxton. 1998. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Trans. Comput.-*

Hum. Interact. 5, 4 (December 1998), 326-359. DOI=http://dx.doi.org/10.1145/300520.300522

22. Elizabeth D. Mynatt and W. Keith Edwards. 1992. Mapping GUIs to auditory interfaces. In *Proceedings of the 5th annual ACM symposium on User interface software and technology* (UIST '92). ACM, New York, NY, USA, 61-70. DOI=http://dx.doi.org/10.1145/142621.142629
23. Elizabeth D. Mynatt and Gerhard Weber. 1994. Nonvisual presentation of graphical user interfaces: contrasting two approaches. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '94), Beth Adelson, Susan Dumais, and Judith Olson (Eds.). ACM, New York, NY, USA, 166-172. DOI=http://dx.doi.org/10.1145/191666.191732
24. Christopher Olston and Ed H. Chi. 2003. ScentTrails: Integrating browsing and searching on the Web. *ACM Trans. Comput.-Hum. Interact.* 10, 3 (September 2003), 177-197. DOI=http://dx.doi.org/10.1145/937549.937550
25. Peter Pirolli and Stuart Card. 1995. Information foraging in information access environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '95), Irvin R. Katz, Robert Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen (Eds.). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 51-58. DOI= http://dx.doi.org/10.1145/223904.223911
26. Denise Prescher, Gerhard Weber, and Martin Spindler. 2010. A tactile windowing system for blind users. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility* (ASSETS '10). ACM, New York, NY, USA, 91-98. DOI=http://dx.doi.org/10.1145/1878803.1878821
27. I. V., Ramakrishnan, Vikas Ashok, and Syed Masum Billah. Non-visual Web Browsing: Beyond Web Accessibility. In *International Conference on Universal Access in Human-Computer Interaction*, pp. 322-334. Springer, Cham, 2017.
28. T. V. Raman. 1996. Emacspeak—a speech interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '96), Michael J. Tauber (Ed.). ACM, New York, NY, USA, 66-71. DOI=http://dx.doi.org/10.1145/238386.238405
29. Julian Ramos, Zhen Li, Johana Rosas, Nikola Banovic, Jennifer Mankoff, and Anind Dey. “Keyboard Surface Interaction: Making the keyboard into a pointing device.” *arXiv preprint arXiv:1601.04029* (2016).
30. J.C. Roberts, K. Franklin. 2005. Haptic glyphs (hlyphs)-structured haptic objects for haptic visualization. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint* (pp. 369-374). IEEE. DOI=http://dx.doi.org/10.1109/WHC.2005.68
31. Roy Shilkrot, Jochen Huber, Wong Meng Ee, Pattie Maes, and Suranga Chandima Nanayakkara. 2015. FingerReader: A Wearable Device to Explore Printed Text on the Go. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 2363-2372. DOI: https://doi.org/10.1145/2702123.2702421
32. Stuart Taylor, Cem Keskin, Otmar Hilliges, Shahram Izadi, and John Helmes. 2014. Type-hover-swipe in 96 bytes: a motion sensing mechanical keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 1695-1704. DOI: http://dx.doi.org/10.1145/2556288.2557030
33. David Ternes, Karon E. MacLean. 2008. Designing large sets of haptic icons with rhythm. In *Haptics: perception, devices and scenarios* (pp. 199-208). Springer Berlin Heidelberg.
34. Max Wertheimer. 1923. “Laws of organization in perceptual forms.” *A source book of Gestalt Psychology*. London: Routledge & Kegan Paul.
35. Limin Zeng, Mei Miao, and Gerhard Weber. “Interactive audio-haptic map explorer on a tactile display.” In *Interacting with Computers* 27.4, 2014.
36. Haimo Zhang and Yang Li. 2014. GestKeyboard: enabling gesture-based interaction on ordinary physical keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 1675-1684. DOI=http://dx.doi.org/10.1145/2556288.2557362