Wheeler: A Three-Wheeled Input Device for Usable, Efficient, and Versatile Non-Visual Interaction

Md Touhidul Islam*
Pennsylvania State University
University Park, PA, USA
touhid@psu.edu

Noushad Sojib*
University of New Hampshire
Durham, NH, USA
noushad.sojib@unh.edu

Imran Kabir
Pennsylvania State University
University Park, PA, USA
ibk5106@psu.edu

Ashiqur Rahman Amit
Innovation Garage Limited
Dhaka, Bangladesh
amit@innovationgarage.com.bd

Mohammad Ruhul Amin
Fordham University
Bronx, NY, USA
mamin17@fordham.edu

Syed Masum Billah
Pennsylvania State University
University Park, PA, United States
sbillah@psu.edu

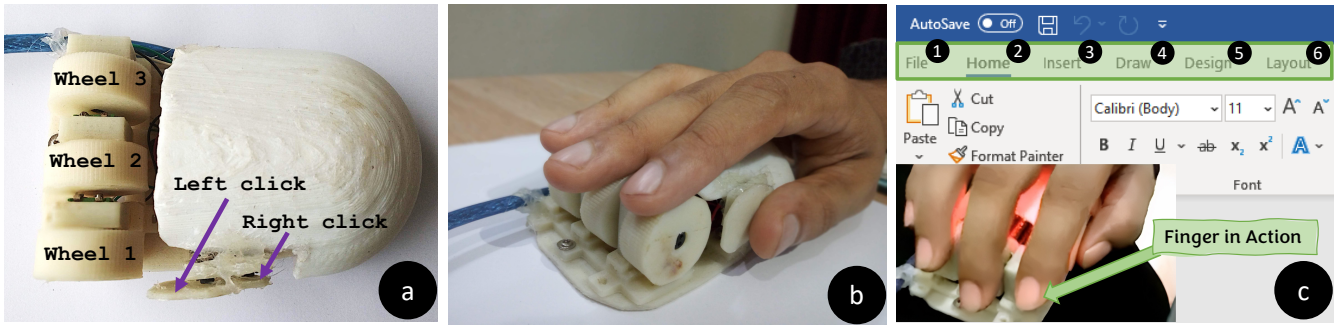


Figure 1: Wheeler input device and its usage in the wild. (a) shows a 3D-printed implementation of Wheeler having three wheels and two push buttons (primary and secondary) on the side; (b) shows a blind user holding the device, placing three central fingers on the three wheels, and the thumb over the two side buttons; (c) shows how a blind user can navigate the multi-level, hierarchical menu (e.g., Microsoft Word's ribbon menu) using Wheeler's H-nav mode. The pale green rectangle at the top shows the first level menu in the app, with the numbers 1-6 each representing individual menu items (e.g., File, Home, Insert). The user is using *Wheel 1* with their *index finger*. By rotating the wheel, the user can move the focus in the menu. The other two wheels are used for navigating the second ((e.g., Font) and third-level menu items (e.g., Boldface, Italic).

ABSTRACT

Blind users rely on keyboards and assistive technologies like screen readers to interact with user interface (UI) elements. In modern applications with complex UI hierarchies, navigating to different UI elements poses a significant accessibility challenge. Users must listen to screen reader audio descriptions and press relevant keyboard keys one at a time. This paper introduces Wheeler, a novel three-wheeled, mouse-shaped stationary input device, to address this issue. Informed by participatory sessions, Wheeler enables blind users to navigate up to three hierarchical levels in an app independently using three wheels instead of navigating just one level at a time using a keyboard. The three wheels also offer versatility, allowing users to repurpose them for other tasks, such as 2D cursor manipulation. A study with 12 blind users indicates a significant reduction (40%) in navigation time compared to using a keyboard. Further, a diary study with our blind co-author highlights Wheeler's additional benefits, such as accessing UI elements with partial metadata and facilitating mixed-ability collaboration.

CCS CONCEPTS

• **Human-centered computing** → **Accessibility technologies**; *Interaction design theory, concepts and paradigms*; **Pointing devices**.

KEYWORDS

Non-visual interaction, input device, mouse, haptics, multi-wheel, rotational input, blind, vision impairments.

ACM Reference Format:
Md Touhidul Islam*, Noushad Sojib*, Imran Kabir, Ashiqur Rahman Amit, Mohammad Ruhul Amin, and Syed Masum Billah. 2024. Wheeler: A Three-Wheeled Input Device for Usable, Efficient, and Versatile Non-Visual Interaction. In *The 37th Annual ACM Symposium on User Interface Software and*

*Equal Contribution.

UIST '24, October 13–16, 2024, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-0628-8/24/10…$15.00
https://doi.org/10.1145/3654777.3676396

Technology (UIST '24), October 13–16, 2024, Pittsburgh, PA, USA. ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3654777.3676396

1 INTRODUCTION

Blind users rely on assistive technologies (AT) like screen readers to interact with application user interfaces (UIs). On desktops, popular screen readers are NVDA [8], JAWS [7], VoiceOver [15], which offer numerous keyboard shortcuts. A keyboard serves as the primary input device for blind users on desktops. Utilizing the operating systems' built-in accessibility support, commonly known as accessible APIs [14, 48, 50], screen readers (SR) create a `DOM` tree-like textual meta-representation of an application [73]. Navigating an app using SR and keyboard shortcuts is akin to navigating this underlying meta-representation. Most screen readers allow blind users to navigate either *serially*, from left to right and top to bottom, using the `TAB` and directional `Arrow` keys, or *hierarchically*, starting from the root node and proceeding to its children and grandchildren, using a combination of shortcuts. For example, users can press `Control+Option+Shift+Down` or `Up Arrow` to move into or out of a parent node in VoiceOver. However, both navigation strategies are slow and tedious, as blind users can go to only one of the four possible neighboring elements (left, right, up, down) at a time until they reach the target element. Moreover, Recent studies have shown that apps requiring a higher average number of keystrokes for navigation are perceived as less accessible [35]. For this reason, non-visual interaction can take up to three times longer than visual interaction [52, 59].

To address the challenge of complex hierarchy navigation for blind users, we propose and design a mouse-shaped, three-wheel, stationary input device named **Wheeler**. The design of Wheeler is informed by both prior works and based on the findings from multiple participatory design sessions with four blind individuals. Figures 1a and 1b show the Wheeler prototype. Wheeler's first mode of operation is the hierarchical navigation mode, or in short, the **`H-nav`** mode. In `H-nav` mode, shown in Figure 2, the three wheels of the device are assigned to the top three levels of hierarchy in an app. The user places three fingers on the three wheels and can rotate each wheel individually using a single finger or multiple fingers simultaneously. The `H-nav` mode allows blind users to navigate complex multi-level UI hierarchies efficiently, using one wheel for each hierarchy level. There are also two buttons on the device that the user's thumb can access. The big button, which is larger, serves as the primary button, similar to the Left-click button of a mouse, and is expected to be used frequently. On the other hand, the small button serves as the secondary or Right-click button. Additionally, Wheeler has haptic feedback that notifies users when they have reached a boundary condition, such as the end of a list of menu items.

The ideation and design of Wheeler draw inspiration from previous research that proposed rotary input devices to improve the speed of website and app navigation. Two such devices, Speed-Dial [18] and NVMouse [41], were particularly influential. In Speed-Dial [18], Billah et al. used a rotary input device (e.g., a Surface Dial [49]) to interact with web pages, demonstrating that blind participants performed data tasks significantly faster using the rotary input compared to screen-reader-provided keyboard-based navigation. Lee et al. reported similar findings with their NVMouse prototype [41, 42]. While the improvements by these approaches are noteworthy, Speed-Dial or NVMouse does not solve the difficulty of navigating user interface elements that belong to different sub-trees, such as c.2 and c.3 in Figure 2c. Users are still required to navigate through the parent nodes one by one until they reach the grandparent node that contains the target node. Wheeler's three-wheel design overcomes this limitation by allowing blind users to navigate three different levels of hierarchy independently, with each wheel dedicated to a different level.

The three-wheel design of Wheeler also offers versatility, allowing users to repurpose them for other tasks, such as 2D cursor manipulation, which is a feature that is available in certain screen readers such as JAWS. JAWS allows users to explore UI elements flatly, from left to right, top to bottom. Moreover, our blind co-author in this paper mentioned using Windows Mouse Keys (MKs) to move the cursor when a keyboard alone is insufficient. However, MKs are uncomfortable with long-distance cursor movement and lack precise speed control. Additionally, they do not offer helpful cursor localization feedback, only emitting beeps. To address this issue, we create a second mode for Wheeler, the 2d navigation mode, or in short, the **`2d-nav`** mode. The illustration of `2d-nav` mode is shown in Figure 5. In this mode, the user uses `Wheel-1` and `Wheel-2` to move the cursor horizontally and vertically, respectively. `Wheel-3` is used to change the speed of the cursor movement. At any time, the user can probe the cursor's location with respect to the top-left corner of the screen using the `CTRL` button.

A user study with 12 blind participants showed that they needed 40% less time doing hierarchy navigation tasks using Wheeler's `H-nav` mode compared to using the combination of keyboard and SR. In addition, they moved the cursor around the screen to acquire targets using Wheeler's `2d-nav` mode, as instructed by a sighted confederate, enabling them to participate in mixed-ability collaborations. This study, accompanied by a diary study with our blind co-author (Section 7) also revealed that Wheeler is easy to learn and easy to use; and it offers serendipitous benefits, such as collaborating with sighted users on a shared screen and clicking on partially accessible UI elements that are otherwise unreachable via a keyboard.

We summarize our contributions as follows:

- We design and develop a multi-wheel input device named Wheeler that offers numerous benefits over typical keyboard-SR based non-visual interactions (Sections 3, 4, and 5).
- Wheeler's `H-nav` mode allows blind users to navigate hierarchies in significantly less time than keyboard-SR-based navigation (Sections 3.1 and 6.5.1).
- Wheeler's `2d-nav` mode enables mixed-ability collaboration, allowing blind users to follow the commands issued by a sighted confederate and locate targets accurately in 2D space (Sections 3.2 and 6.5.2).
- Through a diary study with our blind co-author in this paper, we identify how Wheeler provides solutions to unique problems, such as exploring graphical content without proper accessibility labels (Section 7).

2 BACKGROUND AND RELATED WORK

In this section, we first provide background on the inner workings of screen readers and the abstract UI tree. Next, we position our work in the large literature on input devices and non-visual interaction.

2.1 Construction of Abstract UI Tree for Screen Readers

User interfaces (UIs) are typically designed with the assumption that the users have no perceptual and cognitive impairments and use a typical set of input and output devices [29]. Thus, any mismatch between users' effective abilities and the underlying assumptions hampers the effectiveness of user interface design. Often, this diversity of needs is either ignored; or addressed via a manual redesign of the application UI; or via external assistive technologies (ATs), such as screen readers for blind computer users. Although a manual redesign is arguably the best [29], it is neither feasible nor scalable because users' abilities and preferences vary, which can be hard to anticipate by the designers [17].

As a result, users' ability-specific adaptations are carried out by ATs. For example, screen readers adapt application UIs by creating a *manifest* interface for users with vision impairments [60]. These screen readers (e.g., NVDA [8], JAWS [7], and VoiceOver [15]) rely on the Operating System's accessibility support, a set of well-defined functions, commonly known as *Accessibility APIs* [1, 5, 12, 14], to extract a DOM-like hierarchical meta-representation of all UI elements in an app. Each node in this tree contains a textual meta-representation (e.g., name, states) of the UI element it represents. This DOM-like abstract UI tree is invisible to sighted users but accessible via screen readers' keyboard shortcuts—when a blind user selects a node in this tree, the screen reader reads out the textual description of the corresponding UI element (e.g., `"OK Button"`) loudly. In short, the abstract UI tree and screen readers compensate for the inability of users to see a graphical user interface and implement keyboard shortcuts for users as an alternative to using a mouse to point-and-click graphical elements.

Unfortunately, these adaptations come at the expense of user experience. Prior work [31] has shown that when the output modality of a UI-rich application is manifested from its original form to another modality, i.e., consuming an application aurally instead of visually, the adaptation introduces undesired side effects, such as the application may become partially accessible and the task completion time may increase rapidly. For instance, Billah et al. [18] reported that filling out an online form on a travel-booking website could take 224*s* for blind users. On the other hand, the sighted authors of this paper performed a similar task for less than 60*s*, which indicates that the task completion time for blind users, in this case, is 3× more than that of a sighted individual.

2.2 Pointing Devices for Interaction

Pointing devices, such as computer mouses (or mice), were first conceptualized in the sixties [26, 27] and had become an effective input device to interact with graphical user interfaces (UIs) on two-dimensional (2D) screens. In early prototypes, users needed to move the mouse pointer (i.e., *cursor*) along X and Y axes on the screen by rotating a pair of wheels. These wheels were later replaced by buttons [66] in optical mice [58]. The current generation of mice still have a single wheel but for a different purpose, scrolling [32]. Pointing devices are commonly evaluated by two metrics: *performance* and *comfort* [25]. Below, we describe these two metrics.

2.2.1 Performance Measure of Pointing Devices. Fitts' law [28] is commonly used to measure the performance of pointing devices, which include several metrics, such as movement time (i.e., the time required to get to a UI object from another), error rate (i.e., % of mistakes made during a specific task), and throughput (a metric combining both movement time and accuracy). Fitts' law states that when moving a cursor from a source UI to a destination UI, the index of difficulty (**ID**, in bits), or in short, the difficulty, is proportionate to the distance between the source and the destination and inversely proportional to the width (or area) of the destination. Its initial formulation was for 1D, which Mackenzie et al. [46] extended to 2D by replacing the width with the target's height if the height is smaller or calculating the width along the direction of approach to the target. For UI elements having the same height and width (e.g., icons), considering only the width of the target is sufficient. Mackenzie et al. [47] also propose two extended static measures, orthogonal direction change, and movement direction change; two dynamic measures, movement variability, and movement error, to make the comparisons more comprehensive for individuals with no vision impairments. In our evaluation, we found that static measures are relevant to evaluate Wheeler's performance, especially in `2d-nav` mode with staircase-like 2D movements.

2.2.2 Comfort of Pointing Devices. The comfort of pointing devices is subjective and measured by asking questions on rating different aspects, such as users' physical effort, fatigue and comfort (e.g., hand/wrist posture comfort, clicking comfort), speed and accuracy, and overall usability [22, 24, 25]. However, having too many questions can be confusing to users. Therefore, we evaluated Wheeler on three aspects: clicking comfort, satisfaction, and overall usability.

2.3 Adaptation of Pointing Devices for Non-Visual Interaction

Pointing devices like mice are hardly used by blind users because the pointer (or the mouse cursor) only provides *visual* feedback [18], which blind users cannot perceive. Therefore, researchers attempted to replace this visual feedback with an alternative, such as 2D translational force, vibration, skin stretch, and thermal feedback to convey kinesthetic and tactile information regarding the object under the cursor [40, 75]. Often, this alternative feedback is presented on a small tactile display on one side of the mouse. However, the spatial resolution of this display is coarse due to the small display size. Kim et al. [37] designed an inflatable mouse that uses an air balloon, which senses the pressure from users' hands and thus acts as an input device. Nevertheless, this device is tiring for tasks requiring high finger pressure and precision.

A large body of literature on haptic feedback is devoted to Braille displays, auditory feedback, and tactile pin arrays [10, 38, 39, 56, 57, 62, 69–71, 74, 76]. Some of the work proposed guidelines for designing appropriate haptic sensations for blind users [38]. Soviak et al. [69–71] explored an alternative, glove-like input device to

convey UI elements' boundaries via audio-haptic feedback. These prototypes demonstrated the benefits of haptic feedback for non-visual interaction in obtaining an overview of a web page; however, the unrestrained freedom of movement within the page hinders precise navigation and information finding. Unlike the above prototypes, Wheeler uses haptic feedback to inform user actions and announce screen or UI boundaries only.

2.4 Rotary Input for Interaction

Another line of work represents UI elements on the two-dimensional (2D) screen into a one-dimensional (1D) circular list, then uses rotary devices (e.g., dials, wheels) to rapidly navigate through the list [18, 41, 42]. Because rotary input (e.g., rotate clockwise/anti-clockwise) is inherently one-dimensional, this mode of interaction is more fitting and efficient for blind users when navigating a list. In fact, sighted users also benefit from rotary input with a mouse wheel for scrolling [32]. Lee et al. [43] extended the rotary input for sighted users from a single mouse wheel to three wheels by placing two virtual wheels on both sides of the physical wheel. Their experiment revealed that users performed certain operations (e.g., volume up or volume down in a media player) up to two times faster than a single wheel. In addition, the users get better with more iterations of the same tasks. These encouraging findings inspired us to design Wheeler with multi-wheels.

2.5 Accommodations for Mouse and Virtual Cursors

Operating Systems (OSes) often support accommodations for mouse-based interaction. For example, Windows OS allows users to move the mouse cursor by pressing Num keys [4], commonly known as Mouse Keys; MacOS supports controlling the native VoiceOver [13] screen reader through trackpad inputs [3]. However, during our participatory design sessions, we found that blind participants rarely use these accommodations due to usability issues.

Screen readers usually provide multiple cursors. For example, JAWS [7] screen reader offers three cursors: *PC cursor*, which is the caret in Word/text documents; *JAWS cursor*, which is the mouse pointer; and *virtual PC cursor*, which simulates an invisible caret in webpages. NVDA [54] also supports similar cursors (e.g., NVDA's *review cursor* is similar to JAWS's *virtual PC cursor*). In addition, screen readers can be configured to report the textual content directly beneath the mouse cursor as the user moves the cursor [53]. Multiple cursors are useful for different applications and tasks, as indicated by our participants. Wheeler design aligns with this notion of multiple cursors.

3 OVERVIEW OF WHEELER

Wheeler is a mouse-shaped input device with three wheels and two push buttons on the side, as shown in Figure 1. All three wheels are identical in size. However, the primary button is slightly bigger than the secondary one. The bigger button's role is similar to that of the left click or the primary button of a typical mouse. Likewise, the smaller button acts as the right-click button of a mouse. A user can grip the device with their right hand so that their index finger rests on the first wheel, the middle finger on the second wheel, the ring finger on the third wheel, and the thumb over the two buttons, as shown in Figure 1. When assembled, Wheeler measures 103 mm in length, 60 mm in width, and 33mm in height, meeting the ANSI standard range [33] for mouse dimensions: 120 mm in length, 40-70 mm in width, and 25-40 mm in height.

Unlike a mouse, Wheeler is stationary, i.e., users do not move it on the surface when using it. Instead, they rotate wheels to maneuver the cursor. In our current design, Wheeler is connected to a computer via a USB cable. However, a Bluetooth-based wireless connection is feasible.

Wheeler provides various audio-haptic feedback to communicate the current context of the cursor. Wheeler has a buzzer and haptic motor on its motherboard. The buzzer emits a beep during significant events, while the haptic motor creates a gentle vibration with each rotation—none interfere with the audio output from screen readers.

Wheeler primarily operates in two modes: (i) a hierarchical navigation (`H-nav`) mode and (ii) a two-dimensional or flat navigation (`2d-nav`) mode. Based on the feedback from a diary study with our blind co-author (Section 7), we later introduced another mode named `2d-T-nav`, which is a special case of `2d-nav` mode facilitating the teleportation of the mouse cursor.

Table 1 provides an overview of various input methods and their corresponding results when using Wheeler, considering the device's current operating mode. It is worth noting that we have strategically incorporated the keyboard `CTRL` button as an input modifier in three different scenarios. This approach simplifies user interaction by requiring them to remember just one key on the keyboard.

3.1 Interaction Using Wheeler: Hierarchical Navigation (`H-nav` Mode)

In `H-nav` mode, Wheeler navigates the abstract UI tree of an app, as shown in Figure 2. By default, three wheels of Wheeler point to the top three levels in an app's DOM. Each wheel maintains its own cursor—making a total of three independent cursors. Further, each wheel maintains its own state. For example, a wheel remembers the last UI object a user had focused on the last time in a level and resumes interacting from that element when the user focuses back. Thus, it eliminates the need to explore elements from the beginning in a hierarchy.

The rotate action (e.g., clockwise or counterclockwise) selects an element bidirectionally at a level. While `wheel 1` can select any element in the 1^{st} level, `wheel 2` only selects the immediate children of the element currently selected by `wheel 1`. Recursively, `wheel 3` only selects the immediate children of the element selected by `wheel 2`. When `Wheel-1`'s cursor moves to a certain node in the UI tree, `Wheel-2`'s cursor automatically moves to the first child of the node selected by `Wheel-1`. Similarly, `Wheel-3`'c cursor automatically moves to the first child of the node selected by `Wheel-2`. Figure 2c demonstrates how the menus and ribbons of two applications can be organized in a tree hierarchy and mapped in Wheeler.

To perform a left-/right-click operation, the user presses the primary/secondary side buttons. Users can define the rotation resolution (in degrees) to adjust a wheel's sensitivity. Upon rotation, Wheeler provides audio-haptic feedback to affirm a valid operation

No.	Mode	Input/Action	Outcome
1	H-nav	Wheel-1 scroll	Navigate hierarchy level currently mapped to Wheel-1
2		Wheel-2 scroll	Navigate hierarchy level currently mapped to Wheel-2
3		Wheel-3 scroll	Navigate hierarchy level currently mapped to Wheel-3
4		CTRL + Primary button press	Move all three wheel's assignments one level down in the app hierarchy
5		CTRL + Secondary button press	Move all three wheel's assignments one level up in the app hierarchy
6	2d-nav	Wheel-1 scroll	Move cursor horizontally
7		Wheel-2 scroll	Move cursor vertically
8		Wheel-3 scroll	Adjust cursor speed
9		Secondary button press and hold for 300ms	Turn on or off 2d-T-nav mode
10		CTRL press	Announce the cursor's location on the screen
11	H-nav or 2d-nav	Primary button press	Simulate a mouse left click
12		Secondary button press	Simulate a mouse right click
13		CTRL + Both Button Press	Switch between H-nav and 2d-nav modes

Table 1: Summary of Wheeler's inputs and outcomes in different modes.

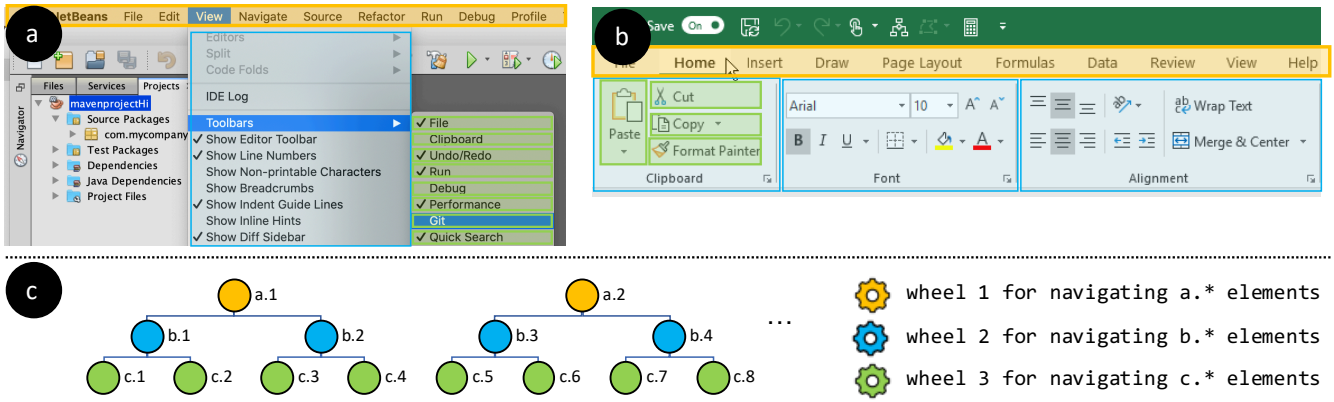


Figure 2: Demonstration of Wheeler's H-nav mode. (a) Multi-level menus in NetBeans; (b) nested structures in ribbons in MS Word; (c) a sample 3-level tree hierarchy to represent menus in either (a) or (b). For example, 1^{st}-level elements, {a.1, a.2, ...} could represent the top-level menu items like {File, Edit, View, ... Profile} in (a) or {Home, Insert, ..., Help} in (b). Assuming a.1 equals Home, the 2^{nd}-level menu items, {b.1, b.2, ...} will be {Clipboard, Font, ...} in (b). Similarly, assuming b.1 equals Clipboard, the 3^{rd}-level menu items, {c.1, c.2, ...} will be {Paste, Cut, ...} in (b). In H-nav mode, Wheel-1 is always mapped to 1^{st}-level menu items (i.e., a.*), Wheel-2 is mapped to 2^{nd}-level menu items (i.e., b.*), and Wheel-3 is mapped to 3^{rd}-level menu items (i.e., c.*).

and sometimes to convey spatial information, such as whether a UI element is the last (or first) among its siblings.

3.1.1 H-nav Mode vs. Using Keyboard and Screen Reader. To demonstrate the advantage of H-nav mode over using a keyboard and a screen reader combo, suppose a user wants to move from node c.2 (source) to node c.8 (destination) in Figure 2c.

Fig. 3 shows the navigation steps when the user is using a keyboard and a screen reader. Note that it would require at least six operations (i.e., TAB or [SHIFT+TAB] key press) in total.

Fig. 4a- 4d shows the navigation steps when the user is using Wheeler for the same navigation task. Notice how the user can complete the task in three rotations.

3.1.2 Traversing Apps with More than 3 Levels. If an application has more than 3 levels, the user can move all three cursors one level down in the app hierarchy by pressing Wheeler's Primary button while holding the CTRL key. Likewise, to move all three cursors upward, they can do so by holding the CTRL key and pressing Wheeler's Secondary button. These actions are also shown in Table 1.

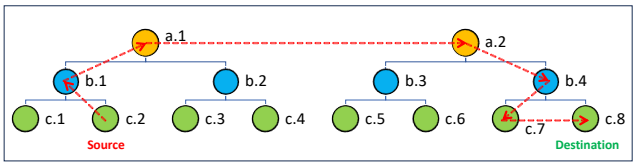


Figure 3: The red arrows highlight the path (`c.2` -> `b.1` -> `a.1` -> `a.2` -> `b.4` -> `c.7` -> `c.8`) a blind user would have to take using a combination of keyboard and screen reader when going from `c.2` to `c.8` in the hierarchy from Fig. 2c. There are six steps in total, and at least six keystrokes would be required.

3.2 Interaction Using Wheeler: Two-Dimensional Navigation (`2d-nav` and `2d-T-nav` Modes)

In `2d-nav` mode, the wheels have different roles: `Wheel-1` moves the cursor along the `X-axis`, `Wheel-2` moves it along the `Y-axis`, and `Wheel-3` is used to control the speed of the cursor movement. `2d-nav` mode is demonstrated in Figure 5, which depicts a scenario of a blind user moving the cursor on a 2D screen from the lower-left corner to the upper-right corner. While moving the cursor, the user can rotate `Wheel-3` to change the speed of the cursor movement. Users can scroll both `Wheel-1` and `Wheel-2` simultaneously and it would result in diagonal cursor movement.

Loss of context is a prominent issue for users with visual impairment while navigating in 2d space [34]. To address this, users can probe the cursor location anytime by pressing the `CTRL` key in `2d-nav` mode. When `CTRL` is pressed, Wheeler reads out the cursor location as a percentage of `X` and `Y` coordinates with respect to the screen width and height. For instance, if the user's cursor is above item `B` in Figure 5, Wheeler would read out something like *"30% from the left and 10% from the top"*. On cursor-hover, Wheeler's built-in TTS engine automatically reads out the name of a UI element.

3.2.1 `2d-T-nav` *Mode.* `2d-T-nav` is a special case of `2d-nav` mode, where Wheeler teleports the mouse cursor to the closest neighboring UI along the direction of the cursor movement. This is faster than `2d-nav` to move from one element to another.

3.3 Toggling Modes

To toggle between `H-nav` and `2d-nav` modes, users have to hold the `CTRL` button and press both the primary and secondary buttons of Wheeler at the same time. If Wheeler is in `2d-nav` mode, the users can turn on or off the `2d-T-nav` mode by pressing and holding the secondary (i.e., the small) button for some time (e.g., 300 ms).

4 IDEATION OF WHEELER

The initial design of Wheeler is informed by the literature. Below, we synthesize relevant prior work and describe how it inspired our design.

4.1 Three Rotary inputs

We observed that certain desktop screen readers (SRs), such as VoiceOver and ChromeVox, allow users to navigate application UIs hierarchically, similar to traversing an HTML DOM tree. These SRs let users go to a UI's immediate parent, child, or siblings one at a time. Often, UI elements next to each other visually belong to different parents, i.e., sub-trees, in the DOM. It creates a challenge for blind users because they must traverse different sub-trees to navigate those elements. This observation led us to design an input device that lets blind users traverse different sub-trees independently.

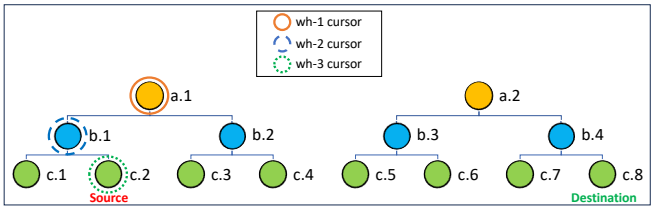
We were also inspired by Speed-Dial [18], where Billah et al. demonstrated that a rotary input device could emulate mouse-like functions for blind users. However, Speed-Dial does not address the challenge of navigating UI elements in different sub-trees—users still need to go to the parent nodes individually until they find the grandparent whose child is the target node. Therefore, we conceptualized a hypothetical device with three rotary inputs, where each rotor is mapped to a level to reduce the number of times users need to go up in the parents. Moreover, developers often organize their apps hierarchically using a standard template—at the high level, there are menus, toolbars, sidebars, status bars, and client areas, each of which can have a second level, e.g., sub-menus, split toolbars, and containers/groups; and the most interactive elements (e.g., buttons, text areas) appear at the third level. This template inspired us to use three wheels, one for each level, to maximize coverage.

However, we found that the UI hierarchy of most applications spans more than three levels. We considered adding another wheel (under the pinky), but in our design mock-up, we found rotating this wheel difficult. This is due to the connection between the intrinsic muscles of human hands and innervation—the pinky and medial half of the ring finger are connected with the ulnar nerve (for gross hand movement). In contrast, the index, middle, and lateral half of the ring finger are connected by the median nerve (for more precise hand movement). As such, we kept the number of wheels to three, assigning the most frequently used wheel under the index and the least frequently used wheel under the ring.

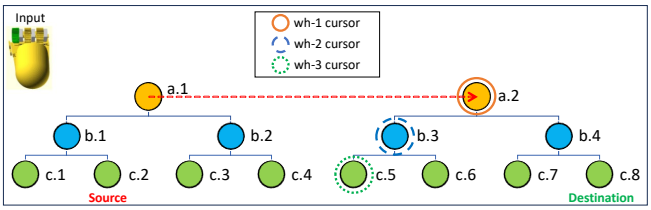
For a similar reason, we placed the wheels vertically so that a finger only requires flexion/extension movement, for which maximum biomechanical advantage is available (e.g., more muscle groups are involved). Placing the wheel horizontally will require a finger abduction/adduction movement, for which fewer muscle groups are involved.

4.2 Flat 2D Navigation Mode

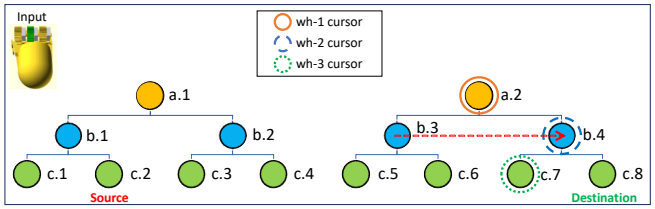
We also observed that certain screen readers (e.g., JAWS) allow users to explore UI elements flatly, from left to right, top to bottom. Further, the blind co-author of this paper stated that he occasionally uses Windows Mouse Keys (MKs) to manipulate the cursor when an element is not reachable by the keyboard. However, MKs are hard to use— pressing these arrow keys for long-distance cursor movement is uncomfortable. In addition, MKs lack a fine-grained cursor pace control. Moreover, MKs do not provide useful feedback to localize the cursor; it only beeps. These pieces of information motivated us to devise a separate mode in our device that can act as a usable mouse for blind users.



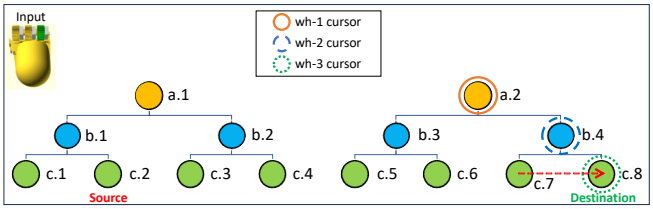
(a) The cursors for the three wheels at their initial position.



(b) User rotates `Wheel-1` to move its cursor from `a.1` to `a.2`. The cursors for `Wheel-2` and `Wheel-3` automatically move to `b.3` and `c.5`, respectively.



(c) User rotates `Wheel-2` to move its cursor from `b.3` to `b.4`. The cursor for `Wheel-3` automatically moves to `c.7`.



(d) User rotates `Wheel-3` to move its cursor from `c.7` to `c.8`

Figure 4: Navigating from `c.2` to `c.8` in the hierarchy from Fig. 2c using Wheeler's `H-nav` mode. (a)-(d) shows the positions of the three Wheeler cursors mapped to its three wheels at different stages of the navigation. Only three rotations are required in total.

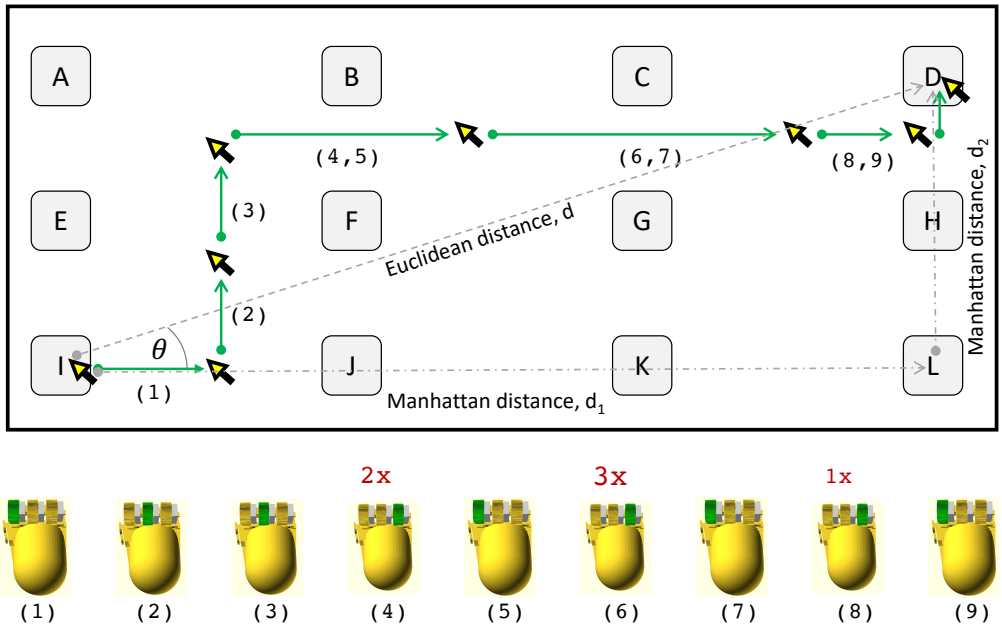


Figure 5: Illustrations of Wheeler's `2d-nav` mode. (Top) A blind user moves the cursor from the lower-left corner to the upper-right corner of a 2D screen with 12 UIs organized in a 3×4 grid. (Bottom) shows the sequence of wheel operations the user performs at different steps to achieve this goal. Each step `(i)` is marked in both the top and the bottom figures. The green-colored wheel indicates which wheel the user rotated at step `(i)`. The user rotates `Wheel-1` or `Wheel-2` to move the cursor horizontally or vertically and adjusts the speed of the cursor by rotating `Wheel-3`. Note that the Euclidean distance between the source and the destination is d, which sighted users usually take using a mouse. In contrast, blind users take the Manhattan distance between two nodes (e.g., d_1 along X, plus d_2 along Y) in `2d-nav` mode.

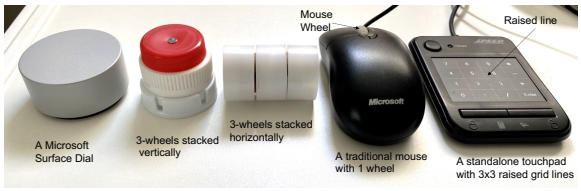


Figure 6: Several low-fidelity prototypes during ideation.

4.3 Initial Design

After conceptualizing the design, we organized a participatory design session with four blind individuals. The session aimed to brainstorm the design of a device capable of embodying our navigational paradigm. We presented several low-fidelity prototypes (shown in Figure 6) with a different number of rotary inputs as well as existing input devices that sighted users use. These include a Surface Dial, a 3-wheel vertical device, a 3-wheel horizontal device, a mouse, and a touchpad that supports multi-touch.

We explained our design idea to them and asked their opinion about a usable device. Participants preferred the form factor of the Surface Dial and mentioned that a 3-wheel vertical device could be an extension to Surface Dial but reported difficulty in rotating multiple wheels all at the same time. All participants preferred the 3-wheel horizontal device because they could comfortably place 3 fingers over 3 wheels. However, they mentioned that rotating the wheels without supporting their palm was difficult. When one participant pointed out the idea of combining a mouse with the 3-wheel horizontal device, all participants were elated; they remarked that it could be a viable prototype. When asked about the touchpad supporting multi-finger inputs, everyone was firmly against it. They contended that employing three fingers simultaneously would lead to a highly counter-intuitive experience, as moving one finger in one direction and the others in another direction would be confusing. Essentially, they believed they had to swipe all three fingers either upward or downward, which did not align with our intended concept of three *independent* cursors.

Once we established the initial design, we asked the participants where to place the two mouse buttons. Three participants proposed making both buttons easily accessible by the thumb, suggesting placing them on the left side. Other participants initially recommended placing one button on each side, but they soon recognized that pressing the right button with the pinky finger would be challenging. In conclusion, the most promising design that emerged was a 3-wheel horizontal device resembling a mouse, featuring two buttons that were easily accessible by the thumb.

4.4 Design Iterations

The version of Wheeler we presented in this paper resulted from three major design iterations. As one of the authors of this paper is a blind power user, we had the privilege to discuss, update, and evaluate mini-iterations of Wheeler in-house. In each major iteration, we invited the same four blind individuals to collect their feedback and recommendations and to ensure that we incorporated their earlier feedback into the current iteration of the design. Appendix A contains more details on these design iterations.

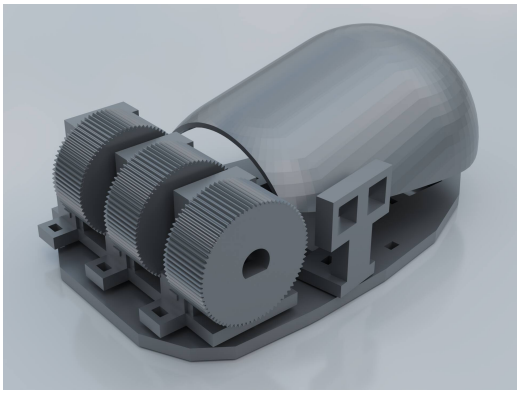


Figure 7: A high-level rendering of Wheeler putting individual 3D parts together.

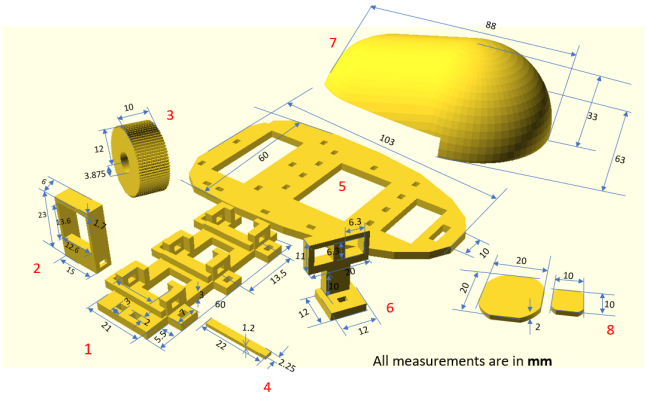


Figure 8: Individual 3D components of Wheeler with dimensions in millimeters: (i) wheelbase, (ii) encoder holder, (iii) wheel, (iv) connector, (v) main base, (vi) button stand, (vii) top shell, and (viii) button covers.

5 WHEELER IMPLEMENTATION

This section discusses Wheeler's hardware implementation, electrical components, and firmware design.

5.1 Hardware Components

We designed the 3D components in modules using OpenSCAD [2]. The modular parts were printed using a Prusa i3 3D printer with white PLA filament. Figure 7 shows a high-level rendering of the device putting all individual 3D parts together, and Figure 8 shows a rendering of its eight distinct 3D parts: wheelbase, encoder holder, wheel, connector, main base, button stand, top shell, and button covers. Important dimensions are given in millimeters (mm). For example, the radius of each wheel is 12 mm, and the width is 10 mm. We plan to release our design on public repositories for wider adoption.

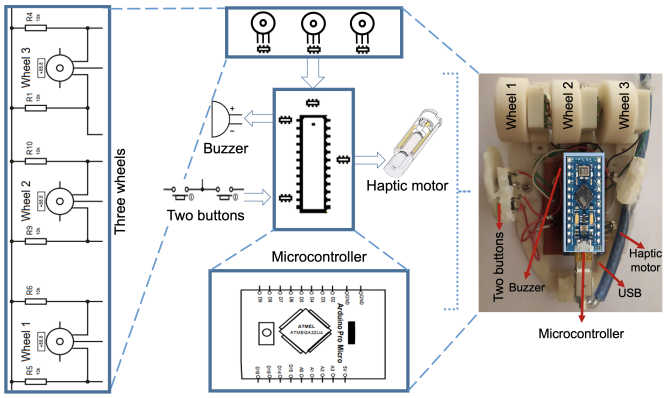


Figure 9: A high-level block diagram of the major electrical components of Wheeler. The microcontroller measures the rotation of each of the three wheels and captures each button press to generate an appropriate system-level event.

5.2 Electrical Components

We used an Arduino Pro Micro as the main controller and three rotary encoders to detect rotation in three wheels. To provide audio-haptic feedback, we used a buzzer and a pager motor. The buzzer alerts the user with beep feedback, and the pager motor provides haptic feedback when instructed. These electronic components are assembled within the 3D printed parts. Figure 9 shows a schematic diagram of these components put together—all of which cost less than $30 USD in total. We will publicly release the model, part number of all electrical components, and the 3D design.

5.3 Wheeler Firmware

Our firmware was developed on the Arduino platform. Since Arduino Pro Micro works as a Human Interface Device (HID), Wheeler does not require an additional device driver to recognize it as a mouse-like device on a computer when connected through a serial port. Each rotary encoder has *twelve* slots—each of which provides a clicking effect (mild haptic feedback) in the finger when rotating the wheel. In `2d-nav` mode, the firmware detects the rotation direction for each wheel, calculates each rotation step, encodes this information into a mouse event, and sends it to the computer, which interprets it as a regular mouse event.

However, in `H-nav` mode, Wheeler's firmware is integrated with NVDA, an open-source screen reader. The firmware appears as an NVDA add-on and has access to the UI hierarchy of any app from the NVDA APIs, which internally consume Windows' native UI Automation API [50] to extract the UI tree and relay the rotational input on the tree.

In `2d-T-nav` mode, the firmware issues a mouse event. It consumes an accessibility API call (`AccessibleObjectFromPoint`) to check the closest neighboring UI along the direction of the cursor movement.

6 EVALUATION OF WHEELER

We recruited 12 blind participants and conducted two lab studies to evaluate Wheeler. To ensure fair evaluation, we did not invite the four participants who took part in our early design iterations (Sec. 4.4).

First, we evaluated the effectiveness of Wheeler's `H-nav` mode in *Study 1*, described in Section 6.3. This study was completed in a single session. **Second**, we evaluated Wheeler's `2d-nav` mode in *Study 2*, described in Section 6.4. This study was completed in six sessions and occurred on six different days over a month. All studies were IRB-approved. Table 3 presents an overview of two studies, including tasks, conditions, related hypotheses, and the number of sessions or duration. Next, we describe participants' demographics and common study procedure, followed by Study 1 and Study 2, and finally, the findings (Sec. 6.5).

6.1 Participants

12 blind participants (8 male, 4 female) were recruited with an average age of 31.33 ($SD = 5.48$, $Range = 22 - 39$) through an institution that provides services to people with vision impairments. All of them were familiar with Windows desktop screen readers (e.g., JAWS and NVDA). None had any motor impairments. Some participants had light perception. Participants came from a diverse background; most of them were students. Table 2 presents participant demographics and individuals' self-reported expertise with screen readers.

6.2 Study Procedure

This section describes the procedure that was common in both studies. In addition, study-specific procedures are described in the respective study sections.

The lab studies were conducted in an office environment by two authors. After verbal consent, the conductors asked participants to introduce themselves, their history of blindness, their expertise in screen readers (self-disclosed), and their use of point devices and screen reader cursors. The experiment was set up on a Windows 10 laptop with 1366×768 screen resolution. This laptop had the following software installed: two screen readers (JAWS and NVDA), a video conferencing software (Zoom), a remote desktop software (TeamViewer), and the device driver for Wheeler, which was connected via a USB port.

Of the two authors who administered the study, one was blind and interacted with the participants in person, following social-distancing guidelines. The other author was sighted and assisted in setting up the study environment and trials over TeamViewer and supervised each session remotely over Zoom video conferencing software. The participants were given sufficient instructions and time ($10 - 30$ minutes) to familiarize themselves with Wheeler. Each session lasted an hour and was video-recorded and later transcribed for further analysis. Each participant was compensated with an hourly rate of USD $10.

Upon completing each study, the experimenters engaged in an open-ended discussion, seeking subjective feedback, recommendations, and ratings on different aspects of Wheeler, such as the placement of wheels, ease of use, the dynamic pace control feature, and perceived challenges in learning this new interaction paradigm and potential of using this device in everyday technology use.

ID	Age/ Sex	Expertise	Light Perception?	History of Blindness	Profession	Screen Readers
P1	35/M	Expert	No	Advantageous	Student	JAWS, NVDA
P2	25/F	Beginner	No	Congenital	College Student	JAWS, NVDA
P3	32/F	Beginner	Yes	Advantageous	Undergrad student	JAWS, NVDA
P4	38/M	Beginner	No	Advantageous	Undergrad student	JAWS, NVDA
P5	31/F	Beginner	No	Advantageous	Graduate student	JAWS
P6	31/M	Beginner	No	Advantageous	Education	JAWS
P7	28/F	Beginner	No	Congential	Special Ed. Trainee	JAWS
P8	22/F	Beginner	No	Congential	NGO Worker	JAWS, NVDA
P9	32/M	Beginner	Yes	Congential	Sales	JAWS, NVDA
P10	30/M	Beginner	Yes	Congential	Communication support	NVDA, Magnifier
P11	26/M	Expert	No	Congential	IT program officer	JAWS, NVDA
P12	39/M	Expert	No	Advantageous	Small business owner	NVDA, JAWS

Table 2: Participants' demographics, history of blindness, and self-reported expertise with screen readers.

Study	Task	Conditions	Associated Hypotheses	Design	Sessions/ Duration
Study 1: Evaluation of `H-nav` mode	T1	C0, C1	H1	Within-subject	1 Session
Study 2: Evaluation of `2d-nav` mode	T2	C2	H2, H3	Repeated measures	6 Sessions

Table 3: An overview of two studies, including tasks, conditions, related hypotheses, and duration.

6.3 Study 1: Evaluation of `H-nav` Mode

In this study, we aim to validate the following hypothesis:

- **H1**: Participants will navigate hierarchical structures more efficiently with Wheeler's `H-nav` mode than with a keyboard and a screen reader.

6.3.1 Study Design. We chose Microsoft Office Suite's ribbon-based multi-level menu as the representative hierarchical structure. As reported in prior studies [41, 42], navigating ribbon items is particularly challenging for blind users. As such, any improvement in navigating ribbons is important to the blind community.

The participants performed the following navigation task (T1) using two study conditions (described below).

Task ***T1:*** In a multi-level hierarchical menu, navigate to a sub-menu item, given its path in the hierarchy. For example, in Figure 2, a representative task could be *"go to* `Home` *tab, then* `Alignment` *group, then* `Wrap Text` *item"*. Here, `Wrap Text` is the target and its path starting from the top, `Home> Alignment> Wrap Text`, was given to the participants.

The two conditions are described as follows:

Condition ***C0****: Keyboard with Screen Reader.* The participants must use a screen reader and basic navigational keys, including `ARROW` keys (e.g., ↑, ↓, ←, →) and other modifier keys (e.g., `TAB`, `Alt`, and `ESC`). This was our baseline.

Condition ***C1****: Wheeler in* `H-nav` *mode with TTS.* The participants must use Wheeler's `H-nav` mode with a Text-to-Speech (TTS) synthesizer. They were not allowed to use a screen reader or its keyboard shortcuts.

We recorded the task completion times. Five trials were performed in each condition.

6.3.2 Procedure Specific to Study 1. For task T1, we chose ribbon tabs from two commonly used apps in MS Office Suite: MS Word and MS Excel. In addition, we selected the ribbon items that blind users use less frequently. For instance, our blind author informed us that other blind users are less likely to use `References`, `Review`, and `Mailings` ribbon tabs in MS Word and `Draw`, `Formula`, `Data` tabs in MS Excel. We, therefore, included the target from those tabs. A sample of task T1 was as follows: (i) go to `References> Footnotes> Insert Endnote`; (ii) go to `Review> Comments> Show Comments`; and (iii) go to `Formulas> Formula Auditing> Show Formulas`.

The participants practiced `H-nav` mode on a different hierarchy (e.g., the tree-view of `Windows Explorer`) to familiarize themselves with the three wheels. In each trial, the experimenter randomly drew a target (without replacement) from a predefined list of 30 targets. Then, the experimenter read out the target and its path and asked the participant to go to that target using a study condition. The experimenter could repeat this information during a trial if asked.

By default, the `Home` ribbon tab was expanded. Participants were instructed (but not enforced) to start from a ribbon pane they were currently in, as they completed the previous trial. A trial was completed when a participant focused on the target and declared it verbally. We counterbalanced the order of conditions across participants. The experimenter took notes during the session. The experimenters allocated 3 minutes for each trial and recorded 120

data-points (=12 participants × 5 trials × 2 study conditions) for task T1.

6.4 Study 2: Evaluation of `2d-nav` Mode

Evaluating `2d-nav` mode was less straightforward than `H-nav` because `2d-nav` is not directly comparable to a typical mouse. For example, sighted users can acquire the target visually with a mouse cursor, whereas blind users must be given the target's location (e.g., x, y coordinates) on the screen to acquire with `2d-nav` mode. Similarly, `2d-nav` is not comparable to a keyboard either because keyboard-based navigation acquires the target based on its relative position in the abstract UI tree, not its spatial location on the screen.

Therefore, we created a real-world scenario in which a blind participant and a sighted confederate collaborated remotely on a shared screen. The sighted confederate asked the blind peer to move their mouse cursor over a target UI element on the screen. The confederate additionally provided a rough estimation of the target's screen location. For example, in a remote desktop session shown in Figure 17.c, the confederate asked the blind peer to move the cursor over `Google Chrome` icon, which is roughly 5% from the left edge and 60% from the top edge.

Based on the above scenario, we defined the following two hypotheses:

- **H2**: Given the spatial coordinate of the target, participants can independently acquire the target with the mouse cursor using `2d-nav`.
- **H3**: `2d-nav` is easy to learn and use.

6.4.1 Study Design. `2d-nav` mode is a novel interaction technique to manipulate the mouse cursor. As such, we conducted the study in multiple sessions to measure how well participants's performance increased in each session. More specifically, the study was a repeated-measure design, where each participant performed the following task (T2) using `2d-nav` mode in six sessions on six different days over a month.

Task ***T2****: Target acquisition.* Move the mouse cursor to a target point, given its coordinates, as shown in Figure 5 and Figure 17.c. For example, on Windows Desktop, take the mouse cursor over `This PC` icon, which is x% from the left and y% from the top of the screen.

To the best of our knowledge, current screen readers or mouse keys do not allow one to perform the above task. Therefore, we had only one condition (C2) for `2d-nav` mode and no baseline. Participants performed T2 in 6 different sessions; thus the session being the independent variable. Each session had 6 trials.

Moreover, using percentage coordinates in `2d-nav` mode, instead of pixels or inches, makes the location of a UI element agnostic to different screen sizes or resolutions. It also helps sighted collaborators estimate and provide the UI location on the screen to their blind counterparts.

Condition ***C2****: Wheeler in* `2d-nav` *mode with TTS.* The participants must use Wheeler's `2d-nav` mode.

Recall that we measured how well participants's performances improved over sessions (i.e., learning rate) and how well they adopted `2d-nav` mode (i.e., user behavior). Towards that, we recorded the following measures: (i) task completion time for all trials; (ii) number of times cursor-location is probed; (iii) number of times cursor speed is changed; and (iv) the mean cursor speed.

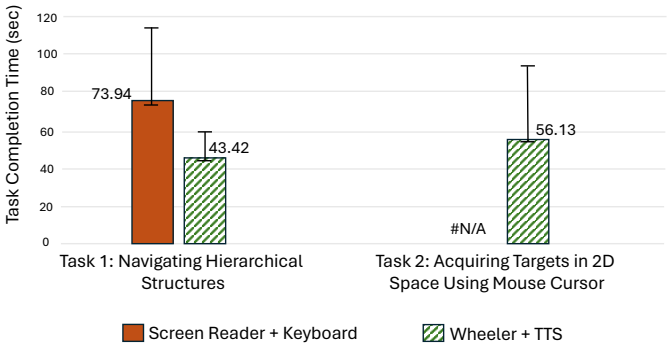


Figure 10: Mean task completion times (the lower, the better) for T1 in two study conditions and T2 in one condition. For T2, no baseline was available, and the reported completion time was observed in the final session. Per session-specific measures are available in Figure 11. Error bars show +1 SD.

6.4.2 Procedure specific to Study 2. In each session, the participants were given sufficient instructions and time (10 minutes) to recap `2d-nav` mode. They practiced on a webpage containing 12 buttons, organized in 3x4 grids, as shown in Figure 5.

For each trial in T2, the sighted experimenter (i.e., the confederate) reoriented 35 icons on the study laptop's desktop screen and randomly chose the target icon. In addition, the experimenter placed the mouse cursor in the top-left corner of the desktop. Each icon was a square with dimensions $36px \times 36px$. The experimenter read out the name of the target icon and the coordinate of its center in percentages from the left and top edges of the screen and asked the participants to bring the mouse cursor over this target using Wheeler's `2d-nav` mode.

A trial was completed when a participant brought the cursor over the target icon, and the TTS read out its name. For each trial, the experimenters allocated 3 minutes. If a participant failed to complete a trial within the stipulated time limit, it was recorded as incomplete. It was removed from the evaluation because of a lack of valid completion time. In sum, out of 432 data-points (=12 participants × 6 trials × 1 condition × 6 sessions), 336 were valid for T2.

The failure cases were primarily due to timeouts, i.e., when a trial exceeded 3 minutes. It happened when the target location was non-trivial to estimate or its size was small. For example, estimating 77% from the left is more challenging than estimating 50% or 80% from the left. In these scenarios, participants often overshot or undershot the target, became more cautious, and decreased the cursor movement speed (`Wheel-3`)—all contributing to increased trial time.

6.5 Findings from Study-1 and Study-2

We analyzed the recorded video data, observations, transcriptions, and experimenters' notes to report our findings. Findings from Study 1 appear first, followed by the Study 2.

6.5.1 Completion Time for Navigating Hierarchical Structure (Study 1). The participants took 40% less time with Wheeler's H-nav mode ($Mean = 43.41s, SD = 16.31s$), compared to the baseline ($Mean = 73.94s, SD = 42.04s$), which is statistically significant, as reported by a paired-t test ($t = 2.303, p < .042$). Figure 10 shows the mean completion time for task T1. Recall that the three wheels in H-nav mode are mapped to three different hierarchies in the UI tree, and each wheel maintains its state independently of the others. For this reason, participants could jump from one sub-tree to another (e.g., between cousin nodes) by simply rotating the first or second wheel. In contrast, to make a similar jump with the baseline, they needed to go to the parent node first, then the parent's siblings (uncles), and finally their (uncles') children, which was cumbersome and time-consuming. Thus, the reduction in completion time in navigating multi-level hierarchies with H-nav was expected and unsurprising. This validates our hypothesis H1.

6.5.2 Completion Time in Acquiring Known Targets with the Mouse Cursor (Study 2). The task completion time in 2d-nav mode generally follows a decreasing trend over sessions. For example, in session 1, the average completion time was 120*s* (shown in Figure 11), which went down to 56.12*s* in session 6 (shown in Figure 11 and Figure 10). These decrements are statistically significant, as indicated by a one-way within-subject ANOVA test, $F(5, 60) = 622.4, p \approx 0$. Although we noticed minor increments in sessions 4 and 5, these could be attributed to a longer interval (e.g., 14 days) between session 3 and session 4, whereas other intervals were 5 to 6 days. Overall, the decreasing trend indicates that the task completion time can decrease even further as someone uses 2d-nav consistently.

During a task, the participants did not ask for assistance on the whereabouts of their cursor. They controlled the cursor pace by using Wheel-3 (see Figures 13 and 14) and probed the cursor location from time to time by pressing CTRL (see Figure 12). All validate our hypothesis H2.

We noticed that the average completion time of 56.12*s* was still longer than the time sighted users spend acquiring a visual target using a mouse. This issue also emerged from the diary study findings with our blind co-author (Section 7). Per our blind co-author's recommendation, we created 2d-T-nav, a special case of 2d-nav mode, to teleport the mouse cursor to the closest neighboring UI along the direction of the cursor movement.

Participants probed the cursor location from time to time (using the CTRL key) to update the corresponding cursor location in their mental map. This increased the mental workload for some participants at the beginning. However, as they progressed through the study, they increasingly became more comfortable and confident with this mapping, which is evident in Figures 11—14.

6.5.3 Learnability in 2d-nav mode. A key insight gleaned from the data is that in 2d-nav mode, the average task completion time correlated well ($R^2 = 0.83$) with the power law of practice [68], as shown in Figure 11), and learning occurred in the last session, suggesting that the participants are more likely to take less time as they practice more. Another key insight is that the participants oriented themselves with the two-tone audio feedback conveying the cursor's current coordinates, as indicated by the decreasing liner trendline ($R^2 = .883$) in Figure 12. In addition, they became more comfortable using 2d-nav after multiple sessions and learned how to operate it more efficiently. For instance, most participants figured out a preferable speed (e.g., 7.0 pixel/rotation) after the first 3 sessions, as shown in Figures 13 and Figures 14. These findings suggest that Wheeler is easy to learn and easy to use. Thus, our hypothesis H3 is validated.

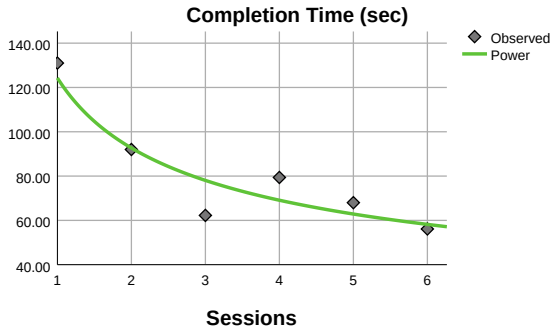


Figure 11: Average task completion time per session for T2 in 2d-nav mode, fitted to a decreasing power trendline, $y = 124.28 * x^{-0.42}$, $R^2 = .83$.

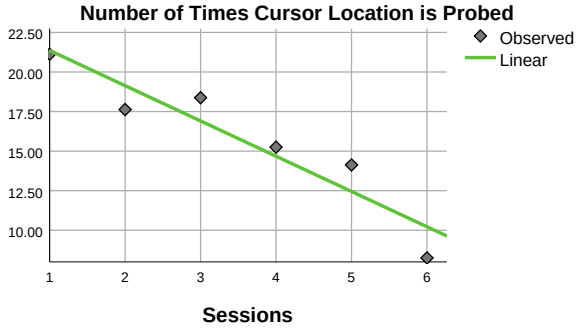


Figure 12: Average number of times cursor location is probed during T2 with 2d-nav, fitted to a linear, decreasing trendline ($R^2 = .883$).

6.5.4 Movement Trajectories in 2d-nav Mode (Study 2). We recreated the paths or trajectories that participants followed to move their cursor in 2d-nav mode from a source UI to a destination UI (the target). Figure 15 shows four such paths ('a' to 'd'), where a red circle (marked with the number 1) is the source, the green circle (marked with the highest number in a path) is the destination, and the blue circles indicate a cursor probe (by pressing CTRL) at that location. These trajectories reveal the following insights:

First, participants could get confused about moving left or right (or up or down) at the beginning (shown in paths *a* and *c*), but they quickly figured it out. For example, in path *a*, the target was on the right side of the source, but a participant initially moved left, realizing the two-tone feedback sounded wrong, then probed the cursor to be certain, and changed the course to the right side (i.e., the correct direction).

Second, most participants moved along the X-axis or Y-axis in a long, continuous stride (e.g., path *a*) instead of moving in a staircase

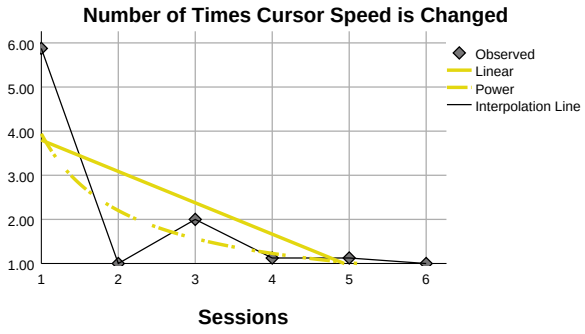


Figure 13: Average number of times cursor speed is changed during T2 with `2d-nav`. The plot shows that most participants had settled on a preferable speed after the first 3 sessions.

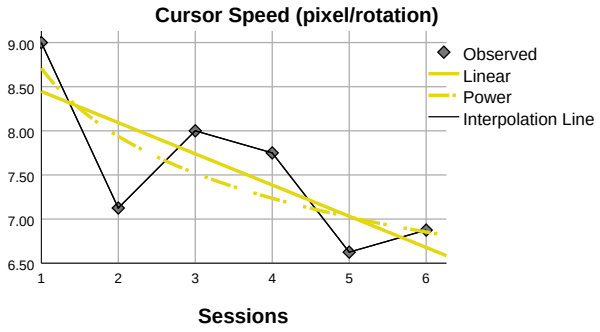


Figure 14: Average cursor speed per session during T2 with `2d-nav`. This plot shows that the cursor speed decreased over time, and participants comfortably settled it on 7.0 pixel/rotation speed.

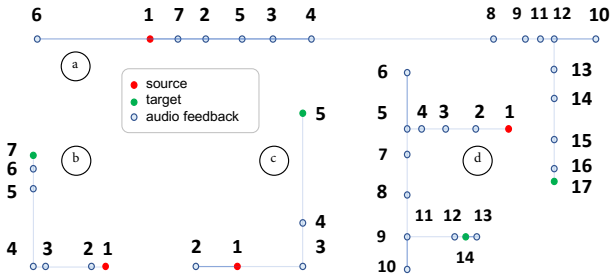


Figure 15: Four dominant paths or trajectories that participants followed to move their cursor in `2d-nav` mode from a source UI to a destination UI (the target). A red circle (marked with the number 1) is the source, the green circle (marked with the highest number in a path) is the destination, and the blue circles indicate a cursor probe (by pressing `CTRL`) at that location.

pattern (along both axes). This was surprising because we anticipated staircase patterns to be dominant. This also indicates that the participants had developed some notion of spatial awareness of their cursor based on various audio-haptic feedback provided by Wheeler.

Third, as participants approached the target or a turn, they probed the cursor more to be ascertained. Although this behavior was unsurprising, we were surprised by the caution participants took not to overshoot the target.

These are encouraging insights that indicate the potential of `2d-nav` for blind users.

It is evident from the above discussion that locating targets in 2D space using Wheeler's `2d-nav` mode can yield trajectories different from what a sighted user would take when locating targets using a mouse. Although the movement time/target locating time can be higher to start with, we show in Appendix B.4 how blind users can achieve sighted user-like performance by changing the cursor pace using *wheel-3* in `2d-nav` mode. The idea presented in Appendix B.4 is mostly theoretical but can offer intriguing insights into possible improvements blind users can achieve through the long-term use of Wheeler.

Category	1pt	2pts	3pts	4pts	5pts
Usability	0	0	2	8	2
Clicking Comfort	0	1	5	5	1
Overall Comfort	0	0	3	6	3

Figure 16: Participants ratings (1 to 5) on the use of comfort of Wheeler in three categories: usability, clicking comfort of the buttons, and overall comfort or satisfaction. One (1) is very negative, and five (5) is very positive. The shade of a cell indicates the frequency of responses, which is also shown numerically within a cell. Note that 4 (i.e., *positive*) is the most frequent response.

6.6 Observations and Subjective Feedback

6.6.1 Feedback on Comfort. Participants rated the use of comfort on a Likert scale of 1 to 5 (1: very negative, 5: very positive) under three categories: usability, clicking comfort of the buttons, and the overall comfort or satisfaction of using the device. Most participants rated 4, i.e., *positive*, on all three categories, as shown in Figure 16. The mean scores for usability, clicking comfort, and overall comfort were 4.0, 3.5, and 4.0. We noted that some participants had difficulty clicking the secondary (small) button. When we asked, they mentioned that the two buttons were placed too closely on the same side. Their suggestions and feedback on improving the design are presented in Section 6.6 below.

Eight participants reported that they found the device very useful. Notably, they liked the idea of moving the cursor with three wheels. Four of them mentioned that they faced difficulties while using it for the first time. However, they remained confident that their experience of using the device would improve over time.

6.6.2 Feedback on Wheeler Design. Six participants suggested making the wheels and buttons smoother to grip the device firmly and

perform operations more comfortably. Four participants mentioned that they found it challenging to use the two buttons under the thumb. They suggested increasing the space between the two buttons. On the contrary, two participants suggested that placing the secondary button on the other side of the device would be more convenient. One participant wished to switch the modes (between `H-nav` and `2d-nav`) from the context menu of the secondary button. Another participant (P9), who had light perception, suggested that a bright-colored cursor could benefit people with low vision. Two participants (P6, P8) suggested placing one wheel that moves the cursor along the `X-axis` horizontally in order to make it more relatable. However, we argue that placing a wheel horizontally will make it difficult to rotate, thus raising a usability concern.

6.6.3 `H-nav` *vs.* `2d-nav` *mode.* All of our participants mentioned that both modes are useful, complementary, and have distinct use cases. For example, 3 participants who frequently use different software in MS Office Suite (e.g., Word, PowerPoint, and Excel) for employment were enthusiastic about `H-nav` mode. They mentioned that they would never use a keyboard to navigate ribbon menus in MS Office Suite. Two other participants mentioned that changing mouse pointer speed, probing the cursor, and two-tone audio feedback helped them picture the spatial layout of the desktop. They were surprised to discover that the 'Window Start' menu was located in the bottom-left corner for the first time. P12 provided several use cases where he must need `2d-nav` mode: editing an image and working with graphical objects in Unity. A very different use case was echoed in P11's comment:

> *"... you know many applications and websites have 'blindspots' where my keyboard cannot reach, and my screen reader does not talk. I think I can access those blind spots with Wheeler [`2d-nav` mode]".*

7 LONG TERM USABILITY OF WHEELER: A DIARY STUDY

Background. After conducting two studies, the blind author (BA) of this paper decided to use Wheeler on his laptop for everyday technology access. BA is a 40-year-old Asian male who runs an institution that provides IT services to blind and low-vision users. He is blind by birth and has no light perception. Also, he is a "power" user and comfortable with multiple screen readers (e.g., JAWS, NVDA). Although BA does not have a background in computer programming, he understands how screen readers work.

BA journaled his experience of using Wheeler and met other (sighted) authors over Zoom (a teleconferencing software) periodically, e.g., every two weeks for six months. In each meeting, BA updates his usage pattern and issues (if any) with the device. Below, we briefly summarize the findings from these meetings.

Usage of Wheeler. BA mentioned that he uses `H-nav` mode extensively, every day, for his job. For example, he accesses the hierarchical structures (e.g., ribbon, multi-layer menus) in Word documents, Excel spreadsheets, and Outlook email clients with `H-nav`. Therefore, `H-nav` works as envisioned.

BA also mentioned how he used `2d-nav` recently to overcome accessibility issues in several scenarios, some of which are shown in Figure 17:

Scenario 1. When the COVID-19 vaccine was available in his country, he needed to make a reservation by registering online. On the registration website, there was a mandatory drop-down (shown in Figure 17.b.1) to select his identity type (e.g., passport, birth certificate), which was not reachable to keyboards. Still, he managed to access it using `2d-nav` mode. He described that moment as follows: *"No one else was there to help. Everyone was registering for themselves. But luckily, Wheeler worked!"* When we investigated the website, we found that the drop-down had an incorrect ARIA label, `aria-hidden = "true"`, making it hidden from the screen reader (shown in Figure 17.b.2).

Scenario 2. He could click on the playback speed, seek, and pause audio in the VLC Media Player app. These buttons have no text labels and are confusing to navigate with a screen reader (shown in Figure 17.a).

Scenario 3. He mentioned that inserting and resizing videos in PowerPoint slides is difficult with keyboard shortcuts, but he found a workaround with `2d-nav`.

Scenario 4. He shared an incident on how he troubleshot a client's computer remotely via Zoom screen sharing. His client was unable to access an image button on their computer. BA asked the client to grant him remote control permission and then used `2d-nav` mode to click on that button for him. In his comment, *"It took some time to figure out where that button was on the screen. But I found it. It felt so good!"*

Feature Request. BA mentioned that he often explores the spatial layout of an app or website using `2d-nav`. However, he found it "too slow" to move from one element to another, especially when two elements have a large gap. So, he suggested an option to jump from one element to another if they are in close proximity. Based on this request, we implemented `2d-T-nav` mode, which teleports the cursor, as described in Section 3.2. After implementing this feature, BA mentioned that he enjoys using Wheeler in `2d-T-nav` mode to explore the spatial layout of different apps and teaches his clients and fellow blind users about the basic layout of popular apps/websites.

In sum, our findings from the diary study indicate that Wheeler can substantially improve the non-visual interaction experience for blind users.

8 DISCUSSION

Our findings show that Wheeler can improve the hierarchical menu navigation in applications with its `H-nav` mode and allows blind users to locate location-known targets in 2D space with its `2d-nav` mode. Here, we discuss the broader implications, limitations, and future directions of Wheeler.

8.1 Need for a Pointing Device in Non-Visual Interaction

For sighted users, interacting with a graphical interface using a mouse is independent of whether individual UI elements contain underlying textual metadata. However, this is not the case for

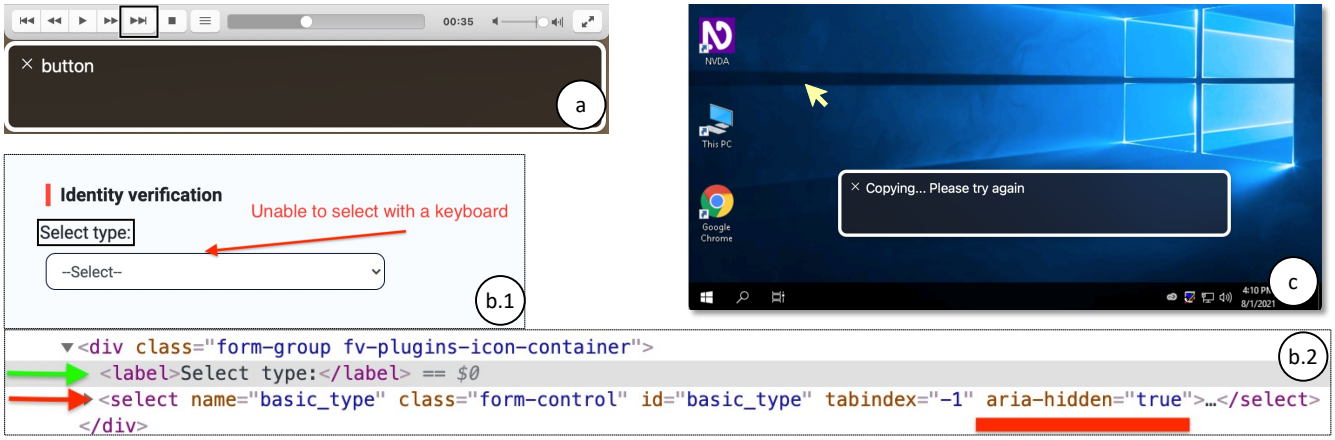


Figure 17: Examples of partially accessible application and website. (a) On the VLC media player in OSX, two image buttons, `Play Next` and `Play Previous`, lack proper labels. As such, the screen reader only reads "button". (b.1) On a COVID-19 vaccine registration page, `Select Type` drop-down list is inaccessible (i.e., hidden) to screen readers because of its incorrect ARIA label, `aria-hidden = "true"`, as highlighted by a thick red line in its DOM tree shown in (b.2). (c) A screenshot of a virtual desktop, which is not accessible to OSX's native screen reader, `VoiceOver`.

blind users because screen readers rely on the metadata to generate audio feedback on the focused element. Therefore, if an element does not have sufficient textual metadata (e.g., empty `label`, `alt-text` attributes [20, 30, 61]) or has incorrect attribute value (e.g., `aria-hidden = "true"` to an otherwise visible web element), the element becomes inaccessible or unreachable to screen readers. Unfortunately, these inaccessible UI elements pose a major challenge for blind users [51], for which they seek sighted assistance (e.g., asking sighted family members or calling remote sighted agents [44] in services like AIRA [11], Be My Eyes [6]) or search for a screen reader plug-in [51]. As indicated in the diary study (Section 7), the blind author (BA) of this paper often encountered inaccessible UI elements; he successfully interacted with those elements using Wheeler with limited or no sighted assistance. This highlights the need for a pointing device in non-visual interaction and Wheeler's potential to fulfill that need.

8.2 An Augmentation to Keyboard and Screen Reader-based Interaction

While Wheeler provides advantages over traditional non-visual interaction methods like keyboard and screen readers, its purpose is to augment the existing methods, not to replace them. A notable advantage of screen reader-based interaction would appear when the user knows most shortcuts for navigating an application's items. Blind users often rely on application shortcuts, but studies indicate limited shortcut knowledge even among experienced blind users [35]. When shortcuts are unknown, unavailable, or hard to remember, Wheeler still facilitates faster hierarchy navigation. Thus, Wheeler complements existing methods without aiming to replace them, offering a proficient alternative for efficient non-visual interaction.

8.3 Increasing Productivity of Blind Users

For blind users, the inefficiency in operating commonly used office software is a major hindrance to employment [21]. To attain basic proficiency with productivity software (e.g., Microsoft Office Suite), blind users typically go through social services and federal- or state-funded specialized training programs [16]. Their training process can be summarized as memorizing numerous keyboard shortcuts and practicing to build muscle memory [19]. Wheeler, as indicated in our findings, can lessen this burden by making access to multi-layer menus fast and structured.

8.4 Enabling Technology

Our data suggest that Wheeler can enable mixed-ability, blind-sighted remote collaboration. For example, a blind user can acquire the target on a shared screen like a sighted user if the target's location is roughly estimated. This is important for blind users because the increased acceptance of remote work and improved connectivity software may open new employment opportunities for them [64]. However, the lack of accessibility to remote collaboration tools is a known issue [19, 20, 23, 72], which can hinder those opportunities [45]. Wheeler can offer an alternative to circumvent these accessibility issues with remote collaboration tools.

Our findings also suggest that Wheeler can enable working with graphical data, such as editing an image, interacting with online data visualization tools (e.g., Plotly [9]), and 3D modeling with CAD software. Although employment opportunities in these areas are increasing, these are largely inaccessible to blind users [63, 65, 67]. Nevertheless, Wheeler-like devices make them workable for blind users.

8.5 Potential Application in Virtual Reality

Wheeler can be repurposed to be used as an input device in virtual reality (VR). The current input mechanisms in VR are mostly

limited to handheld controllers and gestures—although gesture-based control is still in its infancy for platforms such as Oculus. Handheld controllers, on the other hand, cannot provide any 3D orientation feedback but provide basic haptics support for events such as boundary hits. Additionally, there is limited keyboard access and no notion of screen readers in any VR platform, making VR an exclusive domain to sighted users. With wheeler's `2d-nav` mode, accompanied by its feedback mechanism, navigation in the 3D world could be possible for blind users— especially in menu navigation, which is mostly 2D yet lacks any keyboard or screen reader support, something blind users get on desktop platforms such as Windows.

8.6 Potential Application in Data Visualization

Our study findings indicate that `H-nav` mode relies on blind users' ability to conceptualize and navigate hierarchical structures without visual cues. This mental mapping enables them to systematically anticipate and traverse different information levels, similar to how sighted users might visually scan and interpret hierarchical data representations. We believe any data visualization that can be represented as a graph or tree-like form can be traversed conveniently using Wheeler's three wheels. For instance, if the visualization is encoded as a graph, one way to create a hierarchy is to consider the current node as the root, and nodes one hop away from the root are at level 1, two hops away are at level 2, and so on. Thus, Wheeler's 3-wheel architecture provides a versatile framework for navigating various hierarchical data visualizations such as dendrograms, social networks, and organizational charts.

In dendrograms used for hierarchical clustering, `Wheel-1` would allow users to select the root cluster, while `Wheel-2` would facilitate exploration of immediate clusters and `Wheel-3` wouls help dive deeper into sub-clusters. For social networks, Wheeler would enable users to start from a central node (e.g., an influencer or organization) using `Wheel-1`. `Wheel-2` then would allow exploration of immediate connections (e.g., friends or followers), while `Wheel-3` would extend exploration to secondary connections, providing insights into community structures and information flow across the network. In organizational charts, users can begin with the CEO or top-level executive using `Wheel-1`. `Wheel-2` then would enable exploration of direct reports and major departments, while `Wheel-3` would allow deeper dives into teams and divisions within each department. This structured navigation aids in understanding reporting relationships, departmental structures, and organizational hierarchies.

8.7 Limitations and Future Work

The findings from our diary study (Sec. 7) indicate Wheeler's room for improvement, particularly regarding hardware design, such as the placement of the two side buttons. In future work, we plan to experiment with different placements and sizes of the primary and secondary buttons. Another limitation of Wheeler's `2d-nav` mode is the assumption that users have prior knowledge of their desired target location within the UI, either from experience or from a sighted confederate. One potential solution is integrating an AI assistant capable of receiving spoken instructions from the user and providing tentative screen coordinates for the desired target.

In the future, we plan to write a separate device driver for Wheeler to broadcast rotational events system-wide so that applications can consume these events directly, similar to standard mouse/keyboard events, thus augmenting the input space for non-visual interaction. This can also eliminate writing the application-specific adaptation for rotational inputs. Moreover, we plan to integrate Wheeler with an open-source screen reader, such as NVDA [55], to make the transition between keyboard-based interaction to Wheeler-based interaction seamless. Finally, we will make Wheeler prototype open source by releasing the 3D design, schematic diagram, and part numbers of various electrical components for wide adoption.

9 CONCLUSION

This paper presents a three-wheel mouse-shaped stationary input device, Wheeler, to make non-visual interaction efficient and versatile. This device adopts a rotational input paradigm that prior work has found helpful for blind users. Informed by prior work, the design of Wheeler is refined by participatory design sessions and the experience of a blind co-author in this paper. The prototype is made of 3D-printed components and commercially available electrical components. Wheeler is evaluated by 12 blind participants in two user studies. The study findings suggest that Wheeler can take up to 40% less time navigating dense, hierarchical UI structures. Moreover, blind participants can maneuver the mouse cursor to acquire a target on the screen given its location. Further, Wheeler is easy to use and learn, and users' performance can improve over time. A diary study with our blind co-author indicates that the device works as envisioned by large. It can increase the productivity of blind users in using office software and offer several serendipitous benefits, including remote collaboration, interacting with partially inaccessible applications and websites, and promoting independence.

REFERENCES

[1] [n.d.]. Microsoft Active Accessibility and UI Automation Compared. http://msdn.microsoft.com/en-us/library/windows/desktop/dd561918(v=vs.85).aspx
[2] [n.d.]. OpenSCAD. https://openscad.org/
[3] [n.d.]. Touchpad with voiceover. https://support.apple.com/en-us/HT205770
[4] [n.d.]. Use mouse keys to move pointer. https://support.microsoft.com/en-us/windows/use-mouse-keys-to-move-the-mouse-pointer-9e0c72c8-b882-7918-8e7b-391fd62adf33
[5] [n.d.]. Windows Automation API. http://msdn.microsoft.com/en-us/library/windows/desktop/ff486375(v=vs.85).aspx
[6] 2015. Be My Eyes: Bringing sight to blind and low vision people. https://www.bemyeyes.com/
[7] 2018. What's New in JAWS 2018 Screen Reading Software. Retrieved September 19, 2018 from https://www.freedomscientific.com/downloads/JAWS/JAWSWhatsNew
[8] 2020. NV Access. https://www.nvaccess.org/. (Accessed on 09/20/2018).
[9] 2021. Plotly. https://plotly.com/
[10] AFB. [n.d.]. Refreshable Braille Displays. http://www.afb.org/ProdBrowseCatResults.aspx?CatID=43
[11] Aira. 2018. Aira. https://aira.io/. Retrieved May 23, 2020 from https://aira.io
[12] Android. [n.d.]. Accessibility. https://developer.android.com/guide/topics/ui/accessibility/index.html
[13] Apple. [n.d.]. Vision Accessibility in iPhone. https://www.apple.com/accessibility/iphone/vision/
[14] Apple. 2011. *NSAccessibility*. https://developer.apple.com/documentation/appkit/accessibility_for_macos/nsaccessibility [Accessed: 2020-06-29].
[15] Apple Inc. 2020. VoiceOver. https://www.apple.com/accessibility/osx/voiceover/.
[16] Edward C. Bell and Natalia M Mino. 2021. Employment Outcomes for Blind and Visually Impaired Adults. https://nfb.org/images/nfb/publications/jbir/jbir15/jbir050202.html
[17] Eric Bergman and Earl Johnson. 1995. Towards accessible human-computer interaction. *Advances in human-computer interaction* 5, 1 (1995), 87–114.

[18] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and I.V. Ramakrishnan. 2017. Speed-Dial: A Surrogate Mouse for Non-Visual Web Browsing. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 3132531, 110–119. https://doi.org/10.1145/3132525.3132531

[19] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and I.V. Ramakrishnan. 2017. Ubiquitous Accessibility for People with Visual Impairments: Are We There Yet?. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 5862–5868. https://doi.org/10.1145/3025453.3025731

[20] Syed Masum Billah, Donald E. Porter, and I. V. Ramakrishnan. 2016. Sinter: low-bandwidth remote access for the visually-impaired. In *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 2901335, 1–16. https://doi.org/10.1145/2901318.2901335

[21] Ben Clements, Graeme Douglas, and Sue Pavey. 2011. Which factors affect the chances of paid employment for individuals with visual impairment in Britain? *Work* 39, 1 (2011), 21–30.

[22] Denis A Coelho, Miguel L Lourenço, and Isabel L Nunes. 2017. Psychometric analysis of scales for usability evaluation of pointing devices. In *International Conference on Applied Human Factors and Ergonomics*. Springer, 419–426.

[23] Maitraye Das, Darren Gergle, and Anne Marie Piper. 2019. " It doesn't win you friends" Understanding Accessibility in Collaborative Writing for People with Vision Impairments. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–26.

[24] Naser Dehghan, Alireza Choobineh, Mohsen Razeghi, Jafar Hasanzadeh, Moslem Irandoost, and Samaneh Ebrahimi. 2015. Assessment of functional parameters and comfort of a new computer mouse as compared with other types of input devices. *International Journal of Occupational Safety and Ergonomics* 21, 4 (2015), 493–497.

[25] Sarah A Douglas, Arthur E Kirkpatrick, and I Scott MacKenzie. 1999. Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 215–222.

[26] Douglas C Engelbart. 1962. Augmenting human intellect: A conceptual framework. *Menlo Park, CA* (1962).

[27] William K English, Douglas C Engelbart, and Melvyn L Berman. 1967. Display-selection techniques for text manipulation. *IEEE Transactions on Human Factors in Electronics* 1 (1967), 5–15.

[28] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381.

[29] Krzysztof Gajos and Daniel S. Weld. 2004. SUPPLE: Automatically Generating User Interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces* (Funchal, Madeira, Portugal) *(IUI '04)*. Association for Computing Machinery, New York, NY, USA, 93–100. https://doi.org/10.1145/964442.964461

[30] Darren Guinness, Edward Cutrell, and Meredith Ringel Morris. 2018. Caption Crawler: Enabling Reusable Alternative Text Descriptions Using Reverse Image Search. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 518, 11 pages. https://doi.org/10.1145/3173574.3174092

[31] Kip Harris. 2006. Challenges and solutions for screen reader/I.T. interoperability. *SIGACCESS Access. Comput.* 85 (2006), 10–20. https://doi.org/10.1145/1166118.1166120

[32] Ken Hinckley and Daniel Wigdor. 2002. Input technologies and techniques. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (2002), 151–168.

[33] American National Standards Institute. 2007. ANSI/HFES 100–2007 Human Factors Engineering of Computer Workstations. *Hum Fact Ergon Soc* 10 (2007), 89.

[34] Md Touhidul Islam and Syed Masum Billah. 2023. SpaceX Mag: An Automatic, Scalable, and Rapid Space Compactor for Optimizing Smartphone App Interfaces for Low-Vision Users. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 2 (2023), 1–36.

[35] Md Touhidul Islam, E Porter Donald, and Syed Masum Billah. 2023. A Probabilistic Model and Metrics for Estimating Perceived Accessibility of Desktop Applications in Keystroke-Based Non-Visual Interactions. In *The 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM. https://doi.org/10.1145/3544548.3581400

[36] Richard J Jagacinski and John M Flach. 2018. *Control theory for humans: Quantitative approaches to modeling performance*. CRC press.

[37] Seoktae Kim, Hyunjung Kim, Boram Lee, Tek-Jin Nam, and Woohun Lee. Year. Inflatable mouse: volume-adjustable mouse with air-pressure-sensitive input and haptic feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 211–224.

[38] 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*. ACM, 1525–1534.

[39] Ravi Kuber, Shaojian Zhu, Yevgeniy Arber, Kirk Norman, and Charlotte Magnusson. 2014. Augmenting the Non-Visual Web Browsing Process Using the Geomagic Touch Haptic Device. *ACM SIGACCESS Accessibility and Computing* 109 (2014), 4–10.

[40] Ki-Uk Kyung, Dong-Soo Kwon, and Gi-Hun Yang. 2006. A novel interactive mouse system for holistic haptic display in a human-computer interface. *International Journal of Human-Computer Interaction* 20, 3 (2006), 247–270.

[41] Hae-Na Lee, Vikas Ashok, and I. V. Ramakrishnan. 2020. Repurposing Visual Input Modalities for Blind Users: A Case Study of Word Processors. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2714–2721. https://doi.org/10.1109/SMC42975.2020.9283015

[42] Hae-Na Lee, Vikas Ashok, and I. V. Ramakrishnan. 2020. Rotate-and-Press: A Non-visual Alternative to Point-and-Click?. In *HCI International 2020 – Late Breaking Papers: Universal Access and Inclusive Design*, Constantine Stephanidis, Margherita Antona, Qin Gao, and Jia Zhou (Eds.). Springer, Springer International Publishing, Cham, 291–305.

[43] Kuo-Wei Lee and Ying-Chu Lee. 2010. Design and validation of virtually multiple mouse wheels. *International Journal of Industrial Ergonomics* 40, 4 (2010), 392–401.

[44] Sooyeon Lee, Madison Reddie, Chun-Hua Tsai, Jordan Beck, Mary Beth Rosson, and John M Carroll. 2020. The emerging professional practice of remote sighted assistance for people with visual impairments. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[45] Kelly Mack, Maitraye Das, Dhruv Jain, Danielle Bragg, John Tang, Andrew Begel, Erin Beneteau, Josh Urban Davis, Abraham Glasser, Joon Sung Park, et al. 2021. Mixed Abilities and Varied Experiences: a group autoethnography of a virtual summer internship. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility* (Virtual Event, USA) *(ASSETS '21)*. Association for Computing Machinery, New York, NY, USA, Article 21, 13 pages. https://doi.org/10.1145/3441852.3471199

[46] I Scott MacKenzie and William Buxton. 1992. Extending Fitts' law to two-dimensional tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 219–226.

[47] I Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. 2001. Accuracy measures for evaluating computer pointing devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 9–16.

[48] Microsoft. [n.d.]. Microsoft Active Accessibility: Architecture. https://msdn.microsoft.com/en-us/library/windows/desktop/dd373592(v=vs.85).aspx [Online; accessed 15-March-2015].

[49] Microsoft. 2017. Surface Dial. https://www.microsoft.com/en-us/surface/accessories/surface-dial

[50] Microsoft Inc. 2020. UI Automation Overview. http://msdn.microsoft.com/en-us/library/ms747327.aspx

[51] Farhani Momotaz, Md Touhidul Islam, Md Ehtesham-Ul-Haque, and Syed Masum Billah. 2021. Understanding Screen Readers' Plugins. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 1–10. https://doi.org/10.1145/3441852.3471205

[52] Emma Murphy, Ravi Kuber, Graham McAllister, Philip Strain, and Wai Yu. 2008. An empirical investigation into the difficulties experienced by visually impaired Internet users. *Universal Access in the Information Society* 7, 1-2 (2008), 79–91. https://doi.org/10.1007/s10209-007-0098-4

[53] NV Access. 2020. NVDA Mouse Navigation. https://www.nvaccess.org/files/nvda/documentation/userGuide.html#NavigatingWithTheMouse

[54] NVAccess. 2020. NV Access: Home of the free NVDA Screen Reader. http://www.nvaccess.org/.

[55] NVDA-Project. 2020. GitHub - nvaccess/nvda: NVDA, the free and open source Screen Reader for Microsoft Windows. https://github.com/nvaccess/nvda. Accessed: 2020-06-29.

[56] M Sile O'Modhrain and Brent Gillespie. Year. The moose: A haptic user interface for blind persons. In *Proc. Third WWW6 Conference*.

[57] T. Pietrzak, A. Crossan, S. A. Brewster, B. Martin, and I. Pecci. 2009. Creating Usable Pin Array Tactons for Nonvisual Information. *IEEE Transactions on Haptics* 2, 2 (2009), 61–72. https://doi.org/10.1109/TOH.2009.6

[58] P Popov, S Pulov, and V Pulov. 2004. A laser speckle pattern technique for designing an optical computer mouse. *Optics and Lasers in Engineering* 42, 1 (2004), 21–26.

[59] Yury Puzis, Yevgen Borodin, Rami Puzis, and I.V. Ramakrishnan. 2013. Predictive web automation assistant for people with vision impairments. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2488478, 1031–1040. https://doi.org/10.1145/2488388.2488478

[60] Robert Reimann, Alan Cooper, David Cronin, and Chris Noessel. 2014. *About Face: The Essentials of Interaction Design, 4th Edition*.

[61] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2018. Examining Image-Based Button Labeling for Accessibility in Android Apps through Large-Scale Analysis. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (Galway, Ireland) *(ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 119–130. https://doi.org/10.1145/3234695.3236364

[62] Martin Rotard, Sven Knödler, and Thomas Ertl. 2005. A tactile web browser for the visually disabled. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. ACM, 1083361, 15–22. https://doi.org/10.1145/1083356.1083361

[63] Anastasia Schaadhardt, Alexis Hiniker, and Jacob O Wobbrock. 2021. Understanding Blind Screen-Reader Users' Experiences of Digital Artboards. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–19.
[64] Lisa A Schur, Mason Ameri, and Douglas Kruse. 2020. Telework after COVID: a "silver lining" for workers with disabilities? *Journal of occupational rehabilitation* 30, 4 (2020), 521–536.
[65] Ather Sharif, Sanjana Shivani Chintalapati, Jacob O Wobbrock, and Katharina Reinecke. 2021. Understanding Screen-Reader Users' Experiences with Online Data Visualizations. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 1–16.
[66] Sol Sherr. 2012. *Input devices*. Vol. 1. Elsevier.
[67] Lei Shi, Yuhang Zhao, and Shiri Azenkot. 2017. Markit and Talkit: a low-barrier toolkit to augment 3D printed models with audio annotations. In *Proceedings of the 30th annual acm symposium on user interface software and technology*. 493–506.
[68] G. S. Snoddy. [n.d.]. Learning and stability: a psychophysiological analysis of a case of motor learning with clinical applications.
[69] Andrii Soviak. 2015. Haptic Gloves Prototype for Audio-Tactile Web Browsing. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. 363–364.
[70] Andrii Soviak, Vikas Ashok, Yevgen Borodin, Yury Puzis, and IV Ramakrishnan. 2015. Feel the Web: Towards the Design of Haptic Screen Interfaces for Accessible Web Browsing. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. 391–392.
[71] Andrii Soviak, Anatoliy Borodin, Vikas Ashok, Yevgen Borodin, Yury Puzis, and IV Ramakrishnan. 2016. Tactile Accessibility: Does Anyone Need a Haptic Glove?. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 101–109.
[72] John Tang. 2021. Understanding the Telework Experience of People with Disabilities. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 30 (April 2021), 27 pages. https://doi.org/10.1145/3449104
[73] The Web Hypertext Application Technology Working Group (WHATWG). 2022. DOM Structure. https://dom.spec.whatwg.org/
[74] Eu Jin Wong, Kian Meng Yap, Jason Alexander, and Abhijit Karnik. 2015. HABOS: Towards a platform of haptic-audio based online shopping for the visually impaired. In *Open Systems (ICOS), 2015 IEEE Confernece on*. IEEE, 62–67.
[75] Gi-Hun Yang, Ki-Uk Kyung, Young-Ju Jeong, and Dong-Soo Kwon. 2005. Novel haptic mouse system for holistic haptic display and potential of vibrotactile stimulation. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1980–1985.
[76] Wai Yu, Ravi Kuber, Emma Murphy, Philip Strain, and Graham McAllister. 2006. A novel multimodal interface for improving visually impaired people's web accessibility. *Virtual Reality* 9, 2-3 (2006), 133–148.

A WHEELER'S DESIGN ITERATIONS

As mentioned in Section 3, the version of Wheeler we presented in this paper has been a result of three major design iterations. In each iteration, we mirrored the workshop's setup. We briefed participants on the device's objective, notes from previous meetings, and the device's current status. We handed them the in-progress prototype and asked them to navigate the Windows Directory Tree on a computer, first with a screen reader, and then with the device.

Toward the end of the session, we discussed any issues observed and queried the reasoning behind their actions. We also asked for general feedback about the device's current state. We then prioritized the next steps based on the gathered information, making adjustments to the design and adding/removing electronic components.

In the first iteration, we incorporated 3 wheels, 2 buttons, and a buzzer into the base device. In the second iteration, we added haptic feedback and replaced the Arduino Uno with the Arduino Micro, as the former could not issue a mouse hardware event. In the 3rd iteration, we included the `2d-nav` mode, a toggle event to switch between `H-nav` and `2d-nav` modes, text-to-speech readout of the mouse cursor's (x,y) location upon pressing the `CTRL` button on the keyboard, and a two-tone audible to indicate the mouse cursor's current position relative to width and height of the screen.

B MOVEMENT TIME MATHEMATICAL FORMULATIONS

B.1 Acquiring Targets Using the Shortest Distance

Let us consider the scenario presented in Figure 18; where the user is trying to move the cursor from the source to the target object. Here, A is the distance between the source and the target, and W is the width of the target. Jagacinski et al. [36] presented a derivation of the required time for such a move using the first-order lag system of Control Theory. Here is what the formulation looks like:

$$t = \frac{\ln(2)}{k} \log_2\left(\frac{2A}{W}\right) \tag{1}$$

Where, k is called the gain factor, which determines the speed at which the target is acquired [36].

Equation 1 is indeed another formulation of the Fitts' law [28]. According to Fitts' law:

$$t = a + b * ID \tag{2}$$

Where ID stands for Index of Difficulty and,

$$ID = \log_2\left(\frac{2A}{W}\right) \tag{3}$$

B.2 Acquiring Targets Using Rectilinear Distances

In Wheeler's `2d-nav` mode, users use rectilinear movements to go from source (X_0, Y_0) to target (X_1, Y_1). For example, In Figure 19, the users would travel the distances A_1 and then A_2, instead of only A like sighted users usually do. In this section, we try to formulate an equation for movement time using such movements.

In this scenario, we can derive the movement times for A_1 and A_2 separately using Equation 1. If t_1 and t_2 unit times are needed for traversing distances A_1 and A_2 respectively, we can write:

$$t_1 = \frac{\ln(2)}{k} \log_2\left(\frac{2A_1}{W}\right) \tag{4}$$

and,

$$t_2 = \frac{\ln(2)}{k} \log_2\left(\frac{2A_2}{W}\right) \tag{5}$$

If total movement time in this case is T_{rec}, then,

$$\begin{aligned} T_{rec} &= t_1 + t_2 \\ &= \frac{\ln(2)}{k}\left[\log_2\left(\frac{2A_1}{W}\right) + \log_2\left(\frac{2A_2}{W}\right)\right] \\ &= \frac{\ln(2)}{k}\left[\log_2\left(\frac{4A_1A_2}{W^2}\right)\right] \end{aligned} \tag{6}$$

Where, $A_1 = |X_1 - X_0|$ and $A_2 = |Y_1 - Y_0|$.

B.3 Rectilinear Vs. Shortest Path Travel

In Figure 19, If path A was taken instead of $\{A_1, A_2\}$ to reach the target, then the movement time $T_{shortest}$ would be:

$$T_{shortest} = \frac{\ln(2)}{k} \log_2\left(\frac{2A}{W}\right) \tag{7}$$

Figure 18: Acquiring Targets Using Shortest Distance.

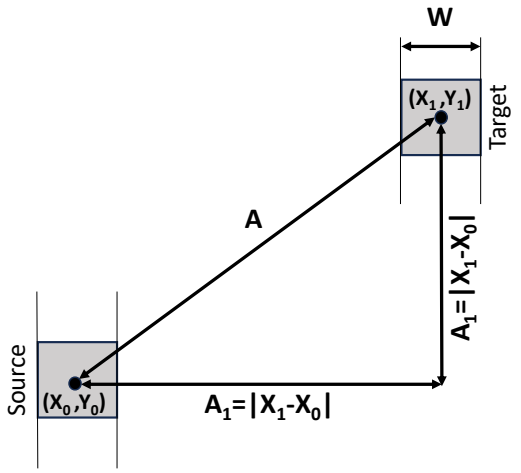


Figure 19: Acquiring Targets Using Rectilinear Movements.

From Figure 19, it is easy to see that $T_{shortest}$ is smaller than T_{rec}. The following calculation shows the difference between T_{rec} and $T_{shortest}$:

$$\begin{aligned}\Delta T &= T_{rec} - T_{shortest} \\ &= \frac{\ln(2)}{k}[\log_2(\frac{4A_1 * A_2}{W^2})] - \frac{\ln(2)}{k}\log_2(\frac{2A}{W}) \\ &= \frac{\ln(2)}{k}[\log_2(\frac{4A_1 * A_2}{W^2}) - \log_2(\frac{2A}{W})] \\ &= \frac{\ln(2)}{k}[\log_2(\frac{4A_1 * A_2}{W^2} * \frac{W}{2A})] \\ &= \frac{\ln(2)}{k}[\log_2(\frac{2A_1 * A_2}{AW})]\end{aligned} \tag{8}$$

Using the Pythagorean theorem, A can be written in terms of A_1 and A_2 as follows: $\sqrt{A_1^2 + A_2^2}$. Therefore, Eq. 8 can be written as below:

$$\begin{aligned}\Delta T &= T_{rec} - T_{shortest} \\ &= \frac{\ln(2)}{k}[\log_2(\frac{A_1 * A_2}{A} * \frac{2}{W})] \\ &= \frac{\ln(2)}{k}[\log_2(\frac{A_1 * A_2}{\sqrt{A_1^2 + A_2^2}} * \frac{2}{W})] \\ &= \frac{\ln(2)}{k}[\log_2(\frac{1}{\sqrt{\frac{1}{A_1^2} + \frac{1}{A_2^2}}} * \frac{2}{W})]\end{aligned} \tag{9}$$

One can interpret the above Eq. 9 as follows:

- If $A_1 \to 0$ or $A_2 \to 0$, i.e., both the source and the target are parallel to X- or Y-axis, the difference in time for rectilinear and shortest movements reduces to zero ($\Delta T \to 0$).
- If $A_1 \to A_2$ or $A_2 \to A_1$, indicating the angle between X- or Y-axis and the line connecting the source and the target becomes 45 degrees, ΔT reaches its maximum value, i.e., $\frac{\ln(2)}{k}[\log_2(\frac{A_1}{\sqrt{2}} * \frac{2}{W})]$ or $\frac{\ln(2)}{k}[\log_2(\frac{A_2}{\sqrt{2}} * \frac{2}{W})]$.
- If one component is significantly larger than other, $A_1 >> A_2$, ΔT depends on the smaller component ($\Delta T \approx \frac{\ln(2)}{k}[\log_2(A_2 * \frac{2}{W})]$, assuming $\frac{1}{A_1^2} \to 0$).

B.4 Rectilinear Travel at different Speeds

Wheeler has the option to modify the cursor speed (via the third wheel) in `2d-nav` mode. This would enable the users to speed up the movements in rectilinear motions if they want to. If the user speeds up the cursor movement by a factor of s (**assuming s>1**), then the effective distances they have to travel become $\frac{A_1}{s}$ and $\frac{A_2}{s}$ instead of A_1 and A_2 respectively. If we call the movement time $T_{rec-speed}$ this time, Equation 6 becomes:

$$\begin{aligned}T_{rec-speed} &= \frac{\ln(2)}{k}[\log_2(\frac{\frac{2A_1}{s}}{W}) + \log_2(\frac{\frac{2A_2}{s}}{W})] \\ &= \frac{\ln(2)}{k}[\log_2(\frac{2A_1}{Ws}) + \log_2(\frac{2A_2}{Ws})] \\ &= \frac{\ln(2)}{k}[\log_2(\frac{4A_1A_2}{W^2s^2})]\end{aligned} \tag{10}$$

Now, let us consider that we pick s in a way so that $T_{rec-speed}$ and $T_{shortest}$ becomes equal. In other words, we want to see how much we have to speed up (s) the movement to achieve the shortest path (sighted) performance from the rectilinear path.

Hence,

$$\begin{aligned}&T_{rec-speed} = T_{shortest} \\ &\implies \frac{\ln(2)}{k}[\log_2(\frac{4A_1A_2}{W^2s^2}) = \frac{\ln(2)}{k}\log_2(\frac{2A}{W}) \\ &\implies \frac{4A_1A_2}{W^2s^2} = \frac{2A}{W} \\ &\implies As^2W = 2A_1A_2 \\ &\implies s = \sqrt{\frac{2A_1A_2}{AW}}\end{aligned} \tag{11}$$

Where,
$A_1 = |X_1 - X_0|$, $A_2 = |Y_1 - Y_0|$, and $A = \sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2}$
Thus, if we have :

(1) the source (X_0, Y_0) and the target (X_1, Y_1) coordinates, and
(2) the width of the target (W),

we can determine how much speed increment in rectilinear movements is necessary to achieve the same performance as that of taking the shortest path.

When s<1: s=1 means we expect the speed in rectilinear movements to be the same as the shortest path movement. However, with some users, the conditions s=1 or s>1 may not be achievable. In Wheeler's `2d-nav` mode, the third wheel also allows the users to slow down the cursor i.e., activate a scenario where s<1. When they do so, the movement time will be higher as indicated by Equation 10.