Towards Multi-Wheel Input Device for Non-Visual Interaction

Noushad Sojib
North East University
Bangladesh, Bangladesh
nsojib@neub.edu.bd

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

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

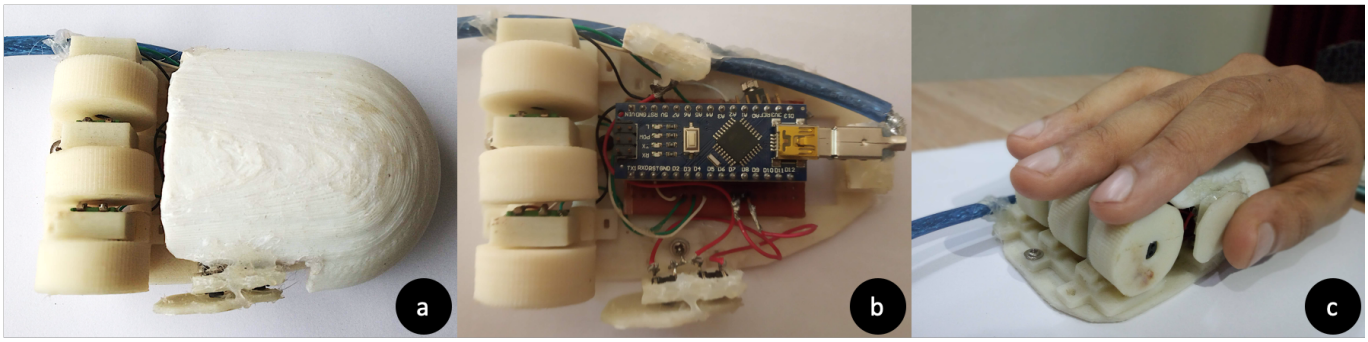

Figure 1. (a) a 3D-printed prototype of proposed multi-wheel based input device, *Wheeler*. It has three rotational wheels, two push buttons on the left side, and audio-haptic capability; (b) internal components and electronics of Wheeler; and (c) a user is using Wheeler non-visually.

ABSTRACT

While sighted users leverage both keyboard and mouse input devices to interact with desktops, non-visual users, i.e., users who are blind, cannot use a mouse as it only provides feedback through a visual cursor. As a result, these users rely on keyboard-only interaction, which is often cumbersome, inefficient, and error-prone. Prior work has shown that using a small, rotary input device benefits blind users significantly, as it simulates mouse-like operations. In this paper, we extend this prior work by proposing *Wheeler*, a multi-wheel based input device that provides simultaneous access to UI elements at three different hierarchies, to facilitate rapid navigation and mouse-like interaction. We designed Wheeler from scratch in multiple iterations and assembled it using 3D printed models and commercially available electronics. A preliminary user study with six blind-folded sighted users revealed its potential to become an essential input device for blind users, as well as a training and learning tool.

Author Keywords

Input device, mouse, multi-wheel, rotation; non-visual, blind.

CCS Concepts

•**Human-centered computing** → *Auditory feedback;* **Pointing; Accessibility systems and tools;**

UIST '20 Adjunct, October 20–23, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7515-3/20/10.
http://dx.doi.org/10.1145/10.1145/3379350.3416168

MOTIVATION

Sighted users typically use a keyboard and mouse to interact with desktop computers. Keyboards are used to enter text and issue quick shortcuts, whereas mice are used for pointing and clicking on user interface (UI) elements on the 2D screen. Unfortunately, non-visual users, i.e., users who are blind, cannot use a mouse, as it provides *visual feedback* through a cursor. For this reason, blind users primarily rely on keyboard shortcuts, in addition to assistive technologies like screen readers [1] that describe screen content aurally.

While this mode of interaction works, the lack of mouse-based point-and-click functions makes it less efficient. Compared to sighted users, blind users take a longer ($\sim 3\times$) time to complete a given task [7, 9]. Memorizing numerous keyboard shortcuts also poses a significant learning challenge; and blind users, regardless of their expertise, could accidentally mix-up one shortcut with another [3].

To overcome these problems, we propose ***Wheeler***, a mouse-shaped, stationary, input device that has three rotary wheels and two side buttons (shown in Figure 1), where user rests her three fingers over the three wheels and her thumb over the two side buttons. Rotating a wheel is akin to moving the mouse cursor, and pressing the side buttons is akin to making a left-/right-click with the mouse, respectively.

WHEELER

Wheeler assumes that UI elements are semantically alike and inherently hierarchical. By leveraging these semantics, one can construct their hierarchies [5, 4]. As an example, Figure 2.c shows how the menus and ribbons of two applications (shown in Figure 2.a-b) can be organized in a tree hierarchy of three levels. Once such hierarchies are constructed, the

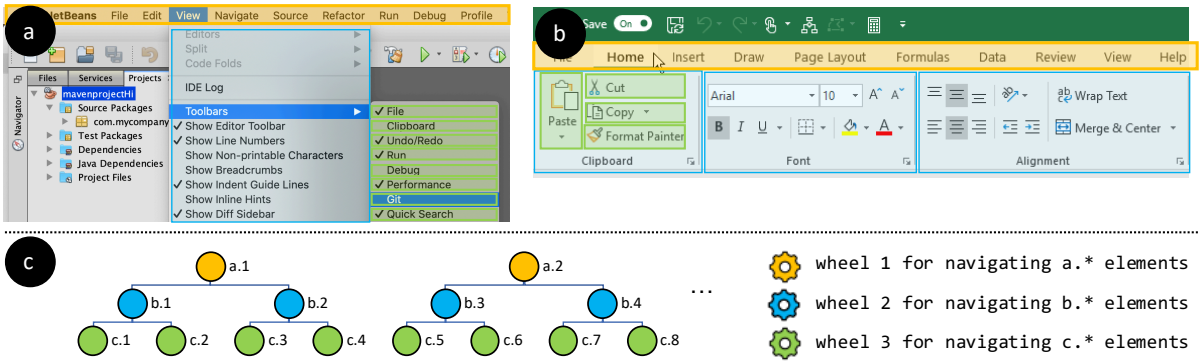

Figure 2. (a) Multi-level menus in NetBeans; (b) nested structures in ribbons in MS Word; (c) a sample 3-level tree hierarchy to represent (a) or (b). E.g., 1^{st}-level elements, `{a.1, a.2, ...}` could represent the top-level menu items like `{Edit, View, ...}`) in (a) or `{Home, Insert, ...}`) in (b).

three wheels of Wheeler, namely, `wheel 1`, `wheel 2`, and `wheel 3`, are dedicated to navigating UI elements in three different levels, namely, 1^{st}, 2^{nd}, and 3^{rd}, respectively. The rotate gesture (e.g., clockwise, and counterclockwise) selects an element in a level bi-bidirectionally. 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`.

Wheeler provides the effect of having three hierarchical cursors. It also maintains the states of which elements were selected last time in a subtree rooted at the element selected by `wheel 1` and `wheel 2`. When revisited, it selects those elements automatically, otherwise selects the first element at any level. To perform a left-/right-click operation, the user presses either of the side buttons.

In order to adjust a wheel's sensitivity, users can define the rotation resolution (in degrees). Upon rotation, Wheeler provides audio-haptic feedback to affirm a valid operation and sometimes to convey spatial information, such as whether a UI element is the last (or first) among its siblings.

PROTOTYPE DESIGN AND IMPLEMENTATION

The design of Wheeler was informed by our previous work, Speed-Dial [2], where Billah et al. demonstrated the effectiveness of a rotatory input device that emulates mouse-like functions for blind users. Unlike Speed-Dial that has one rotary input, Wheeler has three. Wheeler also eliminates the need for manually navigating UI hierarchy by overriding the press operation (e.g., single-press to go down, or double-press to go up in the hierarchy), which was found to be problematic.

We designed Wheeler from scratch and in multiple iterations. Figure 3 shows different modules of our design, along with their dimensions. At each iteration, we sought feedback from the local blind community to ensure user comfort. Based on user feedback, we placed both buttons on the left side of the device and changed their sizes to make thumb-accessible.

Electronic components include one Arduino Nano controller, three rotary encoders, one mini vibrator motor, and one buzzer (shown in Figure 1.b) – all of which cost less than $30 USD.

The device is connected to a PC via a USB cable. Currently, it only works for applications that support plug-ins

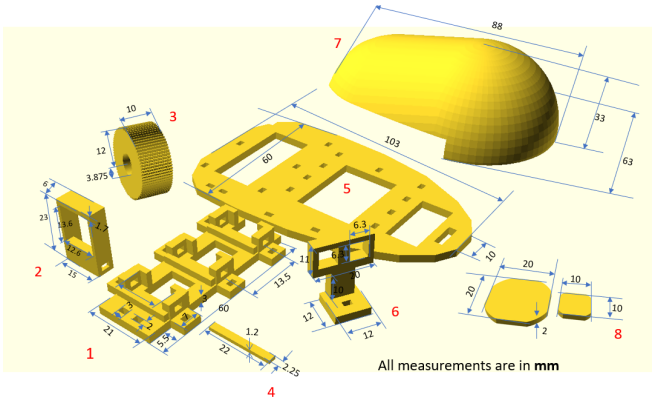

Figure 3. 3D components of Wheeler with dimensions in millimeters.

(e.g., MS Office Suite, NetBeans, and Web Browsers). We wrote application-specific plug-ins to receive rotation and press events from the device. To extract the UI hierarchy of an application, we used Window's accessibility API (e.g., UI Automation [6]).

PRELIMINARY EVALUATION AND FUTURE WORK

Due to COVID-19, we were unable to recruit blind participants to evaluate Wheeler. As such, we conducted a preliminary study with six blind-folded sighted participants, where they were asked to find a given ribbon element (e.g., find `Format Painter` from `Clipboard` group under `Home` in Figure 2.b).

The study revealed that Wheeler is easy to use and learn. The ability to use three wheels concurrently helped participants quickly familiarize themselves with the dense, hierarchical UI structure of an application. Participants commented on the weak integration of Wheeler with Narrator (i.e., screen reader), as it was slow to report selection changes and did not perform audio ducking when participants rotated a wheel multiple times in quick succession. Despite these issues, they commented on its potential to become an essential input device for blind users, as well as a training and learning tool.

In the future, we plan to tightly integrate Wheeler with an open-source screen reader, NVDA [8]. Furthermore, we aim to expand its capability to work with UI hierarchies deeper than three levels. Finally, we plan to conduct a comprehensive study with blind users to measure the effectiveness of our device, both qualitatively and quantitatively.

REFERENCES

[1] AFB. 2018. Screen Readers. https://www.afb.org/node/16207/screen-readers. (2018). Accessed: 2019-04-03.

[2] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and I.V. Ramakrishnan. 2017a. 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. DOI:http://dx.doi.org/10.1145/3132525.3132531

[3] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and I.V. Ramakrishnan. 2017b. 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. DOI: http://dx.doi.org/10.1145/3025453.3025731

[4] Morgan Dixon, Daniel Leventhal, and James Fogarty. 2011. Content and hierarchy in pixel-based methods for reverse engineering interface structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1979086, 969–978. DOI: http://dx.doi.org/10.1145/1978942.1979086

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

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

[7] 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. DOI: http://dx.doi.org/10.1007/s10209-007-0098-4

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

[9] 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. DOI: http://dx.doi.org/10.1145/2488388.2488478