Developing and Deploying a Real-World Solution for Accessible Slide Reading and Authoring for Blind Users

Zhuohao (Jerry) Zhang
The Information School | DUB Group,
University of Washington
Seattle, USA
zhuohao@uw.edu

Gene S-H Kim
Stanford University
Stanford, USA
gene.sh.kim@stanford.edu

Jacob O. Wobbrock
The Information School | DUB Group,
University of Washington
Seattle, USA
wobbrock@uw.edu

ABSTRACT

Presentation software like Microsoft PowerPoint and Google Slides remains largely inaccessible for blind users because screen readers are not well suited to 2-D "artboards" that contain different objects in arbitrary arrangements lacking any inherent reading order. To investigate this problem, prior work by Zhang & Wobbrock (2023) developed multimodal interaction techniques in a prototype system called *A11yBoard*, but their system was limited to a single artboard in a self-contained prototype and was unable to support real-world use. In this work, we present a major extension of A11yBoard that expands upon its initial interaction techniques, addresses numerous real-world issues, and makes it deployable with Google Slides. We describe the new features developed for *A11yBoard for Google Slides* along with our participatory design process with a blind co-author. We also present two case studies based on real-world deployments showing that participants were able to independently complete slide reading and authoring tasks that were not possible without sighted assistance previously. We conclude with several design guidelines for making accessible digital content creation tools.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI); Accessibility technologies**.

ACM Reference Format:
Zhuohao (Jerry) Zhang, Gene S-H Kim, and Jacob O. Wobbrock. 2023. Developing and Deploying a Real-World Solution for Accessible Slide Reading and Authoring for Blind Users. In *The 25th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '23), October 22–25, 2023, New York, NY, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3597638.3608418

1 INTRODUCTION

People today regularly use presentation software like Microsoft PowerPoint, Google Slides, and Apple Keynote for business, education, and creative purposes. These software tools employ slides based on a digital "artboard" canvas, as described by Schaadhardt et al. [43], which can contain various objects such as text boxes, shapes, images, videos, charts, and diagrams. For blind users, *interpreting* existing slides and *generating* new ones both remain largely inaccessible, which contribute to significant educational and professional barriers [43]. To address these challenges, prior work by Zhang & Wobbrock [59] developed a multi-device multimodal system called *A11yBoard* to make digital artboards accessible. Although A11yBoard shed light on interaction techniques that make rich information in 2-D canvases accessible to read and edit using touch, gesture, audio, speech, keyboard input, and search, A11yBoard was limited to a proof-of-concept prototype that worked on an open-source drawing canvas—only a single self-contained artboard. Furthermore, A11yBoard's evaluation was based only on curated usability tasks in a laboratory setting. Although A11yBoard enabled an important initial exploration of accessible artboards, it could not support real-world use. Moreover, the literature is clear that moving from self-contained research prototypes to real-world field deployments inevitably elevates not only practical design and engineering issues, but uncovers new knowledge about the problem domain [47]. Therefore, to further our knowledge of how to design, develop, and deploy accessible artboard creation tools, we created *A11yBoard for Google Slides*, a major extension of the original self-contained A11yBoard prototype.

A11yBoard for Google Slides is a deployable multi-device multimodal system that consists of a mobile touch screen application and a Chrome browser extension (see Figure 1). Created out of a participatory design process with a blind co-author, A11yBoard for Google Slides mirrors desktop slides onto a touch screen device, and enables multimodal interactions to read and edit slide contents. For example, users can employ finger-driven screen reading [21] on the touch screen to explore slide content without fear of altering it accidentally [43]. Audio tones and customized screen reader outputs are displayed in response to a user's (1) touches and gestures, (2) speech commands through the touch screen device, and (3) keyboard commands through an accompanying Chrome browser extension that works exclusively on Google Slides pages.

A11yBoard for Google Slides was created through a participatory design process with a blind co-author over multiple sessions. In this design process, we first identified issues with the prototype version of A11yBoard [59] and explored how presentation software currently works with commercial screen readers.[1] We then repeatedly tested and improved the design through these participatory design sessions. As a result, compared to the original A11yBoard [59], A11yBoard for Google Slides offers more flexibility in slide

ASSETS '23, October 22–25, 2023, New York, NY, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0220-4/23/10.
https://doi.org/10.1145/3597638.3608418

[1] A11yBoard for Google Slides employs its own custom speech output because existing screen readers do not handle slide contents in an accessible manner. For example, on a Microsoft PowerPoint slide, NVDA would read out objects in their Z-order, regardless of their placement on the canvas.

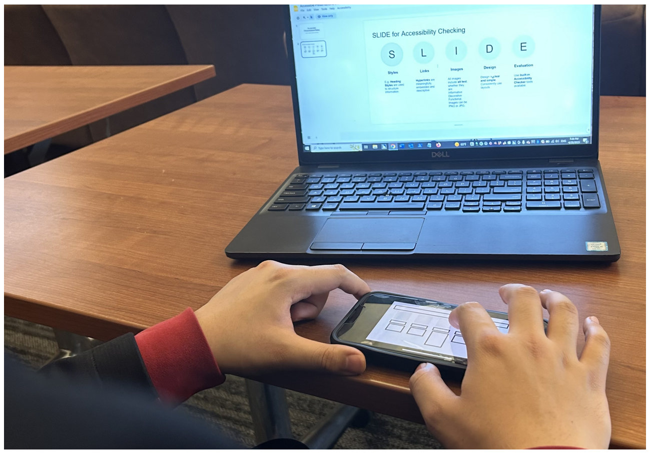


Figure 1: A blind user "finger reading" [21] a slide using *A11yBoard for Google Slides*, which consists of a browser extension and an Apple iOS app. The app shows the slide and enables touch, gesture, and speech interactions with it.

exploration, is more adaptable to blind users' workflow and devices, and is integrated into Google Slides' existing features.

We conducted two case studies as field deployments [47] to evaluate A11yBoard for Google Slides in real-world applications. Two blind participants were recruited and used the tool independently for five and seven days, respectively. They utilized A11yBoard to read and recreate various slide decks, totaling 4.5 hours each, including tutorial usage. Feedback was obtained through interviews, and back-end log data was analyzed. Our results show that participants were able to use A11yBoard for Google Slides to read and create slides independently without sighted assistance, which was a first for both of them. Our results also show that although blind users still feel the need to seek sighted confirmation before they actually use slides in a presentation, A11yBoard for Google Slides greatly reduced the amount of back-and-forth when checking with sighted collaborators. We discuss lessons learned from the design process and evaluation that could inform the future design of assistive technologies for digital content creation. Specifically, we offer design recommendations for making content creation on 2-D canvases more accessible for blind users.

2 RELATED WORK

Prior work related to A11yBoard for Google Slides can be classified into (1) exploration of blind users' experiences with 2-D digital content, including presentation software and beyond, and (2) non-visual interaction techniques for blind users.

2.1 Blind Users' Experiences with 2-D Digital Content

Various approaches have been proposed to facilitate access for blind and low-vision users to 2-D digital content, including digital artboards, formatted documents, visualization charts, images, animations, and videos. Prior work on A11yBoard by Zhang & Wobbrock [58, 59] explored multi-device multimodal interaction techniques to make digital artboards accessible. However, their system had limitations as it was confined to a single artboard in a self-contained prototype, limiting real-world usefulness. Other works also demonstrated similar efforts. AVScript [17] enabled blind users to edit videos using text-based interactions through narration and transcripts. VoxLens [45] provided an inclusive solution for blind or low-vision users to interact with online data visualizations through data sonification and speech recognition. Chart Reader [50] used a navigation flow for screen reader users to explore and read visualization charts through their data insights, axes, data points, filters, etc. Machine learning models were utilized in SciA11y [53], which extracted the scientific content from Adobe PDF files and converted it into an accessible HTML format with additional navigational tools to aid screen readers. Lee et al. [28] demonstrated a multi-layered touch method for exploring digital images with AI-generated captions. Relatedly, Zhang et al. [57] offered Ga11y as a combined machine learning and crowdsourcing solution for annotating animated GIF images with alt-text descriptions. Li et al. [29] explored how blind people adopted non-visual interactions to interact with visual artworks. Peng et al. [39–41] proposed a series of methods to non-visually explore presentation videos, visual design changes in presentation slides, and slide content in an automatic way. However, most of these prior approaches focused on providing a non-visually accessible end-result for blind people to consume, rather than giving access to blind people throughout the authoring process (i.e., agency to dynamically author the content independently). In contrast, A11yBoard for Google Slides is an integrated solution that focuses on both content *interpretation* and content *creation*. This is in keeping with Ladner's [24] call to develop tools for people with disabilities to participate in *all* phases of the design process, including in prototyping and development, not just in user research, ideation, and evaluation [5, 15, 34].

Current presentation software tools like Microsoft PowerPoint or Google Slides provide some built-in accessibility features for blind people. For instance, Microsoft PowerPoint provides screen reader support [30] for blind users to navigate through its user interface elements, including views and ribbon tabs. It also offers a full set of keyboard shortcuts for creating, deleting, rearranging, and organizing slides. But these aspects of PowerPoint exist *outside* the artboard itself, which is a largely unstructured space in which arbitrary objects can exist in any arrangement. As a result, Microsoft PowerPoint is very difficult to use with a screen reader. Google Slides offers more, but is still left wanting. It provides a verbalization of a selected object's content and formatting styles [13]. But it still remains difficult to know which objects are present and to select them for verbalizing. As with PowerPoint, it remains difficult if not impossible for users to "read the artboard" to know what content is present on it. In a related study exploring the accessibility challenges of digital whiteboard tools, Fan et al. [10] found that even when blind users were able to access individual pieces of information on a linked-node diagram, it was cognitively demanding to understand the spatial relationships between individual items, especially with a high degree of confidence.

So, although most presentation software tools provide some accessibility features, they mainly focus on making the software *interface* accessible, rather than making the 2-D *artboard* accessible. In contrast, A11yBoard for Google Slides makes the 2-D canvas accessible through non-visual touch-, gesture-, and speech-based interactions.

2.2 Non-Visual Interactions for Blind Users

We review different input and output modalities that enable non-visual interactions for blind users, including audio, tactile, haptic, and multimodal interactions.

Assistive technologies for blind individuals often use audio interactions, which include speech recognition, text-to-speech, and non-speech audio. Voice assistants and screen readers, such as VoiceOver [1], NVDA [35], JAWS [19], and Windows Narrator [54], enable blind users to access visual elements through speech output. Previous research has also explored various auditory techniques to enhance the accessibility of virtual 2-D spaces, including user interface design [18], graphs [4, 6, 7, 44, 45], maps [8, 9, 46], and documents [27]. Tactile and haptic interfaces have also been shown to support non-visual interactions for blind people. These interactions provide more intuitive representations of graphical and operational information [2, 3, 16, 22, 25, 32, 33, 38, 56]. Previous research has also investigated various forms of tactile and haptic feedback for blind people to interact with maps [48, 49] and graphs [23, 52]. Multimodal designs, which combine audio and tactile interactions, can create more accessible experiences for blind people [11, 20, 42, 51, 52]. In our work here, we employed another set of multi-device multimodal interactions that include touch, gesture, audio, speech, keyboard, and search to create an accessible 2-D slide reading and editing experience. For different scenarios, A11yBoard for Google Slides may provide different interaction modalities. For instance, a blind user can create an object in multiple ways—by drawing it with a finger, using speech commands, or using keyboard commands accessed via search.

3 A11YBOARD FOR GOOGLE SLIDES: A REAL-WORLD DEPLOYMENT

We present a detailed description of the design and implementation of A11yBoard for Google Slides. To appreciate the significant improvements made during our iterative participatory design process, it is necessary to describe the features of A11yBoard [59], which provided a starting point for our current investigations. Subsequently, we provide a summary of the design challenges and considerations that emerged from our participatory design process. This backdrop will then allow us to reflect on the improvements and new features introduced in A11yBoard for Google Slides.

3.1 A11yBoard in Review

A11yBoard employed a variety of multimodal inputs and outputs. It supported touch and gesture to enable a user to interpret an artboard. Blind people could use one finger, the "reading finger" [21], to explore the artboard by touching its mirrored image on the touch screen device, receiving different audio tones as feedback indicating whether they had entered or left an object's borders. Furthermore, speech output revealed objects' shapes. While a user explored an artboard with their "reading finger," they could also split-tap (*i.e.*, a "second-finger tap" issued anywhere on the screen while the first "reading finger" remained on the intended target [21]) to receive detailed information about objects (*e.g.*, their positions, sizes, and colors) as well as to select objects for further action. When a split-tap was performed on empty space, a "dull" audio tone was played and the empty location was selected for further action. Other supported gestures included a two-finger directional flick to discover nearby objects in the flick direction, and a double-tap to traverse objects' Z-order under the current "reading-finger." Finally, a single-finger dwell initiated speech input, like holding down a walkie-talkie button before speaking.

Regarding speech input, A11yBoard allows users to issue speech commands and receive spoken feedback while their finger remains on the screen. The feedback can be either brief or detailed, providing information about object properties such as position, size, color, text, and the closest or farthest objects. Additionally, A11yBoard supports editing operations through speech commands, enabling users to create, move, and resize objects with ease. Unlike typical drag-and-drop methods found in most artboard tools, A11yBoard separates the moving and resizing process into two phases: First, users indicate the object they want to move or resize, and second, they can explore the canvas to find the desired destination, thereby *deferring* the placement decision and reducing cognitive load. Furthermore, A11yBoard facilitates aligning two objects when moving or resizing one towards another.

To ensure blind users could also execute additional commands and edit object properties, A11yBoard also offered a search-driven keyboard interface. This interface allowed users to browse command keywords in an accessible input box and select them with a few keystrokes. Examples of these supported commands included "copy," "delete," "bring to front," "send to back," and many more.

3.2 Design Challenges for A11yBoard for Google Slides

Although the original A11yBoard [59] pioneered a number of useful interaction techniques, it was severely limited as a real-world tool, having only one artboard in a self-contained prototype. It therefore offered no opportunity for real-world use, let alone the ability to support making slide decks in a commercial tool like Google Slides. We therefore set out to create *A11yBoard for Google Slides*, discovering in the process what was necessary for supporting real-world use of accessible artboards. Before we present our A11yBoard for Google Slides system, we summarize six key challenges for designing accessible slide reading and authoring.

3.2.1 Limited Control over Slide Content. When working with our blind co-author, one of the primary challenges we encountered was how existing commercial software like Google Slides gives us very limited control over slide content. Because of this, prototyping a system to control Google Slides is quite challenging. We needed to work around the existing interface and APIs, optimizing A11yBoard's design to fit within the constraints of this commercial software tool.

3.2.2 Moving from a Single Artboard to a Full Slide Deck. Another challenge was transitioning from A11yBoard's single artboard to a multi-slide deck, which is the norm for commercial presentation software. This transition required a redesign of A11yBoard's navigation and editing features to suit this expanded format. Furthermore, we had to develop interactions to support cross-slide operations (*e.g.*, copy an object from one slide for pasting onto another).

3.2.3 More Complex Slide Reading and Authoring Needs. The design of a new A11yBoard experience for blind users needs to consider the more complex reading and authoring needs of slide decks. These needs may include more complex shapes and diagrams, such as arrows and lines, and a large number of slides. Moreover, the system needs to support a variety of operations required for creating, modifying, and presenting slides.

3.2.4 Multi-Device Interference. A11yBoard is a multi-device multimodal system, and as such, there is potential interference of screen readers across different devices. This can cause confusion for the user and interference with the audio output as two screen readers may speak simultaneously. Therefore, in designing A11yBoard, we must consider ways to mitigate the potential for screen reader interference across multiple devices. We must ensure that A11yBoard's audio output is clear and concise, regardless of whether the user is accessing it through their desktop or mobile device, and that it does not interfere with other screen readers the user might be using.

3.2.5 Balancing Efficiency and Expressiveness. Efficiency and expressiveness are two competing priorities that need to be balanced when designing A11yBoard. For example, creating a connector between two objects can have multiple options including whether a line is straight or curved, which line ends have arrows, what are the arrow styles, and what are the line widths and line dash styles. It takes a great amount of unnecessary effort for blind users to indicate these visual properties before creating a single connector. The system must be efficient enough to allow users to complete tasks quickly, while also ensuring that the user's preferences can be expressed fully.

3.2.6 Supporting Individual Differences in Perceptions. Finally, individual differences in perceptions of 2-D artboard information must be considered. Users might have different preferences and requirements for how A11yBoard should work, like reporting values in different metrics (*e.g.*, inches, centimeters, or pixels), or creating and placing objects using different methods or sequences. These individual differences must be accommodated insofar as possible. Therefore, our new A11yBoard system must be customizable and adaptable to suit the diverse needs of blind users.

3.3 Overview: A11yBoard for Google Slides

A11yBoard for Google Slides enhances accessibility with extra interactions for exploring and editing slides. It consists of a web browser extension for Chrome and Firefox, and a mobile app for iOS devices. User authentication involves a four-digit code displayed on the extension, which is entered into the iOS app to connect. The server retrieves slide content using the Google Slides API [14] and sends it to the app for rendering basic shapes. Non-visual authoring operations are validated and applied via HTTP requests to the Google Slides API.

The touch screen device supports touch, gesture, speech input, and speech and audio output. Reading operations like selecting an object or switching slides automatically place the focus on the Google Slides' artboard for further editing.

The system supports speech interactions for accessing detailed object properties and relationships. An intelligent keyboard search interface handles complex operations not easily done via touch, gesture, or speech. Customizable speech outputs are generated in response to user actions.

To avoid conflicts with multi-device screen readers, our system uses a custom text-to-speech technique on the touch screen, allowing desktop screen readers like NVDA and JAWS to work alongside it. This approach ensures all visual elements like text input in the browser extension are accessible to screen readers without interfering with touch and gesture inputs.

3.4 Supported Interactions

We now present A11yBoard's supported interactions in detail, organized by input modality: (1) touch and gesture, (2) speech commands and corresponding feedback, and (3) intelligent keyboard search.

3.4.1 Touch and Gesture. Similar to the original A11yBoard [59], A11yBoard for Google Slides also comprises a mobile application that runs on a touch screen device, providing a safe way for blind users to spatially read slides without fear of accidentally altering them [43]. The objects on the current slide will be shown on the touch screen, enabling exploration via touch and gesture (see Figure 2).

Interpretive Touch and Gestures. A11yBoard for Google Slides supports single-finger reading to explore the slide and a second-finger split-tap to select an object and access more detail. However, audio tone and speech feedback in A11yBoard for Google Slides have been significantly improved over A11yBoard [59] to address the design challenges in Section 3.2.

When exploring a slide using a "reading finger," A11yBoard for Google Slides employs a layered method to notify users about objects' Z-order using different audio tones. For example, users hear a "step-up" sound (notes F-B) when entering an object from the empty canvas. When the user enters an object that overlaps that object, users hear a higher "step-up" sound (notes G-C), indicating that they have entered another object in a "higher" place in the Z-order. The audio tones get progressively higher as users "step-up" into more and more overlapping objects. The same scheme works in the opposite direction when users "step-out" of an object into another overlapped object. Comparing to the original A11yBoard [59], which only had a single step-up and step-down sound, A11yBoard for Google Slides provides much more spatial information by adding richer Z-order feedback.

A11yBoard for Google Slides provides much more detailed reporting for different types of objects. In the original version of A11yBoard [59], when a split-tap happens, all objects are reported with their color, location, and size. However, different object types serve different purposes and should be reported in different ways. For example, in addition to color, location, and size, A11yBoard for Google Slides reports any text inside a shape or a text box. If there are long paragraphs inside an object, A11yBoard for Google Slides will intelligently report a title, a first sentence, or a first bullet point to represent that content. For other objects like a connector, A11yBoard for Google Slides reports what objects are connected by it, and where the starting and ending points are, which matter

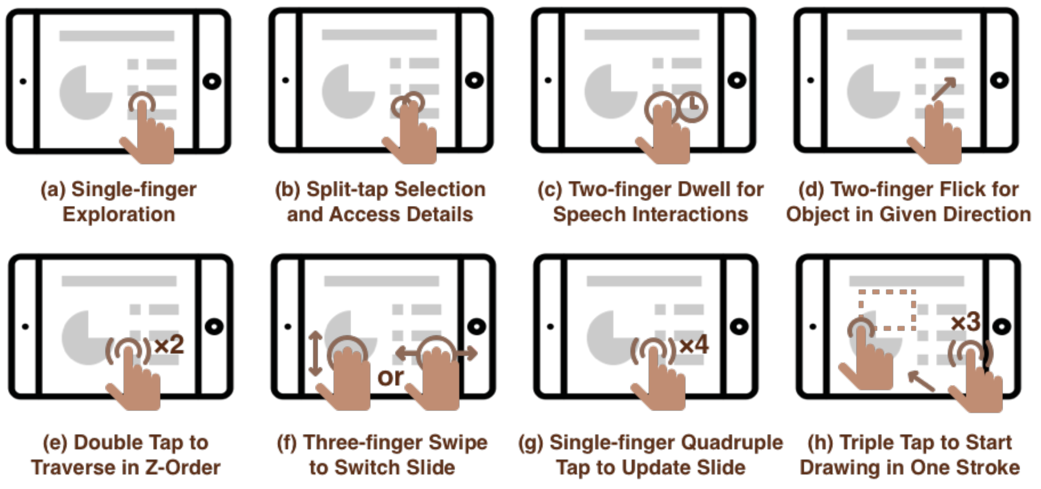

(a) Single-finger Exploration
(b) Split-tap Selection and Access Details
(c) Two-finger Dwell for Speech Interactions
(d) Two-finger Flick for Object in Given Direction
×2
(e) Double Tap to Traverse in Z-Order
or
(f) Three-finger Swipe to Switch Slide
×4
(g) Single-finger Quadruple Tap to Update Slide
×3
(h) Triple Tap to Start Drawing in One Stroke

Figure 2: Eight touch- and gesture-based interactions, including (a) single-finger exploration to spatially "read" artboard objects, (b) split-tap to select an object and access more detail, (c) two-finger dwell to initiate speech recognition, (d) two-finger flick to reveal nearby objects in a given direction, (e) double-tap to step through the Z-order of overlapping objects, (f) three-finger swipe right/down to switch to the previous slide, and swipe left/up to switch to the next slide, (g) single-finger tap four times to update the slide for any other changes, and (h) single-finger triple-tap to start creating an object, followed by a unistroke object drawing.

more than the connector's location and size. An example of speech output for a connector is: "A curved line connecting a text box at top-left corner to a round rectangle at bottom-right corner."

Similar to A11yBoard [59], when an object is selected via split-tap on the touch screen app, that object will become selected on the Google Slide in the desktop web browser. Unlike in A11yBoard [59], where users needed to open the keyboard search interface to perform actions like typing text, they can now perform direct operations on objects, like pressing the Enter key to start typing text into an object.

To support navigating through a slide deck, A11yBoard for Google Slides added a new gesture, a three-finger swipe that switches to the previous slide (by swiping right) or next slide (by swiping left), which is consistent with gestures to navigate pages on the iOS home screen, apps in the app switcher, or images in the Photos app. When users arrive at a new slide, A11yBoard for Google Slides will report the current slide number and an overview of the slide, which includes the number of different objects on the slide.

Another new gesture added to A11yBoard for Google Slides is a single-finger quadruple-tap to actively refresh the touch screen device's view of the current slide. This gesture is for situations when a user makes a change to the current slide using the desktop web browser outside of what A11yBoard provides. After a quadruple-tap, A11yBoard retrieves the current slide's contents and refreshes, providing feedback with a spoken "slide updated" response. Although this situation arises rarely, this gesture provides a way of forcing the iOS screen to refresh.

We decided to employ the original A11yBoard's other gestures, like a two-finger directional swipe to discover the closest object in a given direction, and a double-tap to traverse the Z-order under the current finger location. These gestures were all reported to be useful and straightforward [59].

Generative Gestures. In addition to touch and gestures that serve to interpret slides, A11yBoard for Google Slides also supports a single-finger triple-tap to start creating objects by drawing. After the triple-tap, A11yBoard gives a speech-based notification, "start drawing," to inform users that they should start drawing an object on the canvas. A11yBoard for Google Slides will then recognize the drawn shape by using the $1 unistroke recognizer [55] and then fitting a beautified shape to the drawn trace. The supported shapes are shown in Figure 3. Note that to distinguish between a text box and a rectangle, blind users can draw a big "T" unistroke to represent a text box. The horizontal line represents the top side of the text box, with the width as drawn. The vertical line represents the height of the textbox. A "triple-tap" is required before drawing any object to ensure that blind users can still explore the slide in a risk-free way without worrying about accidentally drawing an object on the canvas. Even if users accidentally trigger object-drawing, A11yBoard for Google Slides also supports a big "X" drawing to cancel the current operation.

3.4.2 *Speech-Based Interactions.*

A11yBoard for Google Slides supports similar speech-based interactions as the original A11yBoard [59], which can be inputted by two fingers dwelling on the screen. To improve the user experience, we added more audio and speech feedback when users are talking to the system. For example, when the system stops talking and starts recording again, or when the system takes some time to process HTTP requests with the Google Slides API, users hear audio feedback like a clock ticking sound. A11yBoard for Google Slides also further enhances the range of

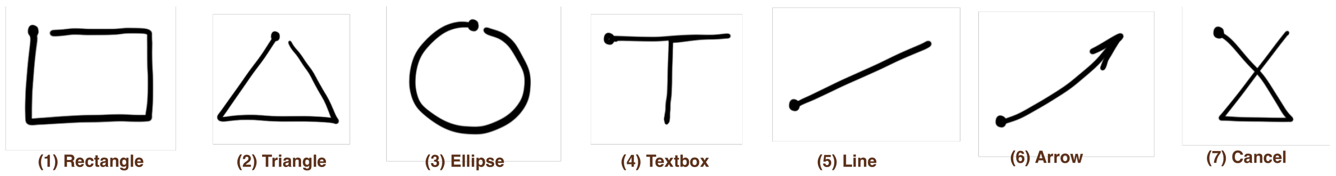


Figure 3: Seven supported shapes that can be drawn as unistrokes [12], including a rectangle, triangle, ellipse, text box, line, arrow, and "cancel the current operation." These unistrokes are recognized with the $1 gesture recognizer [55].

speech outputs by allowing customization. Users can set a speech output mode via a keyboard command. Available modes and their corresponding examples are listed in Table 1.

We divide all speech commands into two categories (see Table 2). First, based on their purpose, the commands can be categorized into Interpretive or Generative commands, meaning those that help users *interpret* existing artboard content or *generate* new artboard content, respectively. Second, depending on whether a command would access or operate on a single object or two objects, the commands can be categorized as Unary or Binary , respectively.

Interpretive unary speech commands include commands like "position," "size," "left," "right," "top," "bottom," "width," "height," and "color," which give the requested properties according to the current reporting mode (see Table 1). An additional keyword, "exact," can be appended to retrieve more precise information. For the "position" and "size" commands, appending "exact" causes the speech output to give exact pixel values. (An exception is when users set the speech output mode to an absolute value in inches or centimeters, appending "exact" gives the output in exact metric values accordingly.) If "color exact" is issued, then RGB values will be reported instead of color names.

Interpretive binary commands include "closest" and "farthest," which report the closest or farthest object, and its direction, from the selected object or current finger position. An example output is, "The closest object is a text box to the south-southwest." A11yBoard for Google Slides uses the closest named directions to report an object's approximate direction. Similar to interpretive unary commands, "exact" can be added after the commands to learn about an object's position and size in pixels, and its direction in degrees. Furthermore, a number can be added after the commands to learn about a number of objects instead of only one. For example, "closest two" requests information about the two closest objects to the finger's position, reported in increasing distance.

A11yBoard for Google Slides also supports a variety of generative commands to create and edit objects. A11yBoard for Google Slides supports generative unary commands like "create," which enables the creation of different types of default objects under the dwelling finger (*e.g.*, "create text box"). Besides creating an object, "move here" and "resize here" are also supported to move or resize an object to a specific position. Particularly, as was described in Section 3.1, "move here" and "resize here" would trigger a two-phase process. First, users would say "move" or "resize" to initialize the moving or resizing process for a selected object or on an empty position. For "resize" specifically, users need to indicate a resizing handle, either a corner (*e.g.*, "top-left") or an edge (*e.g.*, "bottom"), by explicitly speaking this handle name after the "resize" command. Second, users can continue exploring the slide using the full set of touch, gesture, and speech interactions until they find a suitable destination. Users can say "here" to complete the moving or resizing operation.

A11yBoard's generative binary commands, *i.e.*, those that work on two objects while authoring slide content, include "move to align," "resize to align," and "connect with this." Similar to "move here" and "resize here," after an object has been created or selected, the "move," "resize," and "connect" commands enable the object to be moved, resized, or connected to another object on the slide. Generative binary speech commands trigger a two-phase process. After users initiate the moving, resizing, or connecting process by saying the relevant speech command, they can continue exploring the slide until they find another object to align with or connect to. Note that the "here" command is used when users try to move or resize an object to a specific location, whereas the "align" command is used when users try to move or resize an object to align with another object's edge. For example, by saying "align left to left," A11yBoard for Google Slides will resize or move the first selected object's left side to be aligned with the currently selected object's left side. For "connect," users can say "with this" to indicate the object to which they want to connect a first object.

A separate command is "help," which can be used independently or in tandem with any other command. When used independently, A11yBoard for Google Slides will give a quick introduction of available speech commands, which can be stopped at any time by lifting the fingers to exit the speech interaction mode. When used in tandem with other commands, A11yBoard for Google Slides will give a tutorial on how to use the given command, followed by an example.

3.4.3 Intelligent Keyboard Search. To support additional commands that are not easily completed through touch, gesture, and speech, A11yBoard for Google Slides also supports an intelligent keyboard search interface via a browser extension pop-up window, which can be initiated with a preset keyboard shortcut. The interface consists only of a search text box, which embeds a list of supported commands to be selected and executed. Users do not need to remember keywords for this search interface; they only need to type a few characters related to their command. The keyboard commands can also be divided into two categories: Direct editing commands and Navigation commands (see Table 3).

The direct editing commands are implemented as a simplified way to edit objects, like "create," "speech output mode," "bring to front," "bring forward," "send to back," and "send backward." By selecting "create," users are prompted to type in the object type, which creates a default object at the center of the slide. This approach serves as an alternative way to create objects, along with

Table 1: Seven supported speech output modes that can be set to adjust the speech output style and detail, including a brief reporting mode, a detailed reporting mode, a mode that reports properties using relative percentages, one that reports properties using relative fractions, and three others that report properties in absolute values using pixels, inches, or centimeters.

Speech Output Mode	Example Output
Report briefly	"Text box created."
Report in detail	"A new slide created at page 9 with two text boxes inside."
Relative in percentage	"Ellipse moved to 20% of canvas width, 30% of canvas height, with size of 15% by 30%."
Relative in fraction	"From left, about one quarter of canvas width; from top, about two thirds of canvas height; from right, one quarter; from bottom, one eighth."
Absolute in pixels	"Nearest object at 45 degrees is a text box at (528, 491) with size of 200 by 100."
Absolute in inches	"Triangle created at 2.7, 3.9 inches with bounding box of 1.5 by 2.0 inches."
Absolute in centimeters	"Rectangle resized to 3.5, 23.6 centimeters with size of 15.5 by 21.3 centimeters."

Table 2: Speech commands for interpreting and generating objects, including their types, functions, and usage.

Speech Command	Type	Function	Usage
Position (or Left, Right, Top, Bottom)	Interpretive Unary	Report position of an object	Use directly or append "exact"
Size (or Width/Height)	Interpretive Unary	Report size of an object	Use directly or append "exact"
Color	Interpretive Unary	Report color of an object	Use directly or append "exact"
Closest	Interpretive Binary	Report closest object(s) of an object or a position	Use directly, append "exact," and/or append a number
Farthest	Interpretive Binary	Report farthest object(s) of an object or a position	Use directly, append "exact," and/or append a number
Create	Generative Unary	Create an object by type	Append a supported type
Move (A to) here	Generative Unary	Move an object to a position	Use "here" at final destination to trigger
Resize (A to) here	Generative Unary	Resize an object to a position	Append a handle after "resize," use "here" at destination
Move (A to) align (with B)	Generative Binary	Move to align with another object	Use "align [edge] to [edge]" at target object
Resize (A to) to align (with B)	Generative Binary	Resize to align with another object	Use "align [edge] to [edge]" at target object
Connect (A) with this (B)	Generative Binary	Connect with another object	Use "with this" at target object
Overview	Interpretive	Report overview of the current slide	Use directly
Help	N/A	Report tutorial of any given command or in general	Use directly or with any other command

speech commands or finger-drawing an object's shape. Users can also change the speech output style to one of the modes in Table 1. Other direct editing commands are used to change objects' Z-order. They can move an object forward or backward, or send an object to the top or bottom layer. We did not include commands to copy, paste, or delete objects because Google Slides already supports copy, paste, and delete with typical keyboard shortcuts like Ctrl+C, Ctrl+V, and backspace. These shortcuts are already made accessible for blind users.

Another type of keyboard command is used for navigation. Instead of developing a complex editing interface on our own, we utilize the existing interfaces in Google Slides and enable users to navigate to their desired panel by simulating mouse clicks. These panels already contain accessible elements that are labeled for screen readers, but are usually hard to navigate inside the complex visual interface. For example, when an object is selected, users can type in "fill color" to navigate to the fill color panel of this selected object. Users can then select from the pre-defined colors with color name labels or type in exact RGB values. The list of supported commands include "fill color," "border color," "border width," "font family," "font size," etc.

Table 3: Twelve supported keyboard commands including their types and functions that help blind users edit a slide.

Keyboard Command	Type	Function
Report mode	Direct editing	Change the reporting mode as described in Table 1
Create	Direct editing	Create an object with a type selected in a second input box
Fill color, Border color, Border width, Font family, Font size	Navigation	Navigate to the corresponding panel
Bring to front or forward	Direct editing	Bring the current object to front or forward stepwise
Send to back or backward	Direct editing	Send the current object to back or backward stepwise
Insert image	Navigation	Navigate to the insert image window

4 PARTICIPATORY DESIGN OF A11YBOARD FOR GOOGLE SLIDES

In this section, we take a step back to present the participatory design process of A11yBoard for Google Slides. To reconsider and improve upon the original design of A11yBoard [59] in making it suitable for real-world use, we carried out a series of participatory design sessions with the involvement of a blind co-author, GK. The objective of these sessions was twofold: first, to foreground the challenges that must be addressed to enable A11yBoard to work as a real-world tool within Google Slides (see Section 3.2); and second, to develop and refine a fully functional system that would be well-suited to the needs of blind users.

4.1 Method

GK, a co-author on this paper, collaborated with the other authors over the entire design process. GK was born legally blind and has been completely blind for seven years. He is an undergraduate majoring in Symbolic Systems[2] with years of experience using presentation software like Microsoft PowerPoint and Google Slides with a conventional screen reader. GK's expertise in using these tools was gained through his extensive use of presentation software in college courses, where he has often collaborated with classmates to deliver live presentations in class.

Our participatory design process contained two main stages. First, we conducted an interview with GK to create a shared understanding of how we should design A11yBoard [59] for real-world presentation needs rather than just as a self-contained prototype with a single artboard. The interview began by reviewing A11yBoard's existing features. GK offered reactions and suggestions while exploring each feature.

Second, we designed and implemented an initial prototype of A11yBoard for Google Slides over eight weeks, followed by three iterative design sessions with GK over four weeks to improve the usability and functionality of A11yBoard for Google Slides. A description of the system features can be found in Section 3. An overview of the insights we gained from our design process appears in the section below.

4.2 Insights

In the interview, we discussed limitations of the prior version of A11yBoard [59], which included that it was limited to a self-contained single artboard that could not be saved or shared. Also, it was limited to basic shapes like text boxes, rectangles, triangles, and ellipses. In Microsoft PowerPoint and Google Slides, the set of objects is more extensive and objects can have more complex properties, such as rotation, borders, and rich formatting. Another limitation about speech interactions was that although GK was comfortable with how A11yBoard reported object properties, different individuals might have different preferences for how to perceive numeric and descriptive information. Furthermore, GK mentioned that some operations via speech were inefficient when done frequently, like creating objects. He suggested implementing gesture-based object creation as an alternative, which we did.

After we developed the initial prototype, GK guided the other authors in iterative system evaluation and design. We present the insights about how we improved A11yBoard for Google Slides below.

Design more tailored slide exploration. We discovered the need to design more tailored interactions for a better slide-exploration experience. Our prototype provided detailed announcements when accessing objects, but testing with GK showed that more customization options were necessary. For instance, a "brevity mode" could minimize speech announcements and reduce cognitive load. We also enabled other metrics, such as absolute values in centimeters and relative values in fractions of slide width and height. GK also suggested adding a "help" command to provide guidance, assigning different pitches for objects that overlap, and reading out different objects differently based on their typical usage.

Adapt better to users' workflow and devices. We also gathered insights on how to better accommodate the needs and workflow of blind users. One insight was to provide more audio or speech feedback to aid users in operating the app on their devices. GK suggested adding speech reports and audio feedback for system notifications, like entering or leaving the app, and indicating any unrecognized operation. Another insight was to improve the system's recognition of gesture and speech inputs. GK found that the old interaction of using one finger to dwell on the screen was easily misinterpreted into a "finger reading" action. We enhanced the system's tolerance of these inputs to better match the exploration habits of blind users. Lastly, using A11yBoard for Google Slides together with other software can be challenging for blind users, leading to unintended operations and extra cognitive load. GK suggested adding a keystroke to reset and refocus the system, allowing users to recover from accidental movements and continue using the

[2]https://symsys.stanford.edu/

app confidently. This recommendation aligns with previous studies that have shown the need to address user concerns regarding system fragility [43].

Consider trade-offs between accessibility and usability. Incorporating all features of the original A11yBoard system [59] could enhance the accessibility of Google Slides, but its full usability in this new context was initially uncertain. For example, while the original intelligent keyboard search allowed color editing in its interface, Google Slides already offers accessible elements for the "fill color" panel, which can be read by current screen readers. GK proposed a more effective approach of navigating to existing accessible panels instead of introducing additional interfaces for property editing. This strategy improved usability, avoided confusion from extra interfaces, and underscored the importance of assessing an application's existing accessibility features and integrating them with new tools to strike a balance between accessibility and usability.

5 FIELD DEPLOYMENT

For our field deployment [47], we used a case study methodology with two blind users [26]. The goal of our case study was to assess whether blind users can integrate *A11yBoard for Google Slides* into their own workflow and use it to read and author slide decks independently. To achieve this goal, we conducted two field deployments in which blind participants used A11yBoard for Google Slides freely, without any supervision or interference from researchers, over several days. In each case study, we provided a tutorial session to introduce the tasks and the system, and then allowed the participants a few days to complete a task on their own. Once the participants indicated that they were finished, we conducted an interview to collect their final artifacts and feedback about A11yBoard for Google Slides. We also conducted an empirical evaluation of their activities by analyzing the back-end log data and their verbal responses to understand how they used the system.

5.1 Participants

Each case study involved one blind participant (P1, P2 accordingly), recruited through personal communications from our blind coauthor, GK. P1 and P2 both reported being blind since birth, with P1 having some light perception. P1 was a 24-year-old female government relations analyst at a non-profit accessibility organization, while P2 was a 30-year-old male graduate student studying design. Both participants had prior experience using presentation software like Microsoft PowerPoint and Google Slides and used Apple iOS devices and Windows desktop computers or laptops with JAWS screen readers. P1 used presentation software on a weekly basis as part of her job, while P2 used it for university courses on a monthly basis. Participants were compensated $30 for each hour they spent in the study, including the tutorial session, usage of A11yBoard for Google Slides, and the follow-up interview.

5.2 Apparatus

The apparatus deployed and tested in this study was the A11yBoard for Google Slides system, as described in Section 3. Both participants used their personal Apple iPhone and Windows laptop devices when working with A11yBoard for Google Slides. We instructed both participants to turn off their iPhone's VoiceOver software after opening the A11yBoard mobile app. Additionally, both participants used JAWS to interact with Google Slides and our A11yBoard browser extension.

The study task was to understand a slide deck and create a new slide with same or similar content inside the same slide deck. The first study's slide deck (depicted in Figure 4a) was about a fundamental concept in computer science, conditional or if-then statements, which was presented in a flow chart that had five shapes, five connectors, and two text boxes. The second study's slide deck (depicted in Figure 4b) was about guidelines of making slides accessible, including five guidelines presented as shapes with a capital letter inside, and 10 text boxes positioned beneath them.

5.3 Procedure

The study involved three phases. In the initial phase, each participant had a 90-minute tutorial session individually with the authors. P1's session took place on Zoom, while P2's was in person. Before each session, participants answered demographic questions and gave verbal consent. They were instructed to install two required software components on their devices: an iOS app and a Chrome browser extension. During the tutorial, participants received a Google Slides document and a Google Docs tutorial with a "cheat sheet" containing simplified information about A11yBoard's commands. Researchers demonstrated the system's features and had participants try them out. Usability issues were noted and tips were shared to address them. An exit interview took place after seven days (for P1) and five days (for P2). During the interview, participants discussed their experiences, the system's usefulness, task completion, and comparisons with other tools, along with potential improvements.

5.4 Analysis

The data analyzed in this study consisted of four parts: (1) observational results from the tutorial session, (2) the final slides made by participants, (3) back-end time-stamped log data indicating how P1 and P2 used the system, and (4) the exit interviews and feedback about participants' experiences. We report each of these parts separately in Section 6.

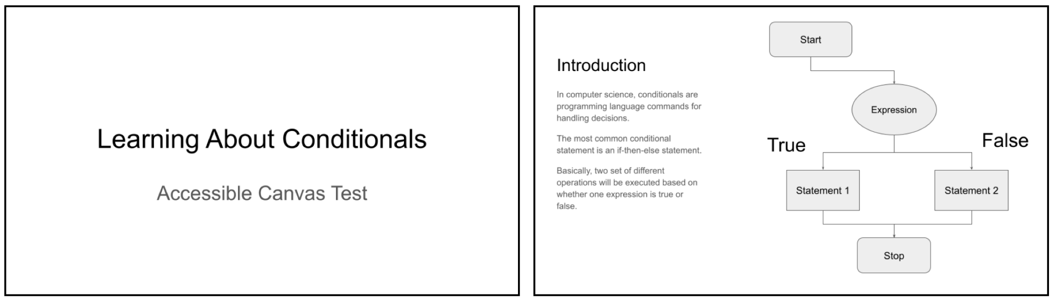
6 RESULTS

We present the results from each case study in turn, focusing on the slides participants made, their use of the A11yBoard for Google Slides system, and their subjective impressions and feedback characterizing their experiences.

6.1 Case Study 1

In this study, after the tutorial session, P1 was instructed to first interpret the slide content and comprehend its structure, and then replicate a similar slide deck. P1 was permitted to use any existing features of Google Slides with the assistance of A11yBoard for Google Slides.

6.1.1 Final Artifact. Overall, the slide created by P1 (Figure 5) contained all the information in target slide 2, with correct content and objects in the correct order and flow. This result indicated that P1 had a complete understanding of the spatial layout of the original



Learning About Conditionals

Accessible Canvas Test

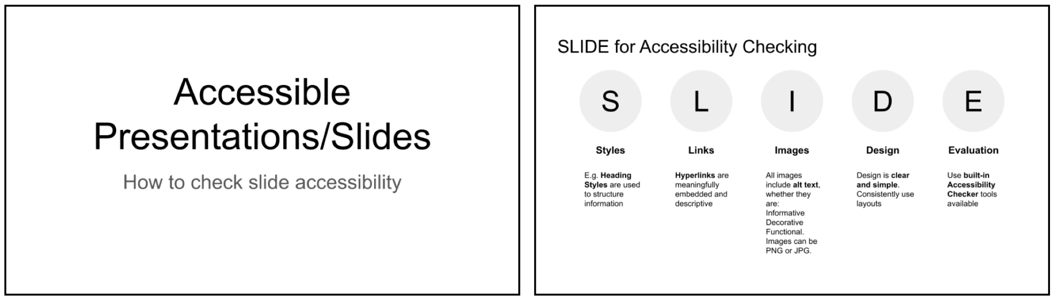
Introduction

In computer science, conditionals are programming language commands for handling decisions.

The most common conditional statement is an if-then-else statement.

Basically, two set of different operations will be executed based on whether one expression is true or false.

(a) Slides used for field study 1



Accessible Presentations/Slides

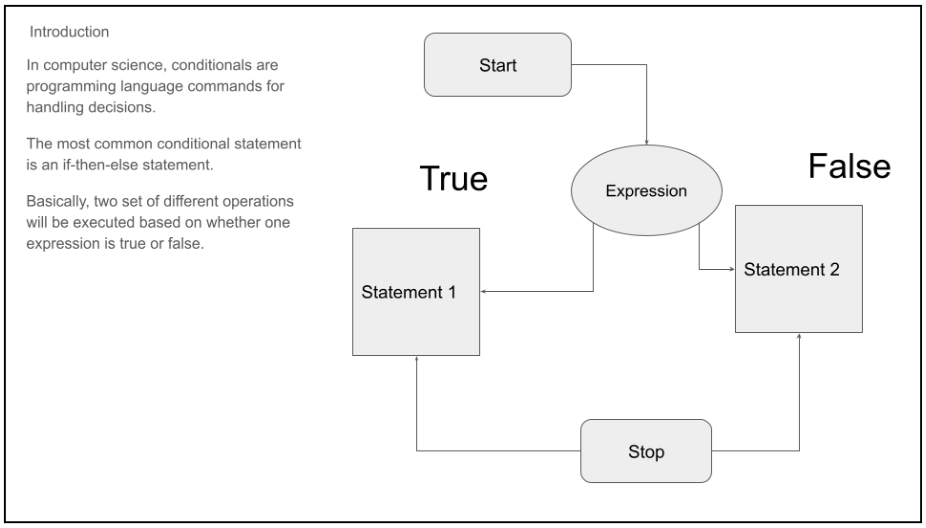
How to check slide accessibility

SLIDE for Accessibility Checking

S	L	I	D	E
Styles	**Links**	**Images**	**Design**	**Evaluation**
E.g. **Heading Styles** are used to structure information	**Hyperlinks** are meaningfully embedded and descriptive	All images include **alt text**, whether they are: Informative Decorative Functional. Images can be PNG or JPG.	Design is **clear and simple**. Consistently use layouts	Use **built-in Accessibility Checker** tools available

(b) Slides used for field study 2

Figure 4: Two slide decks used in two field studies. Each deck contained two slides at the beginning, and participants were instructed to read, understand, and create new slides in the same deck.



Introduction

In computer science, conditionals are programming language commands for handling decisions.

The most common conditional statement is an if-then-else statement.

Basically, two set of different operations will be executed based on whether one expression is true or false.

Figure 5: The slide made by P1 in an effort to recreate the target slide shown in Figure 4a, above. The slide is largely an accurate reproduction except that the two connectors at the bottom have arrowheads on the wrong ends.

slides and could create objects accordingly. However, the created slide also revealed some issues. For instance, the flow chart on the right side displayed the correct workflow, but the two arrows at the bottom pointed in the wrong direction. Additionally, the title text box was not formatted with larger fonts as it should have been, and the title and paragraph text boxes were not quite aligned.

6.1.2 Overall Impressions and Feedback. P1 reported that after the 90-minute tutorial session, it took her an additional 3 hours and

15 minutes to complete the task using our system. Specifically, she spent around 60 minutes familiarizing herself with the touch, gestures, and speech commands, followed by around 45 minutes reading and exploring the slides. Finally, she spent approximately 90 minutes creating the slide and its contents.

As a regular Microsoft PowerPoint and Google Slides user, P1 mentioned that A11yBoard for Google Slides helped her to read the slides more effectively and provided her with the ability to create objects freely. Comparing to her previous experiences, P1 noted that without A11yBoard for Google Slides, she was limited to working with text and had no ability to edit the visual layout of a slide with objects:

> *"There are so many visualizations that screen readers cannot deliver to you" (P1).*

Without A11yBoard for Google Slides, P1 would have needed the help of a sighted co-worker or assistive services like Aira to deal with the visual layouts, images, and charts. P1 reported that producing a slide deck like this all by herself would have been impossible. In particular, she appreciated how she could use a "reading finger" to explore the slides and was "very confident" in understanding the visual content. Another aspect of A11yBoard that P1 enjoyed was the ability to explore the slides in a tactile way and to edit the slides more accurately by using the physical keyboard.

In the opening tutorial, P1 pointed out several usability issues that hindered her from using the system fluently, including gesture and voice misrecognition. Apart from these usability issues, P1 also expressed her feelings about performing editing operations. She thought that there was still too high a cognitive load involved in editing operations. Specifically, she pointed out that she would need to pay attention to the operation itself *and* the spatial position where the operation occurs, which can be challenging. For example, when drawing to create an object, there was no intermediate feedback until she finished drawing in one stroke and heard confirmation of the shape she created.

We discussed with P1 whether A11yBoard for Google Slides could fit into her workflow and how it might be improved in the future. P1 expressed a need for more instant and real-time tactile feedback or confirmation when performing an editing operation, so that she could feel more confident when using the system. P1 also expressed that she would love to use the system in her daily workflow if the interfering gestures from iOS could be mitigated. P1 further pointed out that having a more physical layout beyond the current touch screen and supplementing it with a braille display would be helpful, which is an interesting venue for future work.

6.1.3 Performance and Activities. In view of the final artifact alongside P1's verbal statements and the back-end log data, we could reconstruct the process of how P1 used A11yBoard for Google Slides to complete the assigned task. To begin, P1 explored the canvases on slides 1 and 2 to understand their contents. This exploration was not strictly separated from the editing process, which was happening throughout her usage of the system. Specifically, P1 frequently used split-taps to examine objects' contents, positions, and sizes in detail. P1 then created several slides and experimented with the system's functionality. During this process, P1 confirmed how A11yBoard for Google Slides works by realizing that the objects would be selected automatically in her laptop browser, and she just needed to press the Enter key to start typing text inside them.

Next, P1 created two text boxes, a title and an introduction text box, on the left side of the slide by triple-tapping on the screen and then drawing uppercase unistroke "T" letters. P1 also used speech commands to create two shapes, the rectangles containing "Statement 1" and "Statement 2," on the right side of the screen. Because the default shape size was 100 by 100 pixels, the two rectangles were in their default size and were not resized by P1. For the other shapes, P1 chose to copy and paste the "Start," "Expression," "Stop," "True," and "False" objects directly from the original slide. We acknowledge that copying and pasting objects is within the purview of free and unfettered usage of A11yBoard for Google Slides, and this activity showed that A11yBoard for Google Slides served as a complement to the existing Google Slides system, not a replacement, which fits our expectation. After copying and pasting objects, P1 then created five connectors in sequence to build the flow chart. P1 first connected "Start" to "Expression," then connected "Expression" to "Statement 1" and "Statement 2." Finally, P1 created the last two connectors from "Stop" to "Statement 1" and "Statement 2," which is opposite the intended direction.

As for the keyboard interface, P1 did not utilize the keyboard search interface much for navigating among interface panels or performing editing operations. When asked about this, P1 responded that there was not much formatting needed, as the main focus was on ensuring the content was correct. However, when prompted to recall the tutorial session, P1 acknowledged that using the keyboard search feature would have been helpful for editing object properties and navigating through panels.

In conclusion, while there were some usability issues and inherent limitations in A11yBoard for Google Slides as a real-world solution for making 2-D content accessible, P1 was still able to successfully read and edit her slides independently. Vitally, P1's accomplishment transformed a formerly "impossible task" (her words) into a possible one.

6.2 Case Study 2

Similar to study 1, the second case study also involved understanding and recreating a slide deck, which is about design guidelines for making accessible presentations. Unlike the first case study, which involved understanding the flow of a connected shapes diagram, the main challenge in this case study was to understand the layout of the five object groups, their corresponding positions, and how to create, edit, and align them correctly.

6.2.1 Final Artifact. Figure 6 shows the final slide created by P2. The slide deck created by P2 captured most of the content in the original slide, including the five shapes that contained the five design guidelines, with five text boxes positioned under each elliptical shape. This indicates that P2 was able to understand the layout of the slide and recreate the objects in the correct order and position. However, there were a few imperfect details in the recreated slide. Firstly, the five text boxes at the bottom that explained each design guideline were missing. P2 explained that he skipped creating them due to personal time constraints, as creating a set of five more text boxes was a redundant process to what P2 had already done. Secondly, the five elliptical shapes were not exactly the same size, and

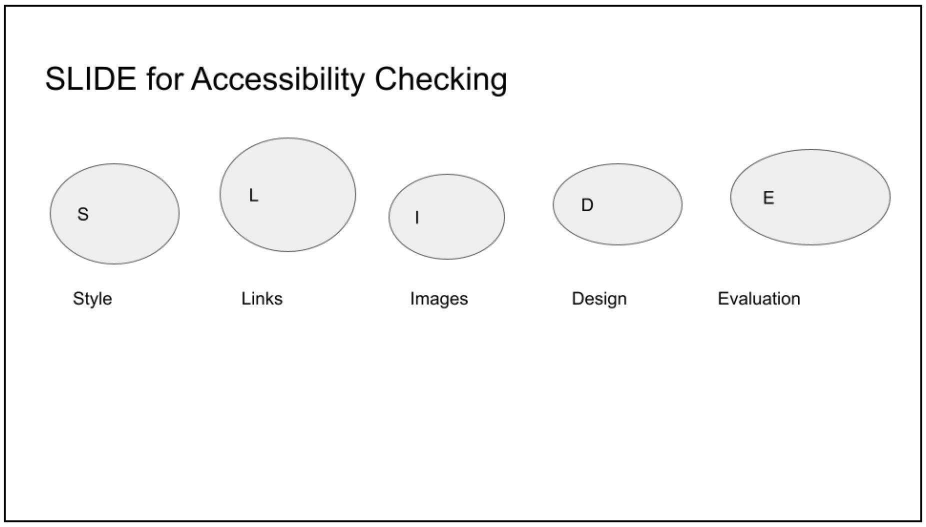

SLIDE for Accessibility Checking

Figure 6: The slide made by P2 in an effort to recreate the target slide shown in Figure 4b, above. Note that five text boxes of explanation were missing because P2 reported that creating a similar set of text boxes were trivial and he did not want to repeat.

they were not perfectly aligned. Finally, the letters inside the shapes were not formatted in the same way as the target slide. But overall, the recreated slide by P2 demonstrated a good understanding of the original slide's layout and content.

6.2.2 Overall Impressions and Feedback. P2 reported that after 90 minutes of the tutorial session, he spent 3 hours completing the task. He took full advantage of the tutorial document and explored all possible touch, gesture, and speech commands for around 1.5 hours to understand the system and the target slide's layout. P2 then created a new slide using speech commands and completed the task after another 1.5 hours.

P2 compared his prior experience of using presentation software with this experience of using A11yBoard for Google Slides. He said that previously, he could only make slides out of existing templates and copy and paste text from written documents into slides with a title and a paragraph text box. Any other slide layouts, or using objects other than text boxes, were simply impossible for him to attempt. With A11yBoard, he was able to perform editing operations on objects, which was groundbreaking for him.

> *"I won't be able to create this kind of slide before [using A11yBoard for Google Slides]. I would have to take visual assistance [without the system]"* (P2).

P2 also raised some usability issues that happened during the field deployment, including the same issue P1 had of accidentally leaving the A11yBoard iOS app because of the iOS's swipe-up app-switching gesture. P2 also faced the challenge of drawing smaller objects, like a small text box on a relatively small Apple iPhone screen, which is the universal fat finger problem that can be mitigated on a bigger touch screen like an Apple iPad.

P2 said that A11yBoard definitely would be beneficial when working with Google Slides, given that the task was not accessible at all without A11yBoard. He said that he would not be able to create a slide deck with so many objects without sighted assistance. He did say that he would still want sighted assistance for final confirmation of his slides after using A11yBoard if he were to deliver a presentation, but that the use of A11yBoard would significantly reduce the time needed to interact with a sighted assistant from Aira or his co-workers.

> *"I would like to spend as much time as possible working on slides independently without sighted assistance. Right now I think I have at least 90% to 95% independence [with A11yBoard for Google Slides]"* (P2).

P2 also believed that A11yBoard could not only be used in business and educational settings but also by middle- and high-school students as the first tool for them to interact with 2-D canvases.

As for improvements, P2 pointed out the same issue as P1, which is the challenge of executing an operation while also maintaining spatial awareness of nearby objects. P2 would like a solution involving a physical layout, such as using Wikki Stix [3] to represent objects. In this case, he would be more confident in editing one object while being able to touch and sense other objects nearby in a physical form.

6.2.3 Performance and Activities. We studied how P2 created his slides using A11yBoard for Google Slides. Initially, P2 created a new slide and entered a title in the default title text box. He then focused on the top-half of the slide and created five elliptical shapes by drawing them left to right. P2 was mindful of aligning the objects and tried to keep the size consistent. He later realized that he could copy-and-paste the shapes for consistency. P2 then turned to the laptop and typed letters in each shape. After completing the top-half, P2 repeated the same process for the bottom-half of the slide. However, he stopped before creating the remaining five text

[3] https://www.wikkistix.com/

boxes as he considered it a trivial but repetitive task, given he had already shown he could create text boxes and enter text into them. During the exit interview, P2 confirmed that he was able to read and edit the slides independently, demonstrating that A11yBoard for Google Slides has successfully made previously inaccessible canvases accessible.

In hindsight, P2 expressed that he could have created the slide more efficiently if he had considered using copy-and-paste sooner. Despite a few usability issues raised by his field deployment, P2 believed that A11yBoard could be beneficial in his daily work by significantly reducing time needed for sighted assistance.

7 DISCUSSION

In this section, we reflect on what the A11yBoard deployments have taught us and we discuss the broader implications of making 2-D presentation tools accessible to blind users. We present design recommendations drawn from our own participatory design and deployment process for future use by designers and researchers. We also discuss A11yBoard's limitations and avenues for future research.

7.1 Design Recommendations

We present five design recommendations for making 2-D canvases accessible to blind users. They arise from our participatory design and deployment efforts. We offer them in hopes that they might serve as a resource for future designers and researchers.

7.1.1 Provide Spatial, Intuitive, and Immediate Feedback. We gained a significant insight into the importance of providing spatial and intuitive feedback to blind users. This is due to the complexity of 2-D content, which is inherently challenging for screen readers. With the advancement of touch screen devices and their haptic features, it is increasingly feasible to provide intuitive and immediate spatial feedback for 2-D objects on the touch screen. However, a challenge remains in making sense of 2-D content semantically. For instance, we found that blind participants could easily comprehend a flow chart when they were provided with a brief introduction. Nevertheless, when presented with a new 2-D space, blind users must take considerable time to explore that space and understand its objects, properties, and relationships.

7.1.2 Tailor Feedback Based on Context and Individual Differences. Different objects serve different purposes. When exploring a 2-D artboard through touch and gesture, it is crucial to emphasize different object attributes when delivering audio or speech feedback to efficiently convey the most significant information to blind users. For example, the positions and sizes of rectangles are important, but the positions and sizes of connectors are less important than the rectangles they might connect. Additionally, individual differences in human perception should be considered, and customization options should be provided accordingly. This priority aligns with how all screen readers offer the ability to adjust speech verbosity. Apart from verbosity, metrics and other attributes should also be included in the customization options.

7.1.3 Provide Multimodal Ways to Create and Edit. During our participatory design process, we discovered the significance of providing multimodal options for creating and editing objects, as these "authoring" operations can occur at any point while the user works. For instance, creating an object can happen when a user has just finished typing on the keyboard or while exploring the 2-D canvas. In either case, the user might prefer to continue with their current workflow and avoid switching devices. Thus, it is crucial to offer an accessible means to create and edit objects through each modality, ensuring that the designed system is flexible in this way.

7.1.4 Consider the Role of AI-Generated Content (AIGC). The recent advancements in AIGC technology, such as Microsoft Office Copilot [31] and GPT-4 [36, 37] based models, have made creating 2-D visual content, such as slides, effortless. These tools can generate high-quality visual content from inputs, including prompts and data, which can significantly benefit blind users in creating visual content. However, with such tools, it becomes even *more* crucial to provide blind users with access to the auto-generated content, as A11yBoard for Google Slides does, to ensure they have agency and control over the visuals. Further research is required on how to tailor these AIGC tools to suit accessibility needs. Additionally, AIGC can be utilized to comprehend and standardize users' natural language input as operations, creating a true "virtual assistant" to assist blind users in reading and creating visual content.

7.1.5 Balance Cognitive Load and Functionality. Another crucial lesson we learned from our design process is the importance of balancing cognitive load and functionality. Manipulating 2-D content can be challenging, as software such as Adobe PhotoShop or Illustrator requires users to possess professional skills that are acquired over years. The full range of functionality can create a significant cognitive load for users, even for software that is relatively feature-light, such as Microsoft PowerPoint and Google Slides. While the initial effort of learning a complex but useful system should not be perceived as a barrier, we still need to meticulously design our system's functionality to avoid overwhelming users with all possible features, rendering it unusable.

7.2 Limitations

One limitation of our study is that it only involved the task of reproducing a few existing slides. We acknowledge that the real-world scenarios people encounter could be more complex and varied. Therefore, while our study provides valuable insights into the design of an accessible 2-D slide creation tool for blind users, further research is needed to fully evaluate the usability and effectiveness of A11yBoard for Google Slides in a greater range of scenarios. Also, the valuable contributions of our blind co-author, GK, primarily pertain to a specific subgroup within the blind community, characterized by individuals with similar needs. While their expertise has been instrumental in addressing the requirements of this particular user group, it is important to acknowledge that their insights may not be universally applicable to the entire blind community.

In addition to the study limitations, there are also some technical limitations to our system. One limitation is the lack of cross-device undo and redo functionality. This is because the Google Slides API does not provide these features, and it is not possible to undo an operation performed on a mobile device from a desktop computer. This limitation could be a source of frustration and confusion for blind users who switch between devices frequently or who accidentally make a mistake and need to backtrack. We recognize that this is a significant usability limitation and suggest that future iterations of the A11yBoard system need to add cross-device undo and redo.

Another limitation of A11yBoard for Google Slides is the lack of integration with Apple's VoiceOver mobile screen reader. This means that blind users who rely on VoiceOver will need to turn it off before using A11yBoard, which can add an additional layer of complexity to their workflow. For example, when they accidentally exit the App, it would be hard for blind users to realize that they have done so without screen reader announcements.

8 FUTURE WORK

This work opens up several areas of future research. One potential direction is exploring the use of advanced AI-generated content (AIGC) techniques, like large language models (LLMs), to automatically generate slides. This could reduce the cognitive load in creating accessible visual content, but careful curation is necessary to meet the needs of blind users. Balancing automation with user control over visual content is another aspect to investigate, along with enabling accessible fine-tuning processes to personalize the generated content.

Another area to explore is enhancing collaborative slide editing accessibility. With A11yBoard's improvements in slide reading and authoring, extending these benefits to collaborative editing becomes essential. Possible approaches include using AIGC techniques to generate alternative descriptions for visual content, aiding team members using different assistive technologies, or developing new collaboration features that support real-time collaboration with various assistive technologies. Whichever approach is chosen, it should prioritize accessibility and intuitiveness for all team members, regardless of their abilities.

9 CONCLUSION

In this work, we have presented A11yBoard for Google Slides, a multi-device multimodal system deployable in real-world scenarios to make Google Slides, a commerical presentation tool, accessible to blind users. We described the A11yBoard system consisting of a Chrome browser extension and an Apple iOS mobile app. We also described the participatory design process we followed with a blind co-author that led to A11yBoard's interaction design. A11yBoard for Google Slides addresses the key challenges of designing an accessible experience of reading and editing slide decks for blind users. To put A11yBoard for Google Slides through its paces, we deployed it in two case studies that showed our blind participants were capable of reading and creating slides independently and efficiently, without the need for sighted assistance, something that had been previously impossible for them. We also offered design recommendations for the further development of accessible content creation tools. In the end, it is our hope that A11yBoard for Google Slides provides a significant step towards making 2-D presentation tools more inclusive and accessible for all users.

ACKNOWLEDGMENTS

This work was supported in part by the University of Washington Center for Research and Education on Accessible Technology and Experiences (CREATE). Any opinions, findings, conclusions or recommendations expressed in our work are those of the authors and do not necessarily reflect those of any supporter.

REFERENCES

[1] Apple. 2022-09-09. Accessibility - Vision. https://www.apple.com/accessibility/vision/

[2] Amine Awada, Youssef Bou Issa, Clara Ghannam, Joe Tekli, and Richard Chbeir. 2012. Towards digital image accessibility for blind users via vibrating touch screen: A feasibility test protocol. In *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*. IEEE, 547–554.

[3] Amine Awada, Youssef Bou Issa, Joe Tekli, and Richard Chbeir. 2013. Evaluation of touch screen vibration accessibility for blind users. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*. Article 48, 2 pages.

[4] L. M. Brown, S. A. Brewster, S. A. Ramloll, R. Burton, and B. Riedel. 2003. Design guidelines for audio presentation of graphs and tables. https://eprints.gla.ac.uk/3196/

[5] Tim Brown et al. 2008. Design thinking. *Harvard business review* 86, 6 (2008), 84.

[6] Matt Calder, Robert F Cohen, Jessica Lanzoni, and Yun Xu. 2006. PLUMB: an interface for users who are blind to display, create, and modify graphs. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*. 263–264.

[7] Robert F Cohen, Arthur Meacham, and Joelle Skaff. 2006. Teaching graphs to visually impaired students using an active auditory interface. *ACM SIGCSE Bulletin* 38, 1 (2006), 279–282.

[8] Franco Delogu, Massimiliano Palmiero, Stefano Federici, Catherine Plaisant, Haixia Zhao, and Olivetti Belardinelli. 2010. Non-visual exploration of geographic maps: does sonification help? *Disability and Rehabilitation: Assistive Technology* 5, 3 (2010), 164–174.

[9] Julie Ducasse, Anke M Brock, and Christophe Jouffrais. 2018. Accessible interactive maps for visually impaired users. In *Mobility of visually impaired people*. Springer, 537–584.

[10] Danyang Fan, Kate Glazko, and Sean Follmer. 2022. *Accessibility of Linked-Node Diagrams on Collaborative Whiteboards for Screen Reader Users: Challenges and Opportunities*. Springer International Publishing, Cham, 97–108. https://doi.org/10.1007/978-3-031-09297-8_6

[11] Nicholas A Giudice, Hari Prasath Palani, Eric Brenner, and Kevin M Kramer. 2012. Learning non-visual graphical information using a touch-based vibro-audio interface. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*. 103–110.

[12] David Goldberg and Cate Richardson. 1993. Touch-Typing with a Stylus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (Amsterdam, The Netherlands) *(CHI '93)*. Association for Computing Machinery, New York, NY, USA, 80–87. https://doi.org/10.1145/169059.169093

[13] Google. 2023-04-15. Edit presentations with a screen reader - Google Docs Editors Help. https://support.google.com/docs/answer/1634140?sjid=12624978275069237619-NA

[14] Google. 2023-04-15. Google Slides API. https://developers.google.com/slides/api/reference/rest

[15] Joan Greenbaum and Morten Kyng. 2020. *Design at work: Cooperative design of computer systems*. CRC Press.

[16] Tiago Guerreiro, Paulo Lagoa, Hugo Nicolau, Daniel Goncalves, and Joaquim A Jorge. 2009. From Tapping to Touching: Making Touch Screens Accessible to Blind Users (vol 15, pg 48, 2008). *IEEE MULTIMEDIA* 16, 1 (2009), 13–13.

[17] Mina Huh, Saelyne Yang, Yi-Hao Peng, Xiang 'Anthony' Chen, Young-Ho Kim, and Amy Pavel. 2023. AVscript: Accessible Video Editing with Audio-Visual Scripts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 796, 17 pages. https://doi.org/10.1145/3544548.3581494

[18] Deepak Jagdish, Rahul Sawhney, Mohit Gupta, and Shreyas Nangia. 2008. Sonic Grid: an auditory interface for the visually impaired to navigate GUI-based environments. In *Proceedings of the 13th international conference on Intelligent user interfaces*. 337–340.

[19] JAWS. 2020-10-15. Jaws® – freedom scientific. https://www.freedomscientific.com/products/software/jaws/

[20] Nikolaos Kaklanis, Konstantinos Votis, and Dimitrios Tzovaras. 2013. A mobile interactive maps application for a visually impaired audience. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*. Article 23, 2 pages.

[21] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. 2008. Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (Assets '08)*. Association for Computing Machinery,

New York, NY, USA, 73–80. https://doi.org/10.1145/1414471.1414487

[22] Shaun K. Kane, Meredith Ringel Morris, and Jacob O. Wobbrock. 2013. Touchplates: low-cost tactile overlays for visually impaired touch screen users. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. Association for Computing Machinery, New York, NY, USA, Article 22, 8 pages. https://doi.org/10.1145/2513383.2513442

[23] Martin Kurze. 1996. TDraw: a computer-based tactile drawing tool for blind people. In *Proceedings of the second annual ACM conference on Assistive technologies (Assets '96)*. Association for Computing Machinery, New York, NY, USA, 131–138. https://doi.org/10.1145/228347.228368

[24] Richard E Ladner. 2015. Design for user empowerment. *interactions* 22, 2 (2015), 24–29.

[25] Steven Landau and Lesley Wells. 2003. Merging tactile sensory input and audio data by means of the Talking Tactile Tablet. In *Proceedings of EuroHaptics*, Vol. 3. 414–418.

[26] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. Case Studies. (2017).

[27] Cheuk Yin Phipson Lee, Zhuohao Zhang, Jaylin Herskovitz, JooYoung Seo, and Anhong Guo. 2022. CollabAlly: Accessible Collaboration Awareness in Document Editing. In *CHI Conference on Human Factors in Computing Systems*. Article 596, 17 pages.

[28] Jaewook Lee, Jaylin Herskovitz, Yi-Hao Peng, and Anhong Guo. 2022. ImageExplorer: Multi-Layered Touch Exploration to Encourage Skepticism Towards Imperfect AI-Generated Image Captions. In *CHI Conference on Human Factors in Computing Systems*. Article 462, 15 pages.

[29] Franklin Mingzhe Li, Lotus Zhang, Maryam Bandukda, Abigale Stangl, Kristen Shinohara, Leah Findlater, and Patrick Carrington. 2023. Understanding Visual Arts Experiences of Blind People. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 60, 21 pages. https://doi.org/10.1145/3544548.3580941

[30] Microsoft. 2023-04-15. Accessibility tools for PowerPoint - Microsoft Support. https://support.microsoft.com/en-us/office/accessibility-tools-for-powerpoint-2b7a387c-bc02-408f-8c49-59534665850f

[31] Microsoft. 2023-04-15. Introducing Microsoft 365 Copilot – your copilot for work. https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/

[32] Joe Mullenbach, Craig Shultz, J Edward Colgate, and Anne Marie Piper. 2014. Exploring affective communication through variable-friction surface haptics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3963–3972.

[33] Joe Mullenbach, Craig Shultz, Anne Marie Piper, Michael Peshkin, and J Edward Colgate. 2013. Surface haptic interactions with a TPad tablet. In *Proceedings of the adjunct publication of the 26th annual ACM symposium on User interface software and technology*. 7–8.

[34] Jakob Nielsen. 1994. *Usability engineering*. Morgan Kaufmann.

[35] NVDA. 2020-10-15. Nv access. https://www.nvaccess.org/

[36] OpenAI. 2023-04-15. GPT-4. https://openai.com/product/gpt-4

[37] OpenAI. 2023-04-15. Introducing ChatGPT. https://openai.com/blog/chatgpt

[38] Hari Prasath Palani, G Bernard Giudice, and Nicholas A Giudice. 2018. Haptic information access using touchscreen devices: design guidelines for accurate perception of angular magnitude and line orientation. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 243–255.

[39] Yi-Hao Peng, Jeffrey P Bigham, and Amy Pavel. 2021. Slidecho: Flexible Non-Visual Exploration of Presentation Videos. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. Article 24, 12 pages.

[40] Yi-Hao Peng, Peggy Chi, Anjuli Kannan, Meredith Ringel Morris, and Ifran Essa. 2023. Slide Gestalt: Automatic Structure Extraction in Slide Decks for Non-Visual Access. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 829, 14 pages. https://doi.org/10.1145/3544548.3580921

[41] Yi-Hao Peng, Jason Wu, Jeffrey Bigham, and Amy Pavel. 2022. Diffscriber: Describing Visual Design Changes to Support Mixed-Ability Collaborative Presentation Authoring. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 35, 13 pages. https://doi.org/10.1145/3526113.3545637

[42] Beryl Plimmer, Andrew Crossan, Stephen A Brewster, and Rachel Blagojevic. 2008. Multimodal collaborative handwriting training for visually-impaired people. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 393–402.

[43] 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* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 270, 19 pages. https://doi.org/10.1145/3411764.3445242

[44] Ather Sharif and Babak Forouraghi. 2018. evoGraphs — A jQuery plugin to create web accessible graphs. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. 1–4. https://doi.org/10.1109/CCNC.2018.8319239 ISSN: 2331-9860.

[45] Ather Sharif, Olivia H Wang, Alida T Muongchan, Katharina Reinecke, and Jacob O Wobbrock. 2022. VoxLens: Making Online Data Visualizations Accessible with an Interactive JavaScript Plug-In. In *CHI Conference on Human Factors in Computing Systems*. Article 478, 19 pages.

[46] Ather Sharif, Andrew M Zhang, Anna Shih, Jacob O Wobbrock, and Katharina Reinecke. 2022. Understanding and improving information extraction from online geospatial data visualizations for screen-reader users *(Assets '22)*. Athens, Greece, Article 61, 5 pages.

[47] Katie A Siek, Gillian R Hayes, Mark W Newman, and John C Tang. 2014. Field deployments: Knowing from using in context. *Ways of Knowing in HCI* (2014), 119–142.

[48] Mathieu Simonnet, Anke M Brock, Antonio Serpa, Bernard Oriola, and Christophe Jouffrais. 2019. Comparing interaction techniques to help blind people explore maps on small tactile devices. *Multimodal Technologies and Interaction* 3, 2 (2019), 27.

[49] Jing Su, Alyssa Rosenzweig, Ashvin Goel, Eyal de Lara, and Khai N Truong. 2010. Timbremap: enabling the visually-impaired to use maps on touch-enabled devices. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*. 17–26.

[50] John Thompson, Jesse Martinez, Alper Sarikaya, Edward Cutrell, and Bongshin Lee. 2023. Chart Reader: Accessible Visualization Experiences Designed with Screen Reader Users. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 802, 18 pages. https://doi.org/10.1145/3544548.3581186

[51] Gregg C Vanderheiden. 1996. Use of audio-haptic interface techniques to allow nonvisual access to touchscreen appliances. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 40. SAGE Publications Sage CA: Los Angeles, CA, 1266–1266.

[52] Steven Wall and Stephen Brewster. 2006. Feeling what you hear: tactile feedback for navigation of audio graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. Association for Computing Machinery, New York, NY, USA, 1123–1132. https://doi.org/10.1145/1124772.1124941

[53] Lucy Lu Wang, Isabel Cachola, Jonathan Bragg, Evie Yu-Yen Cheng, Chelsea Haupt, Matt Latzke, Bailey Kuehl, Madeleine van Zuylen, Linda Wagner, and Daniel S Weld. 2021. Improving the accessibility of scientific documents: Current state, user needs, and a system solution to enhance scientific PDF accessibility for blind and low vision users. *arXiv preprint arXiv:2105.00076* (2021).

[54] Microsoft Windows. 2023-04-15. Complete guide to Narrator - Microsoft Support. https://support.microsoft.com/en-us/windows/complete-guide-to-narrator-e4397a0d-ef4f-b386-d8ae-c172f109bdb1

[55] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 159–168.

[56] Cheng Xu, Ali Israr, Ivan Poupyrev, Olivier Bau, and Chris Harrison. 2011. Tactile display for the visually impaired using TeslaTouch. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*. 317–322.

[57] Mingrui Ray Zhang, Mingyuan Zhong, and Jacob O. Wobbrock. 2022. Ga11y: An Automated GIF Annotation System for Visually Impaired Users. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 197, 16 pages. https://doi.org/10.1145/3491102.3502092

[58] Zhuohao (Jerry) Zhang and Jacob O. Wobbrock. 2022. A11yBoard: Using Multimodal Input and Output to Make Digital Artboards Accessible to Blind Users. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22 Adjunct)*. Association for Computing Machinery, New York, NY, USA, Article 9, 4 pages. https://doi.org/10.1145/3526114.3558695

[59] Zhuohao (Jerry) Zhang and Jacob O. Wobbrock. 2023. A11yBoard: Making Digital Artboards Accessible to Blind and Low-Vision Users. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 55, 17 pages. https://doi.org/10.1145/3544548.3580655