Improving the Accessibility of the EarSketch Web-Based Audio Application for Blind and Visually Impaired Learners

Stephen Garrett
School Of Music
Georgia Institute of Technology
garrett@gatech.edu

Jason Brent Smith
School Of Music
Georgia Institute of Technology
jsmith775@gatech.edu

Amber Blue
School Of Music
Georgia Institute of Technology
amber.blue.n@gmail.com

Zerrin Ondin
Center for Inclusive Design and Innovation
Georgia Institute of Technology
zerrin.ondin@design.gatech.edu

Johan Rempel
Center for Inclusive Design and Innovation
Georgia Institute of Technology
jrempel3@gatech.edu

Kara Mumma
Center for Inclusive Design and Innovation
Georgia Institute of Technology
kara.mumma@design.gatech.edu

Jason Freeman
School Of Music
Georgia Institute of Technology
jason.freeman@gatech.edu

Brian Magerko
Expressive Machinery Lab
Georgia Institute of Technology
magerko@gatech.edu

ABSTRACT

EarSketch is an online learning environment that uses the Web Audio API to teach computer science and computational thinking through music technology, production, and remixing. It is designed to broaden participation in computer science and music education for beginning learners by providing a free, web-based application coupled with curricula for informal and formal learning contexts and a library of audio content spanning multiple popular genres. This paper describes an analysis of EarSketch's accommodations for students who are blind or visually impaired (BVI) and initial improvements to the system to improve accessibility. This paper also presents the findings of a user study designed to determine the biggest accessibility issues for EarSketch users who are BVI, and a discussion of how accessibility improvements can help broaden participation in computing and music education and, more broadly, how all web audio application developers can integrate accessibility considerations into their work.

CCS Concepts

•Applied computing → Education; Interactive learning environments; Sound and music computing; •Social and professional topics → K-12 education;

Web Audio Conference WAC-2024, March 15–17, 2024, West Lafayette, IN, USA.

Keywords

STEAM education, creative computing, broadening participation, accessibility

1 Introduction

1.1 BVI in CS Education

Computer Science (CS) education has gained momentum in the past two decades as computing has become a ubiquitously needed skill in the 21st century. Developers and researchers who are blind or visually impaired (BVI), however, remain severely underrepresented in computing [26]. This is due to a variety of factors, including a lack of accessible environments for students who are BVI to learn computing concepts and write code. Although strides have been made towards creating more accessible programming environments and learning experiences for students who are BVI (e.g. [26, 2, 24, 8, 3, 13, 9, 7, 20]), there is still much work to be done in this space. Visual impairments are a relatively low-incidence disability, and as a result, support and resources for engaging students who are BVI in computing can be in short supply [1]. In particular, many authentic learning experiences that allow for creativity and personal expression have not been made accessible to students who are BVI as they take the form of online learning environments that were designed by sighted people and rely on visual design paradigms such as connecting visual puzzle pieces (or blocks) correctly, drag and drop manipulation using a mouse, and visual animations as the output artifact [27, 14].

Tools for improving the accessibility of code as well as creative and artistic tasks aim to accommodate students who are BVI in CS education, but face issues in areas of distribution and availability. Tools such as StructJumper

assist users who are BVI in navigating text-based code [2].

Other domain-specific languages for students have also been developed, such as the Quorum programming language which contains a curriculum with chapters on audio programming and other creative tasks [25].

Block-based languages (BBL) such as Scratch [19] provide a way for students to focus on the semantics of programming over the syntax of text-based code, and are well-suited to creative tasks such as Scratch's focus on animation and sound projects. Students with visual impairments can benefit from this approach, though the BBL infrastructure has not caught up with the need to provide full screen reader and keyboard support despite some research work in the area [13, 9].

Tangible or tactile programming interfaces have shown success in educational environments, but are heavily limited by the distribution of physical interfaces themselves [14, 7, 20].

These platforms that incorporate music making or audio programming with CS education tend to either require domain knowledge (e.g. music theory and composition) to produce something that students find artistically pleasing, or they have lower skill ceilings that constrain rich expressiveness when compared to environments available to sighted students. In addition, approaches using tangible programming or BBL are often perceived as being inauthentic by learners in high school computing courses, who are eager to learn what they perceive as being "real-world" languages that are used in professional practice [5, 4, 12].

1.2 EarSketch and Accessibility

EarSketch is an online learning environment in which learners use Python or JavaScript code to manipulate audio samples and create music [10, 11]. It contains a curriculum with lessons about introductory coding and music concepts as well as other facets of code and music production such as code review and collaboration. It has shown success across multiple learning contexts and student populations in driving student engagement and intention to persist in computing [28, 10] through an approach that requires no prerequisite computing or music experience, is perceived by students to be authentic in the computational and musical domains, and supports students' personal expression [4, 12]. EarSketch uses the Web Audio API for the real-time manipulation and playback of sounds in its Digital Audio Workstation (DAW) view (see Figure 1) and generating sound recommendations [11, 23].

Unlike many other creative computing environments used in education, such as Scratch [19] and Processing [18], EarSketch focuses exclusively on the creation of audio rather than visual content, which makes for a compelling use case for students who are BVI. However, prior to the work described in this paper, EarSketch was almost completely inaccessible to individuals using assistive technologies such as screen readers.

Payne [17] has identified a need for musicians who are BVI to have multimodal interactions, customizable interfaces, inclusive design approaches, and communication between developers and users to make music technology more accessible. Payne has used this set of user needs to evaluate how musicians who are BVI access music notation [15] as well as the successful integration of collaborative live coding in a classroom environment including students of varying visual acuities [16]. As developers of web-based software that uses music in education, we also consider these priorities when evaluating the accessibility of EarSketch.

In this paper, we examine the accessibility of EarSketch as a tool for learners who are BVI and its alignment with modern accessibility standards[1] and move toward a design that puts users who are BVI at the center.

The contributions of this paper are:

- An accessibility audit of an online, web audio-based music and coding application and an overview of the remediation made to improve accessibility for users who are BVI.
- The design and execution of a user study highlighting issues users who are BVI face when using a web audio-based production environment.
- A discussion of the findings from this study and how they are informing the continued development of EarSketch, along with considerations for how these findings can inform the work of other web audio application developers.

2 Preliminary Work

2.1 Accessibility Audit

The EarSketch team partnered with the Center for Inclusive Design and Innovation (CIDI) at Georgia Tech to conduct an audit of the EarSketch web application from an accessibility perspective. CIDI is a research and service center that provides expertise, tools, and technology for accessibility for education, corporations, non-profits, and government institutions throughout the United States. CIDI provided detailed reports of areas of inaccessibility and made recommendations for improving the EarSketch interface (see Table 1 for an excerpt of the report). The audit and evaluation reports highlighted 38 key accessibility issues in EarSketch, with each issue having one or more occurrences in the user interface. Each issue in the report included the following:

- Corresponding Web Content Accessibility Guidelines (WCAG) [2]
- Level of impact on preventing the user from being able to access the feature or content
- The WCAG guideline level assigned by the W3C (A, AA, AAA)
- Corresponding rendered HTML source(s) where the issue occurred
- Recommendations for how to address the issue in a way that will meet WCAG guidelines and provide access

The EarSketch team supplemented these recommendations with additional audit findings from the automated Microsoft Accessibility Insights[3] and Axe[4] tools.

[1] https://www.w3.org/TR/WCAG21/
[2] https://www.w3.org/TR/WCAG21/#cc1
[3] https://accessibilityinsights.io/
[4] https://www.deque.com/axe/

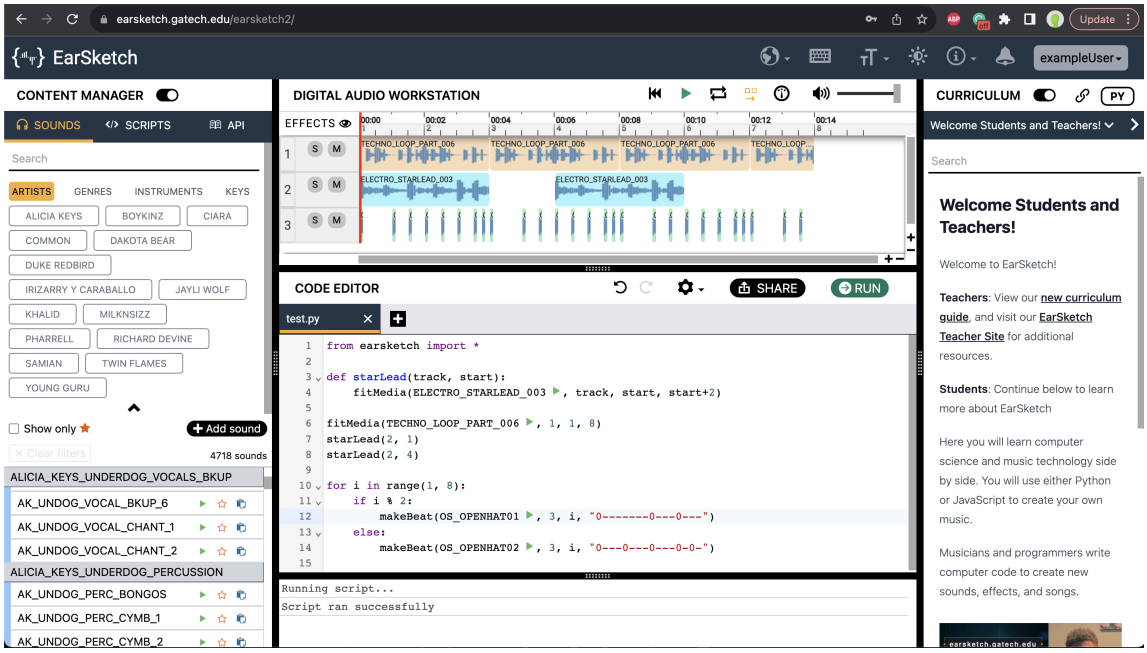

Figure 1: The EarSketch interface, with Content Browser (left), Digital Audio Workstation (top), Code Editor (bottom), Curriculum (right).

2.2 Remediation

Through a series of bi-weekly meetings with the CIDI team, the EarSketch development team iteratively addressed the audit findings by presenting development versions of EarSketch addressing specific accessibility issues to gather feedback from CIDI. The approach was to directly mirror the hierarchy of WCAG guidelines, prioritizing updates according to the WCAG accessibility Levels. This strategy ensured systematic enhancements to the accessibility features of the EarSketch platform, starting with the most critical improvements and progressing to more advanced requirements. This focus was particularly beneficial for users relying on screen readers, high-contrast settings, and magnification tools. These included the following accessibility improvements:

- Enhanced screen reader accessibility for users who are BVI by replacing non-descriptive button names (particularly for buttons only containing icons and no text) with meaningful labels conveying their functionality and intended action. Buttons with only icons can simplify visual/graphical user interfaces but must be described by an accessible name by using an `aria-label`, `aria-labelledby`, or `title` attribute to be accessible to screen readers.
- Improved form navigation for screen readers by adding explicit labels to each input field, providing clear expectations for each input. Without explicit labels (either through enclosing `<label>` tags or Accessible Rich Internet Applications (ARIA) ”label” or ”labeled by” attributes, forms can be difficult to use with screen readers as the input field won’t have a descriptive name. ARIA labels provide explicit descriptions rather than relying on the visual proximity of descriptive text to a form input field.
- Achieved WCAG 2.3 AAA standards for color contrast, aiding users in differentiating text and interactive elements. Low contrast between background and foreground colors (e.g. white text on a light background), or thin/small font sizes can make text or other elements difficult to see for users with low vision. Increasing color contrast makes the interface legible and accessible to more people, including those who may not consider themselves BVI. Figure 2 shows a set of examples of these changes.
- Enhanced navigation and identification of interactive elements by converting instances of clickable `<div>` and `<span>` tags to semantically correct `<button>` tags. Making clickable `<div>` or `<span>` accessible involves marking them up with proper role and cursor attributes so that they are read properly. This can become a long term maintenance issue, and these attributes come for ”free” by just using a `<button>` tag. In our case, `<button>` tags were sometimes avoided to conveniently avoid styling from legacy opinionated CSS frameworks. If an element behaves like a button, the default choice should be a `<button>` tag.
- Adding descriptive ARIA attributes for labels, toggle button states, and menu hierarchy to improve the descriptions, predictability, and contextual understanding of each element. These convey information to the screen reader that is otherwise only available visually, such as whether a toggle button is pressed or not, how many tabs are in a tabbed interface element, and which tab is currently selected, or how many options are available in a multi-select menu and how many are selected.
- Enhanced keyboard-only navigation by improving tab focus ordering, proper header hierarchy (e.g. `h1`, `h2`,

Issue, WCAG Guideline, Impact	Source Code & Recommendation

Zoom in/out `<button>` elements are missing accessible names.
 	
4.1.2 Name, Role, Value
 	
Level of Impact: High
 	
WCAG Level: A
	Zoom In Buttons
	`<button class="zoom-in"><i class="icon-plus2 ">...`
	Zoom Out Buttons
	`<button class="zoom-out"><i class="icon-minus">...`
	
	Explanation: The zoom in/out buttons on the daw-container are missing accessible names/labels. Typically, buttons that use icons instead of text as visual labels should also contain programmatically associated names/labels. Without them, the button's purpose or function may be lost to screen reader users who rely on the screen readers access to the attributes of the button. To put it in perspective, the zoom in button should be read as "zoom in button" via screen reader, but instead it is read as "button". To a user who may be blind, this is not enough information to determine the function of the button.
	Recommendation: Provide an accessible name/label for the zoom in/out button elements. Acceptable methods include providing a `<label>` element or an `aria-label`, `aria-labelledby`, or `title` attribute.
	Supporting Article(s): https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles/button_role

Heading markup should be improved. 1.3.1: Info and Relationships Level of Impact: Medium WCAG Level: A	`<h2 id="welcome">Welcome Students and Teachers!</h2>` **Explanation:** The heading that reads "Welcome Students and Teachers" is marked up as an `<h2>`, but it is not preceded by an `<h1>`. For a proper page structure, headings should follow a logical order (`h1` > `h2`...`h5`). Headings should also be used for their semantic meaning as opposed to their aesthetic value. **Recommendation:** Provide proper heading markup for the "Welcome Students and Teachers" heading in the Curriculum section of the page. **Supporting Article(s):** https://www.w3.org/TR/WCAG20-TECHS/H42.html

Website lacks visible focus indicators. Default Visible Focus Disabled on `<button>` Elements
 	
2.4.7: Focus Visible
 	
Level of Impact: Critical
 	
WCAG Level: AA
	`button:focus {`
	`outline: none !important;`
	`}`
	
	Explanation: Throughout the page, there is a lack of visible focus. In some areas, elements that can receive keyboard focus like buttons, links, and input there is no indication when focus has landed on the element. In other cases, the indication of visible focus is a small, almost unnoticeable, change in the color of the element. The lack of visible focus indicators throughout the page makes it very difficult for users with low vision and keyboard-only users to track where they are on the page and could cause some of these users to abandon the site all together.
	Recommendation: Provide adequate visible focus indicators that can be seen without much difficulty. Also, remove any styles that disable the default visible focus of focusable elements.
	Supporting Article(s): https://www.w3.org/WAI/WCAG21/Techniques/failures/F78.html

Table 1: Examples from the EarSketch accessibility audit

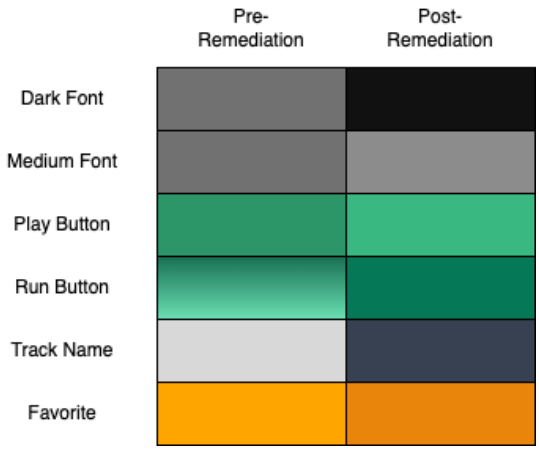

Figure 2: Colors of various EarSketch interface elements, before (left) and after (right) remediation as a result of the accessibility audit.

etc), implementing a quick navigation links menu, allowing the user to skip to the desired header or section of the page, and implementing semantic and ARIA landmarks (e.g. banner, nav, main). These navigation features make the application more usable and accessible to people who cannot use a mouse by allowing users to quickly hear a list of landmarks or headings by triggering a keyboard shortcut. This allows for quick navigation throughout the application rather than having to tab through each element.

- Migrating our code editor to CodeMirror 6, which included substantial accessibility enhancements, such as using the `contentEditable` attribute for direct content manipulation instead of a hidden `<textarea>`, and improved tab handling and enhanced keyboard-only navigation, aligning with WCAG guidelines to avoid keyboard traps[5].
- Internationalizing all of these ARIA labels, descriptions, and other attributes for future translation (EarSketch is currently available in 7 languages) so that screen readers can announce element names, descriptions, and state in users' preferred language

These improvements were iteratively deployed to the live, public-facing version of EarSketch. Once all audit findings were addressed, we began preparations for a user study to test the efficacy of the improvements with users who are blind or visually impaired.

3 Study Design

In Spring 2023, we conducted a preliminary user study to identify and determine how to address the pain points and barriers faced by EarSketch users who are BVI. This study was meant to understand how users who are BVI perform a set of basic tasks in the platform, to evaluate the effectiveness of our remediation efforts to-date and to guide design and development for future accessibility-related improvements and features.

3.1 Recruitment

Subjects were recruited from national, regional, and local (Atlanta, GA) organizations, networks, and disability service providers for the blind and visually impaired. Individuals were presented with our IRB-approved advertisement text which provided links to consent forms based on if they were interested in on-site or remote participation. When individuals signed the informed consent, we had them fill out the background survey, and inclusion was determined based on their survey responses, including amount of functional vision, additional disabilities, use of assistive technologies, and prior coding or EarSketch experience.

3.2 Participants

The EarSketch Accessibility research study consisted of 6 participants: 5 with little or no functional vision that access digital content using screen readers, and 1 with low vision accessing digital content using magnification. Individuals ranged in age from 18 – 49 years (Mage = 26.3 years) and were primarily male (83.3%). All have some college background: two have a 4-year degree, one has a 2-year degree, and three have some college experience (two of those are currently students at a university). There was a wide range of knowledge of coding – one was not knowledgeable at all, two were slightly knowledgeable, two were moderately knowledgeable, and one was extremely knowledgeable. None of the participants had prior experience with the EarSketch platform. In terms of Assistive Technology, three participants use the NVDA[6] screen reader, including one participant who is a certified expert with NVDA and has extensive experience with several screen readers. Two participants use JAWS[7], and one participant with low vision primarily relies on screen magnifiers. Three participants also said they use VoiceOver[8], the built-in screen reader on Apple's Mac and iOS devices.

3.3 Methodology

The study consisted of five benchmark tasks for the users to accomplish in EarSketch:

1) Exploring the EarSketch platform
2) Finding and playing a video in the Curriculum Browser
3) Locating and playing a sound in the Content Manager
4) Opening previously saved scripts in EarSketch
5) Running code and listening to it in the Digital Audio Workstation (DAW)

After each task, subjects were asked to rate the difficulty of the task on a five-point Likert scale and to discuss what changes to the platform would make completing the task easier.

To keep sessions within 1.5 hours, some steps were taken to expedite onboarding. The research team created usernames and passwords for each participant and

[5]https://www.w3.org/TR/WCAG21/#no-keyboard-trap

[6]https://www.nvaccess.org/

[7]https://www.freedomscientific.com/products/software/jaws/

[8]https://www.apple.com/accessibility/vision/

pre-loaded them with a working EarSketch script written in Python. This allowed users to complete tasks 4 and 5 regardless of their coding ability.

Participants were encouraged to think aloud as they were working through the session. The total time spent, observations, rating of task difficulty (on a scale of 1-5 where 1 is very easy and 5 is very difficult), and verbal feedback were collected. After obtaining permission from participants, all sessions were recorded. Overall, the users spent between 60 and 75 minutes to complete tasks outlined in the usability testing protocol.

4 Findings

As Table 2 shows, a majority of participants rated each task as very easy (1) or easy (2). Through their qualitative responses, subjects identified concerns in categories of labeling issues, the use of keyboard shortcuts, live screen reader alerts, and non-standard HTML controls.

Participant	Task 1	Task 2	Task 3	Task 4	Task 5
1	1	2	1	2	1
2	2	2	2	2	5
3	2	3	1	4	1
4	2	3	2	1	1
5	2	1	3	-	-
6	1	1	1.5	1	2
Average	1.67	2	1.75	2	2

Table 2: User Ratings of Task Difficulty

4.1 Labeling

Our descriptive text in the interface or our ARIA labels still shows some sighted bias as well as a need for more precision. Still relying on sighted representations, like shapes or vertical positioning in the text of our ARIA labels, "click the right arrow above to get started" rather, one participant suggested, "Make it so you're not relying on shapes to exclusively signify a meaning. You have to actually say what that meaning is. Seeing "next page" will help to know what a person is looking for." In another example, our play button to preview sounds in the sound browser switches from a play icon (green triangle) to a stop button (black square) while playing, but this was the only representation of the state change. Including dynamic ARIA descriptions on the play button based on state would be helpful. Line numbers in the code editor were not being read properly, which made understanding location within the Python script more challenging.

4.2 Keyboard Shortcuts

We were initially concerned that continuing to add keyboard shortcuts might add cognitive load and become overwhelming, but we received suggestions for more keyboard shortcut keys, especially for the DAW transport controls such as "rewind, fast forward, beginning, end, and may want to jump by measure, set and move through markers, jump by 30 sec, or other portions of time since there will be interference with screen reader output." This participant stated that they would mute JAWS and then use shortcut keys. A person "should be able to play it from the code editor, as doing it with a keyboard is so much more efficient." Another suggestion included adding a search bar at the top so a screen reader user could more easily navigate the interface and shortcuts, similar to the "command palette" design pattern implemented in Visual Studio Code and other apps.

4.3 Live Alerts

We were also overly cautious with the amount of information we presented as live screen reader alerts, and so only included basic notifications of whether a user's script ran successfully or not. One participant "really appreciate(s) announcements whenever there are updates. Their NVDA reads them," and that "so many companies don't do that." However, details of errors in the code were left out for brevity, but one participant noted "the program did indicate that there were one or more errors, but it would be helpful to notify what the errors might be." The participant "couldn't fix it because I didn't know where the error was," and another participant stated that the error alert should just say what line it was on, and that it would be helpful if the error was highlighted or explained.

4.4 Non-standard HTML Controls

Filtering our sound library by artist proved more difficult than needed due to the use of non-standard controls for drop-down lists of artists, genres, and instruments. One participant said "the list box was weird to open (had to press the enter key) and that usually, you can arrow through options in combo boxes, but had to tab into it and do some finagling to choose [the artist] "Common."" Similarly, our use of `<div>` elements to represent the list of sounds proved less navigable than if we had used a standard `<table>` element. Each row in our sound browser contains the name of the sound, a play button to preview the sound, a star button to save the sound to the user's favorites, and a paste button to paste the name of the sound (i.e. the Sound Constant) into the code editor. Use of nonstandard controls can be common in web applications to achieve certain visual aesthetics or override a web browser's default styling of components such as drop-down lists, however, this can obfuscate the semantics of the information for users who are blind or visually impaired.

One participant provided additional perspective, saying "Creating a table can reduce the cognitive load for users; otherwise, they have to remember to arrow three times each time [to move past the three buttons]. It will allow them to determine the end of it and move beyond it and filter out specific information by navigating quickly to what they're interested in and/or quickly eliminating what not interested in so can skip over as well as give a sense of the size of that data," and "As a sighted user, think about the information you want to know/take for granted. There's a scroll bar for sighted viewers – these users want to know that too. Table would give a sense of size, set, and position of set (ex. item 2 of 5)."

There were also other cases where we were attempting to conform to accessibility / ARIA best practices, but the implementation still left room for improvement because of our lack of expertise in navigating websites via keyboard. For example, there is a group of tab buttons at the top of the Content Manager for switching between Sounds, Scripts, and the API browser. These were properly attributed with

the ARIA role of "tab" within a "tab list" but were not properly contained in a "tab panel." One participant stated they "shouldn't have to use the tab key to move from one page tab to another. I had to tab, hit the tab panel, and use arrow keys to navigate, and then tab to get out of it." A tab panel would have added functionality to switch tabs with the keyboard arrow keys once focus landed inside of the tab panel.

Even after manual and automated audits, participants still managed to find instances where non-interactive elements, such as `<div>` elements, had 'on click' behaviors present that weren't caught ahead of the study. The practical implications were that participants found it difficult or confusing to try to open a script from the script browser. One participant noted that they were trying to find an "open" button and "I didn't think there was one. It should have one tab stop and use arrow keys." The participant thought they'd land on a new script and tried to press enter but this didn't work. One participant stated that the Python script filename "sounds like text (JAWS isn't calling it a link), so not hearing anything telling that a person can click on this. Might be able to click on it with a mouse, but with a screen reader a) nothing tells you to click on it and b) would need to try to use the mouse to click on it."

These seemingly small differences from the perspective of a sighted software developer (using a `<div>` instead of a `<button>`) can make our interfaces confusing at best, and unusable at worst to people who use assistive technologies.

4.5 Additional Feedback

Our curriculum video player could use improvement with better button controls as well as audio descriptions for the video content. For the participants with low vision using screen magnification, they commented that the addition of full-screen functionality for the DAW as well as the code editor would be helpful. They also commented that the red playhead line which shows in the DAW timeline was helpful, but a second black/gray line that shows the current mouse position was confusing because the user had to determine which line was presenting the information they wanted. Using a different shape or arrow to indicate the current mouse position on the timeline (for selecting loop regions) was suggested.

While our remediation work for EarSketch made the application conform to the letter of accessibility standards and, on average, users rated the study tasks between easy and very easy, there were still many design elements that were less than ideal from an accessibility standpoint and could affect learners' cognitive load. Many of these issues stem from the fact that EarSketch was originally designed only with sighted users in mind. These findings motivate our future work described below.

5 Discussion & Future Work

Ultimately, this work only seeks to make the existing EarSketch user interface more accessible but does not address major design questions inherent in an application designed by sighted software developers for sighted teachers and students. We now seek to redesign EarSketch through a participatory design process that engages students who are blind or visually impaired, their teachers and parents, and expert music producers and software engineers who are BVI.

We will approach this task from a universal design perspective, with the goal being that both sighted students and students who are BVI will use this new version of EarSketch. We will use the Universal Design for Learning (UDL) [21] principles of offering multiple means of representation, expression, and engagement [22, 6] to create a learning environment that is accessible to the widest variety of students with differing visual abilities.

We hypothesize that music programming learning environments, like EarSketch, can be re-designed for and with students who are BVI to educate learners who are BVI in computer science and improve their attitudes towards computing as a discipline. More specifically, we hypothesize that this new version of EarSketch — designed with and for students who are BVI — will lead to gains similar to those sighted students have enjoyed in prior versions of EarSketch: content knowledge gains in computing and music, interest formation, belongingness, and intent to persist in computing. In addition, we hypothesize that the code complexity, positive attitudinal changes, and evidence of engagement we see amongst students who are BVI will be comparable to changes we see in data from current and former sighted student users of EarSketch.

We have recently begun this participatory design work in collaboration with research teams at Northwestern University and the University of North Texas. To date we have prepared observation and interview protocols for teachers and softare development and music professionals and received IRB approval. We have also visited a partner school, California School for the Blind, in-person to observe coding activities with students who are blind and conducted followup interviews with teachers who conducted these activities.

Unless accessibility for blind and visually impaired users is considered from the outset of application design and development, a bias towards sighted users is likely to make applications difficult or impossible for users who are BVI to effectively utilize. Because of their focus on audio rather than visual forms of output and their easy browser-based, cross-device access, web audio applications present a unique opportunity for users who are BVI. By sharing our ongoing work on EarSketch with the web audio community, our hope is to highlight challenges and opportunities for the accessibility of web audio applications and encourage all web audio developers to integrate universal design, participatory design, and usability testing into their workflows to ensure that their applications not only address the minimum technical standards for accessibility but also incorporate designs that support effective real-world use.

6 Acknowledgments

Preliminary work, study, and findings supported by the GT-Microsoft Accessibility Research Seed Grant Program.

This material is based upon work supported by the National Science Foundation Award No. 2300631. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. EarSketch is available online at https://earsketch.gatech.edu

7 References

[1] J.-H. Ahn, W. Sung, S. W. Lee, J. Hee, et al. Cobrix: A physical computing interface for blind and visually impaired students to learn programming. In *Society for Information Technology & Teacher Education international conference*, pages 727–733. Association for the Advancement of Computing in Education (AACE), 2017.

[2] C. M. Baker, L. R. Milne, and R. E. Ladner. Structjumper: A tool to help blind programmers navigate and understand the structure of code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3043–3052, 2015.

[3] L. B. Caraco, S. Deibel, Y. Ma, and L. R. Milne. Making the blockly library accessible via touchscreen. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, pages 648–650, 2019.

[4] S. Engelman, B. Magerko, T. McKlin, M. Miller, D. Edwards, and J. Freeman. Creativity in authentic steam education with earsketch. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 183–188, 2017.

[5] M. Guzdial and A. E. Tew. Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, page 51–58, New York, NY, USA, 2006. Association for Computing Machinery.

[6] K. E. Koehler and T. A. Wild. Students with visual impairments' access and participation in the science curriculum: Views of teachers of students with visual impairments. 22(1):8.

[7] V. Koushik, D. Guinness, and S. K. Kane. Storyblocks: A tangible programming game to create accessible audio stories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[8] S. Ludi, M. Abadi, Y. Fujiki, P. Sankaran, and S. Herzberg. Jbrick: accessible lego mindstorm programming tool for users who are visually impaired. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*, pages 271–272, 2010.

[9] S. Ludi and M. Spencer. Design considerations to increase block-based language accessibility for blind programmers via blockly. *J. Vis. Lang. Sentient Syst.*, 3(1):119–124, 2017.

[10] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown. Earsketch: A steam-based approach for underrepresented populations in high school computer science education. *ACM Transactions on Computing Education (TOCE)*, 16(4):1–25, 2016.

[11] A. Mahadevan, J. Freeman, B. Magerko, and J. C. Martinez. Earsketch: Teaching computational music remixing in an online web audio based learning environment. In *Web Audio Conference*. Citeseer, 2015.

[12] T. McKlin, B. Magerko, T. Lee, D. Wanzer, D. Edwards, and J. Freeman. Authenticity and personal creativity: How earsketch affects student persistence. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 987–992, 2018.

[13] L. R. Milne and R. E. Ladner. Position: Accessible block-based programming: Why and how. In *2019 IEEE Blocks and Beyond Workshop (B&B)*, pages 19–22. IEEE, 2019.

[14] C. Morrison, N. Villar, A. Thieme, Z. Ashktorab, E. Taysom, O. Salandin, D. Cletheroe, G. Saul, A. F. Blackwell, D. Edge, et al. Torino: A tangible programming language inclusive of children with visual disabilities. *Human–Computer Interaction*, 35(3):191–239, 2020.

[15] W. Payne and A. Hurst. "we avoid pdfs": Improving notation access for blind and visually impaired musicians. In *International Conference on Information*, pages 581–597. Springer, 2023.

[16] W. C. Payne, X. Shen, E. Xu, M. Kaney, M. Graves, M. Herrera, M. Mau, D. Murray, V. Wang, and A. Hurst. Approaches to making live code accessible in a mixed-vision music ensemble. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–5, 2023.

[17] W. C. Payne, A. Y. Xu, F. Ahmed, L. Ye, and A. Hurst. How blind and visually impaired composers, producers, and songwriters leverage and adapt music technology. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–12, 2020.

[18] C. Reas and B. Fry. *Processing: a programming handbook for visual designers and artists*. Mit Press, 2007.

[19] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[20] Z. Rong, N. F. Chan, T. Chen, and K. Zhu. Coderhythm: Designing inclusive tangible programming blocks. In *Companion Publication of the 2020 ACM Designing Interactive Systems Conference*, pages 105–110, 2020.

[21] D. Rose. Universal design for learning. *Journal of Special Education Technology*, 15(4):47–51, 2000.

[22] D. H. Rose, W. S. Harbour, C. S. Johnston, S. G. Daley, and L. Abarbanell. Universal design for learning in postsecondary education: Reflections on principles and their application. *Journal of postsecondary education and disability*, 19(2):135–151, 2006.

[23] J. Smith, M. Jacob, J. Freeman, B. Magerko, and T. Mcklin. Combining collaborative and content filtering in a recommendation system for a web-based daw. In *Proceedings of the International Web Audio Conference*, 2019.

[24] A. Stefik, A. Haywood, S. Mansoor, B. Dunda, and D. Garcia. Sodbeans. In *2009 IEEE 17th International Conference on Program Comprehension*, pages 293–294. IEEE, 2009.

[25] A. Stefik and R. Ladner. The quorum programming

language. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 641–641, 2017.

[26] A. M. Stefik, C. Hundhausen, and D. Smith. On the design of an educational infrastructure for the blind and visually impaired in computer science. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 571–576, 2011.

[27] A. van der Meulen, M. Hartendorp, W. Voorn, and F. Hermans. The perception of teachers on usability and accessibility of programming materials for children with visual impairments. *ACM Trans. Comput. Educ.*, 23(1), dec 2022.

[28] D. L. Wanzer, T. McKlin, J. Freeman, B. Magerko, and T. Lee. Promoting intentions to persist in computing: an examination of six years of the earsketch program. *Computer Science Education*, 30(4):394–419, 2020.