Accessible Computer Science for K-12 Students with Disabilities.

by

Meenakshi Das

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 1, 2021

Keywords: Accessibility, K-12 Education, Block-Based Programming

Abstract

Disability Inclusion and Accessibility in technology and education have today emerged as a necessity of national importance. National Science Foundation(NSF) funded projects like AccessCSforALL, Bootstrap, and CSforAll, are making efforts to make computer science inclusive to the 7.4 million K-12 students with disabilities. Bringing people with disabilities into tech workforce would not only create equal opportunities for all but would also better equip the society to serve its diverse population. Computer Programming is a growing and important field of the present and future. When the tools to learn programming from an early age are inaccessible to people with disabilities, they are denied the opportunity to succeed. This thesis focuses on making computer science concepts accessible to K-12 students who are Deaf/Hard of Hearing, Blind/Low Vision and those with motor disabilities. It attempts to provide technical strategies to break this inaccessibility barrier and make computer science literacy, education, and, ultimately, careers accessible to all.

Acknowledgments

I would first like to thank my advisor, mentor and committee member Dr. Daniela Marghitu whose guidance and expertise was invaluable to me during the entirety of this research. I am grateful for her unwavering support, kindness, and encouragement throughout my master's program. I also profusely thank my committee members Dr. Xiao Qin and Dr. Bo Liu for their advice and support as I complete this academic milestone. My sincere thanks to Dr. Ayanna Howard and Dr. Mahender Mandala from the Georgia Institute of Technology for their collaborations on parts of this research. Last, I thank my parents, Manas and Sunanda Das, for their emotional support in my master's journey.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Computer science is one the most popular skills to have in today's world. According to the US Bureau of Labor Statistics, the employment of computer and information research scientists is projected to grow 15 percent from 2019 to 2029, much faster than the average for other occupations [1]. However, individuals with disabilities are less likely to pursue pursue science, technology, engineering, and mathematics degrees [2]. They are at a disadvantage mainly due to lack of variety of accessible instruction. Work must be done to make this instruction, and eventually technology careers accessible to all.

1.2 Research Justification

Inaccessible computing instruction can range from something that can be easily remediated to solutions that need technology infrastructural changes. These are described below:

1. Curriculum presented in an inaccessible word or PDF document: By using existing tools such as Microsoft's Accessibility checker, many common accessibility issues such as missing alternate text, insufficient contrast and structural headings etc can be remediated. One of the most important characteristics of an accessible PDF is that it should be 'tagged'. PDF tags provide a hidden, structured representation of the PDF content that is presented to screen readers [3]. Adobe Acrobat's Accessibility feature provides autotag amongst several other accessibility features. Due to COVID-19 pandemic, much instruction has shifted to recorded lectures. In this case, captions are crucial for the Deaf/Hard

of Hearing(D/HH) audience. If these documents are made accessible and videos are captioned from the start, many classroom accommodation requests for document accessibility can be prevented. This is the concept of Universal Design of Learning, which aims to make products or services accessible from the outset, and has emerged as a popular framework for creating accessible instruction methods [4].

2. Inaccessible or partially Accessible Integrated Development Environment's(IDEs): These are programming input software's that developers use the write and test code. The inaccessibility of these software makes it difficult for people with disabilities, especially those who are Blind/Low Vision(BLV), who rely on screen-readers, to learn to code. In a study conducted on blind developers, it was reported that some blind participants used alternatives to mainstream IDE's such as NotePad ++ due to latter being more accessible [5]. At the same time, it was found that some participants did use IDE's for some tasks which suggests that the IDE's are partially accessible [5]. Participants further mentioned that features such as debugging tools present accessibility barriers [5]. In addition to screen-reader inaccessibility, access issues for individuals with motor disabilities must also be considered. They may use one or combinations of Speech to Text(STT), eye-tracking and switch control as input alternatives but research has shown that these are often slower than mouse based input [6] [7]. Another programming environment to consider is Block-Based Programming, which was first developed by Google Blockly. Applications such as MIT's Scratch use Google Blockly as their base and are commonly used to teach computer programming to K-12 Students, where students drag and drop coding blocks to construct computer programs. However, these environments are not screen-reader accessible, although research has been done to make them accessible [8] [9].

3. Inaccessible Output: In many K-12 programming activities which use Block-Based Programming, graphical outputs to show results of program are very popular. However, most of them are inaccessible to people to screen-readers. For example, in Code.Org's Hour of Code, which are tutorials designed to expose K-12 students to coding and other aspects of computer science, there are many activities such as Dance Party, Star Wars, Flappy Code

etc where users use code blocks to manipulate 2D characters[10] [11]. Currently only two Hour of Code activities, namely Astronomy and Code with Mary which are accessible via screen-readers [10]. These use the accessible text-based Quorum Language [10]. According to researchers Stefik and Ladner, ”The key problem is to replace an output that cannot be perceived, to an output that can be perceived so that the information obtained is as equivalent as possible” [6]. Stefik and Ladner further go on to mention that next vital steps would be to identify and prioritize working on the visual components of learning CS that do not have good alternative non-visual access [6]. An example of non-visual output is used in Blockly Games, which has a game on Music composition. Here, users are asked compose notes using different coding blocks. Music is a very accessible output for individuals who are BLV. For those who are Deaf/Hard of Hearing(D/HH), video output with captions might be more engaging.

4. Inaccessible Infrastructure: Stefik says that Online curricula such as the Code.org AP CSP curricula use a multitude of tools such as network, data compression , and encryption simulators for creating graphical visual outputs [6]. Hence, making accessible versions of these can be very difficult. Although Blockly has recently added Keyboard Navigation to its blocks which presents capability to make programming input screen-reader accessible, changes on the operating system(OS) level are still needed so that graphical outputs of code can be easily connected to accessibility technologies [6]. Gaming applications which also heavily utilize graphics face similar issues as it quite challenging to deal with OS level accessibility APIs [12]. Stefik further goes on to say, ”Accessibility APIs should be built into the core graphics toolkits, like OpenGL, Direct3D, Metal, and otherwise. Graphics developers today have a herculean task to make their applications accessible, but this need not be so with improvements to accessibility infrastructures at the operating system and programming language level” [6]. This demonstrates the need for advocating to make the lowest level of the software stack accessible.

5. Limited Teacher’s Knowledge: Israel et.al say that many teachers are new to teaching computer science(CS), since most computer science majors prefer industry jobs after

graduation [13]. Hence they are considered twice new: New to CS education and new to inclusive subject-specific practices within CS Education [14]. They also go on to say, ”At the same time, we must be sensitive to the fact that teachers are overburdened, and professional learning experiences around inclusion need to be carefully planned and integrated within wider school considerations” [14]. A study exploring the perspectives of Teachers of the Visually Impaired(TVI) Regarding Accessible K12 Computing Education reaffirms this finding. In the study it was found that the TVI’s biggest challenge was ”lack of accessible learning materials which then falls on the teachers to produce as part of providing usable and accessible computing education to the class” [15]. Hence, this significantly increases the TVI’s workload. Last, sometimes implicit biases and stereotypes regarding students with disabilities may create lower expectations for students. Although IEPs create specialized learning opportunities for K-12 Students with disabilities, that should not let teachers, parents or administrators lower the bar for them [13].

This research works to explore and remedy some of the issues presented above.

Chapter 2

Literature Review

This chapter narrows its focus to discuss the current approaches to make K-12 computing education tools accessible to those who are Blind/Low Vision(BLV), Deaf/Hard of Hearing(D/HH) and/or have motor disabilities since many of accessibility approaches presented for the BLV, D/HH and motor disabilities population indirectly remediate accessibility barriers for other disabled population. For example, providing captions on curriculum videos helps individuals with auditory processing disorders.

2.1 Accessibility of Computer Science Instruction: Current Approaches

2.1.1 Blind/Low Vision(BLV) Population

Quorum is one of the most popular programming languages used to teach computer programming to BLV K-12 Students [16]. It is a ”born accessible” [16] and was designed with accessibility in mind from its inception phase. The Quorum language supports a large number of applications, including LEGO robotics programming, digital signal processing, 2D/3D graphical, physics simulations, 3D positional audio or other programs [16]. Since Quorum was designed with accessible design from the start, it has capability to effectively connect to various accessibility systems and hence support complex graphical outputs [16]. Since Quorum is also an evidence based language, it removes syntactic clutter and hence makes code easier to understand when using accessibility technologies such as screen readers [16] [17].

Syntactic clutter is one of the disadvantages of text-based programming languages. Due to this, block-based programming languages have gained prominence in teaching K-12 Students to code. Students use blocks of code to create programs, thereby eliminating the need to type code. Research has also shown that the blocks-based modality supports learners' conceptual understanding, helps novices advance more quickly, and engages underrepresented populations with computer science [18]. However, as mentioned, block-based language applications are not accessible to BLV population but progress is being made to making them accessible. Google has released Keyboard Navigation for Blockly which is one step closer to accessibility for BLV individuals [19]. Ludi has implemented screen-reader accessibility for Blockly [8]. Milne and Ladner created Blocks4all, an iOS touch-based tablet application which is accessible via VoiceOver [9].

2.1.2 Deaf/Hard of Hearing(D/HH) Population

There is immense diversity in the D/HH Community. According to the National Association of the Deaf, "There are variations in how a person becomes deaf or hard of hearing, level of hearing, age of onset, educational background, communication methods, and cultural identity" [20]. D/HH individuals may use a variety of assistive technologies and/or accommodations such as cochlear implants, captions, sign language interpreters, lip-reading etc. There have been efforts to make resources accessible to the D/HH population in the educational community. The National Association of the Deaf (NAD) has an initiative called the Described and Captioned Media Program (CMP), which has developed a collection of more than 4,000 open-captioned educational video titles [21]. Efforts have been made specific to the computer science education field as well. Automatic speech recognition software are used in many classrooms which assist D/HH students with note-taking. At the Department of Computer Sciences at Villanova University, researchers developed the Villanova University Speech Transcriber (VUST) system using speech recognition and customizing computer science domain specific dictionary tool, to improve captioning capability for their computer science classes [22]. Captioning remains a crucial accommodation for the D/HH Population especially for the post-lingual D/HH individuals(those who became deaf later in life).

In addition to providing captions, Sign Language instruction can be very beneficial for pre-lingual Deaf population who learned signing as their first language. Matt Huenerfauth says that, ”Many Deaf adults have low levels of English literacy because of a lack of exposure to accessible language input during the critical language-acquisition years of childhood” [23]. There have many efforts to make computer science education accessible using ASL instruction. The University of Washington created an online ASL-STEM forum to develop standardized ASL vocabulary for many scientific concepts and terms [24]. Since ASL has a widely dispersed user base [24], these standardized vocabulary are needed for D/HH individuals to learn scientific concepts effectively. However, there is still a dearth of ASL computer science education resources and ASL interpreting is not always available. Automatic English to ASL translation techniques is thoroughly being researched and could be a game-changer for education [23].

2.1.3 Motor Disabilities Population

Individuals with motor disabilities(e.g hand, shoulder, hand) have difficulties operating the Keyboard. They may rely on input modalities such as speech, eye-gaze, or switch control. Speech to text(STT) is a useful interface for individuals with motor disabilities to program and is gaining prominence among individuals with disabilities in recent days. Dragon NaturallySpeaking [25] is a speech recognition software and is widely used for programming by voice. There are also applications such as VoiceCode [25], Serenade [26] and Talon Voice [25] which provide capabilities to write use use natural speech. Talon also provides ability for eye-tracking.

Currently, these tools support text-based programming as far as author’s knowledge and not block-based programming applications. One known issue with VoiceCode is that they typically use words not in the English language [25]. This is done because if users use an English word as a command, such as ‘return’, it may not type out that word in code and instead execute another action mapped to the command ’return’ [25]. We hypothesize this issue will not arise in the context of block-based programming, since most actions will involve selecting, navigating and moving coding blocks, rather than inputting actual programming commands.

There are certain motor disabilities which affect speech as well. For example, many users with cerebral palsy or closed head injury are unable to use speech recognition programs because they may also have dysarthria, which is reduced speech intelligibility caused by neuromotor impairment [27]. In such cases, Augmentative and Alternative Communication (AAC) methods can be used [28] and/or speech recognition systems can be trained on different speech disabilities using Artificial Intelligence to recognize variety of speeches. AAC devices can also be used by students with certain cognitive disabilities [6].

2.2 Research Goals

After evaluating current research research approaches to make K-12 computing education tools accessible, this thesis establishes the following research goals based on the discussed popularity yet inaccessibility of block-based programming and the three most affected disabled population(BLV, D/HH, motor disabilities):

It aims to create Accessible Block-Based Programming tools for the following:

1. BLV population using interactive Text to Speech(TTS), Speech to Text(STT) and screen-reader approaches.

2. Motor Disabilities population by using STT and exploring the case for Language Understanding using Artificial Intelligence to enhance STT solution.

3. D/HH population by creating computer science concept videos in American Sign Language, and informational mini-videos called tooltips for every block-based coding block.

Chapter 3

Accessible Block-Based Programming for K-12 Students who are Blind or Low Vision.

Section 3.1-3.6 material first accepted for publication in HCII 2021 Conference: Das M., Marghitu D., Mandala M., Howard A. (2021) Accessible Block-Based Programming for K-12 Students who are Blind or Low Vision. In: Universal Access in Human-Computer Interaction. HCII 2021. Lecture Notes in Computer Science. Springer, Cham.

3.1 Introduction

Block-based programming relies on visual cues and structures to convey information to users. In addition, they heavily utilize drag and drop gesture to create computer programs. This makes it inaccessible for people who are BLV to interact with them. We reviewed the current approaches to make Block-based programming accessible and used the following research questions in guiding us develop our solution:

1. What interaction techniques can BLV individuals use to understand the structure of a block-based program?
2. What interaction techniques can BLV individuals use to receive non-visual feedback from a block-based program?
3. What interaction techniques can BLV individuals use to debug a block-based program?
4. What are the benefits of a screen-reader vs a non-screen reader approach?

Our contributions ultimately built upon the Keyboard Navigation feature [19] which was released recently by Blockly (Google LLC, Mountain View, CA). Since many Block-based applications utilize the Google Blockly library as a base for their application, we concur building upon their already developed accessible keyboard navigation solution is the best standard approach.

3.2 Related Work

There has been substantial effort in making Block-based programming accessible to individuals with vision needs. Google first released their accessible version of Blockly which replaced the drag and drop layout with a text layout for screen- reader compatibility [29]. Ludi and Spencer's work mainly focused on adding screen reader capability to Blockly, along with keyboard navigation, while maintaining its original block-based interface [8]. Google later released their own version of Blockly with Keyboard navigation for its blocks [19]. Although the blocks in this current version are navigable by keyboard, they neither produce audio output nor are accessible via screen readers. Milne and Ladner created Blocks4all, an iOS touch-based tablet application which is accessible via VoiceOver [9]. Kaushik and Lewis have proposed a non-visual blocks language called Psuedospatial blocks (PB), which distorts spatial layout, and is based on T.V.Raman's idea that instead of making spatial visual data accessible to screen-readers, the focus should be on making better non-visual representations of information. Their application uses synthetic speech instead of a screen-reader for providing output, although they state it will be possible to add screen reader support to it as well [30].

3.3 Our solution

3.3.1 Custom Text-to-speech.

When Google conducted user studies with their initial Accessible Blockly approach, they found that many students were not comfortable using a screen-reader [29]. One of the most popular screen readers, Job Access with Speech(JAWS), is also quite expensive [31]. Some of the students come from low socioeconomic backgrounds and hence may not have had the financial means to purchase such screen-readers. Mac computers come with a free screen-reader called VoiceOver, but a Mac itself is quite expensive. Moreover, learning to use a screen-reader requires considerable practice and immersion making it challenging for users with recent vision loss to become adept at. Therefore, we developed a non-screen reader, self-voicing solution using synthetic speech and interactive text to speech approaches. The ability to use a screen-reader adeptly should not be a prerequisite or a barrier to learning to code, and hence we took this approach.

3.3.2 Retaining spatiality.

The Individuals with Disabilities Education Act(IDEA) requires that students with disabilities receive education in the least restrictive environment, i.e. alongside their non-disabled peers. We make this possible by using a spatial approach to convey information, instead of non-visual approaches which distort spatiality. Although non-visual approaches have been shown to have immense learning benefits for blind users [32], they are not very useful if blind and non-blind individuals want to collaborate and work together. In a study conducted on access overlays for blind users [33], it was shown that, "Access techniques that distort or remove spatial information may reduce users' spatial understanding and memory"; furthermore, this distortion makes it difficult for a blind person to collaborate with sighted peers. Peer programming is commonly used in professional settings and is often utilized by teachers in classrooms. Our approach maintains a spatial yet accessible organization of the block-based coding platform, making it easier for sighted and non-sighted peers to collaborate and learn programming together.

3.3.3 Auditory Cues.

Auditory Cues such as earcons are distinctive sounds that convey certain information. In studies examining teaching robotics to BLV K-12 students, earcons were shown to improve learning of coding concepts [34]. In addition, using non-speech audio cues such as earcons can greatly reduce the cognitive burden that comes with sole speech output [35]. Blockly already features some auditory cues such as a click sound to denote the deletion of a block. Research has shown that sounds produced from different spatial locations are easier to distinguish [36]. For example, when you play video games, you can hear certain audio sounds from only the left/right ear of the headphones or certain sounds can feel "nearer" than others. Audio software enables this process by specifying audio coordinates in a 3D space. This type of binaural spatialization has been effective in math notation feedback and its use has also been investigated in Pencil Code (a block- based coding platform) with positive outcomes [37]. We used this technique to assist students to understand the opening and closing of nested code blocks. For example, a click sound is heard through the left and right side of the earphones respectively to denote the opening and closing of nested blocks.

3.4 Technical Details

Blockly fires an event for almost every change on its workspace [38]. All events contain properties that provide further information about that event. This was the key in developing the voice functionality as we could listen to specific event details on the workspace and convey them via speech to the user. Blockly currently has three cursors in their Keyboard Navigation functionality to navigate through the blocks. We chose the line cursor with few modifications since it closely mimicked a text editor - with the ability to move up and down and next and previous lines of code.

3.4.1 Adding Text to Speech.

Text to speech was added for the following:

1. Toolbox: As the user navigates the toolbox through its various categories and blocks, multiple events are fired. The workspace listens to these events and uses the Web Speech API to communicate the details of that event to the user. In this case, it would be the name of a category or a block.

2. Block connections: Blocks have several connections such as an input connection, block connection, previous connection, next connection, and field connections. Fig 1 shows the different types of connections a block may have. Specific values from these connections were listened to and sent to Web Speech API, and then ultimately conveyed to the user.

3. Marking a connection: In order to connect a block to an existing block on the workspace, a connection has to be marked on the latter. This can be done through navigating to

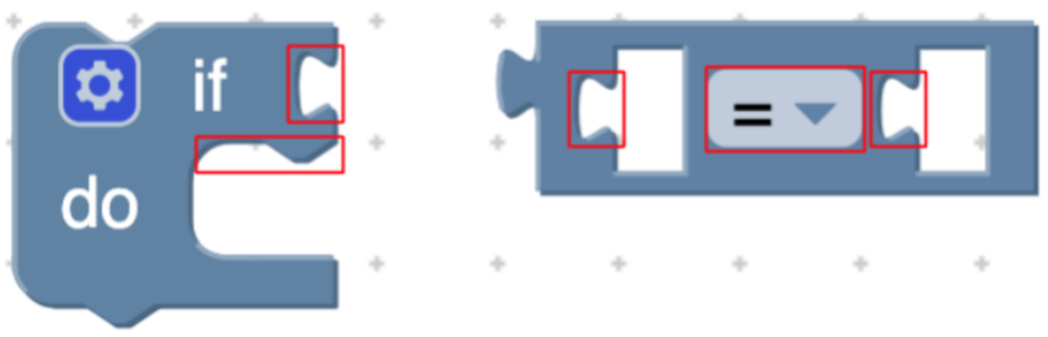


Figure 3.1: Image from Google Blockly's Keyboard Navigation Page which shows different connections of an *if-do* and *logic-compare* block.

the desired connection and pressing the Enter key. This leads to the connection color flickering between red and blue. Since there is no audio feedback for this feature, we listened to this marking event and sent the spoken speech Location marked to the Web Speech API.

4. Connecting two blocks: After marking the desired connection on a block on a workspace, the user proceeds to add a block from the toolbox to connect to that block. We added spoken speech which conveyed the moved block that was connected to the parent block on the workspace. For example, if the repeat block was already on the workspace, and the print block was to be inserted from the toolbox to the repeat block's do connection, the feedback would say, Print block connected to the repeat block. This way the user knows whether the block was inserted into the place it was intended to. If the two blocks are incompatible and cannot be connected, the feedback says, This block can not be inserted at the marked location.

3.4.2 Creating and Firing Custom Events.

At some of the places which required a voice feedback, there was no event present. An example of the location is the dropdown options of a field on a block. In Fig 3.2, a user was able to navigate up and down the dropdown list; however, no event was fired. Hence, there was no capability to add the voice feedback. After consulting with the Google Blockly team on their

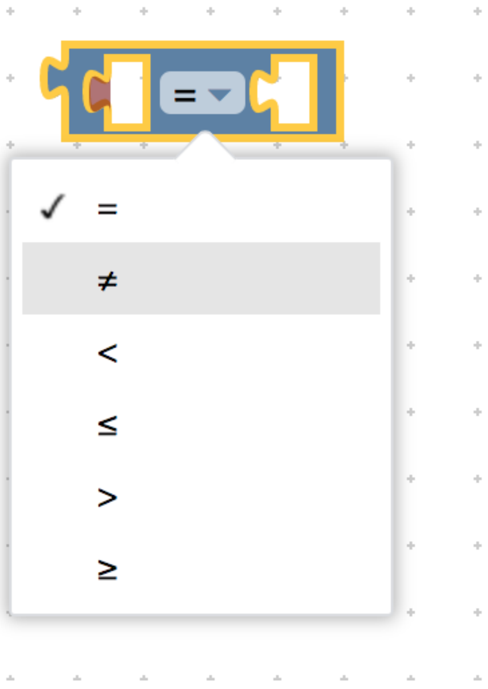


Figure 3.2: Image from Google Blockly which shows the dropdown options of a *logic-compare* block.

forum, we created our own custom events at the needed locations. In our case, for the Fig 3.2 scenario, we added an UI event to go up and down the dropdown field in the core Blockly code, fired and listened to the event, which ultimately allowed us to add the voice feedback.

3.4.3 Adding Keyboard Shortcuts.

Some features of Blockly such as accessing the Tooltip, deleting a block, are not accessible via the keyboard. A Tooltip in Blockly is a user interface element which provides more information on what a block does when you hover over it with a mouse. Hence, after consulting with the Blockly team on their forum, we added custom keyboard shortcuts for these features so a blind user could access them via the keyboard. After selecting a block, the user would have to press CTRL + T to access the tooltip and press the DELETE key to delete a block. In addition to this, we also added a shortcut to get the text representation of a block, including its nested children. For example, on pressing CTRL + I on the outer block in Fig 3.3, the user would hear, if (count = 3) do print " Hello ".

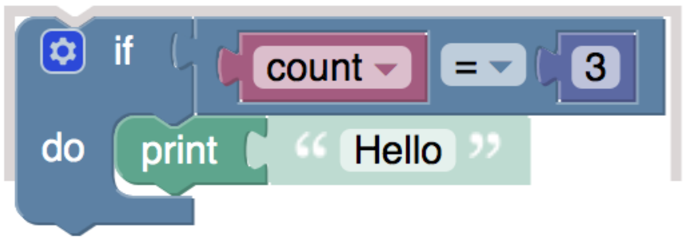


Figure 3.3: A block of code with the following text representation: *if (count = 3) do print " Hello "*.

3.4.4 Adding Auditory cues via Binaural Spatialization.

To signify the opening and closing of a nested block, we made use of binaural spatialization, i.e., directing audio through the left or right audio channel. This was developed using the StereoPannerNode interface of the Web Audio API which provides the capability of panning an audio stream through the left or right. For example, in Fig 3.4, when the cursor is on print block's top connection(as denoted with a red line), a beep is heard through the left audio channel to denote opening of the nested block. Similarly, when the cursor is on print block's bottom connection(as denoted with a blue line), a beep is heard through the right audio channel to denote closing of the nested block.

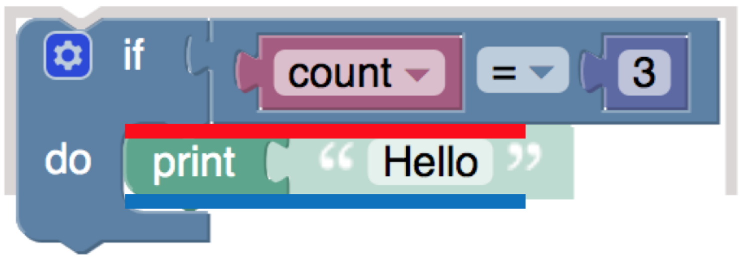


Figure 3.4: A block of code with the following text representation: *if (count = 3) do print " Hello "*. *Print block's* top connection is marked with red, and bottom connection marked with blue.

3.4.5 Accessing Output.

The focus of this work is on Accessible input, i.e. making block navigation, creating, inserting and moving blocks accessible to users who are BLV. For the purposes of testing, we added output in the form of print statements. Participants were asked to write code which printed

Table 3.1: Participant details

Participant	Age	Gender	Level of Corrected Vision	Screen Reader Proficiency
P1	18-22	Female	20/200 to 20/400 severe low vision	Moderate
P2	27-35	Male	Total Blindness with Light Perception	Expert
P3	14-17	Female	20/70 to 20/160: moderate low vision	Low
P4	14-17	Male	more than 20/1,000: near total blindness	Moderate

some text depending upon the logic of the code. This is explained further in the evaluation section. We are currently examining the impact of adding voice feedback to outputs of code in the robot simulation as well. This involves the use of auditory cues and audio descriptions to let user know of the robot's movement and position in a simulation.

3.5 Evaluation

We evaluated our solution with four participants with varying vision needs– low vision to total blindness. Two of these were experienced programmers, while the other two were low or inexperienced in programming. The was done to examine a variety of feedback. Table 1 shows participant's demographic data.

The participants were given some simple warm-up exercises such as entering the toolbox and navigating its categories and blocks, followed by a combination of some, all or related tasks below :

1. Write code using the if-do block which print's Hello world if the value of variable test is equal to 3.
2. Write code using the repeat while block which prints Hello world until the value of a variable named apple reaches 10.
3. Debugging: The following code in Fig 3.5 is supposed to print Hello world until the value of count reaches 3. Find the bug on one line.
4. Debugging: The following code in Fig 3.6 is supposed to print Hello world until the value of count reaches 3. Find the bug on one line.

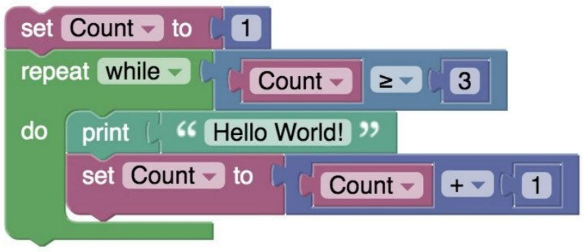


Figure 3.5: A block of code with the following text representation: *set Count to 1, repeat while (Count greater than or equal to 3) do print “ Hello World! ” , set Count to (Count + 1), end of do.*

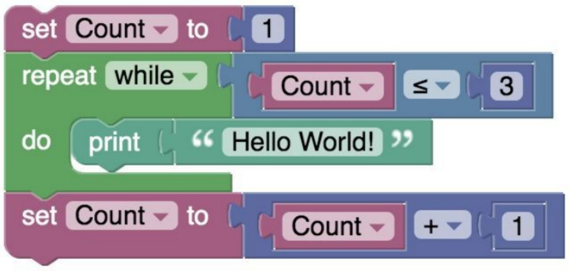


Figure 3.6: A block of code with the following text representation: *set Count to 1, repeat while (Count lesser than or equal to 3) do print “ Hello World! ” end of do, set Count to (Count + 1).*

3.5.1 Participant 1: Experienced Programmer working as a Software Engineer in Industry.

Observation: Participant at first was trying to connect two blocks without marking a connection on the workspace. Due to this, the blocks were simply inserted into the workspace and not connected to the desired block. She tried to move the block from the workspace and connect to the desired block, however current functionality only allows blocks to be connected via toolbox insertion. After this understanding, she was successfully able to complete all the tasks. However, she did not realize that binaural spatialization was used for the nested blocks until specifically told about it. We concur she was still able to debug the statements successfully due to her having light perception and thus using it to understand the nested structure.

Feedback: Participant utilized the audio feedback most of the time with light perception guiding her to understand the structure of the code. As an experienced programmer who does not write code in linear fashion, the major frustration of the participant was not being able to move blocks across the workspace. As a person with moderate screen reader proficiency, she wanted a capability to increase speed of the audio feedback and more options to control the voice and verbosity. The participant liked the navigation and controls and found them easy to work with.

3.5.2 Participant 2: Experienced Programmer working towards a PhD in Computer Science.

Observation: When the participant tried to create a variable, there was no audio feedback provided as to whether the application was ready to take the variable name as input. This is due to the fact that when one clicks on the create variable button in Blockly, a JavaScript alert box pops up asking to type the variable name.

In other words, this is not a part of the Blockly workspace, hence there were no audio feedback provided. We guided the participant to creating the variable in this case. This could have been mitigated if his screen-reader was on to read information outside of the workspace or if synthetic speech was added for this particular instance.

Another instance where we guided the participant was while changing the value of a math block. A value of math block can be changed by navigating to its field and pressing the Enter Key which leads to the current value being selected and thus can be changed. However, no audio feedback was provided after pressing the Enter Key. Hence, we guided the participant in this scenario as well.

In the current cursor, a user can navigate up and down lines/connections of code using keys W and S respectively and can navigate in and out a line of code by using keys A and D respectively. However, if the user reaches the start or end of a block using keys A and D, the cursor will automatically reach the first element of the block on new line, but not the new line's top or input connection. Hence, a connection could not be marked between the two blocks. This led to some confusion to the user as they remained unaware of the start of a new line. Fig 7 illustrates this issue. If a user keeps pressing the D key starting from the if element on line 1, they will eventually reach the do element as marked by orange square in the figure. However, access to the do connection is needed to insert a block between if and do, as illustrated by a black line in the figure. Using the D key does not reach this do connection. Constraining how a user navigates between lines can fix this issue. The participant completed all given tasks.

Feedback: The participant liked the workflow of marking and inserting blocks. He also liked the binaural spatialization to understand the structure of the code once we explained what they meant. The participant said they would have liked more feedback as to how the toolbox was arranged with a more hierarchical description.

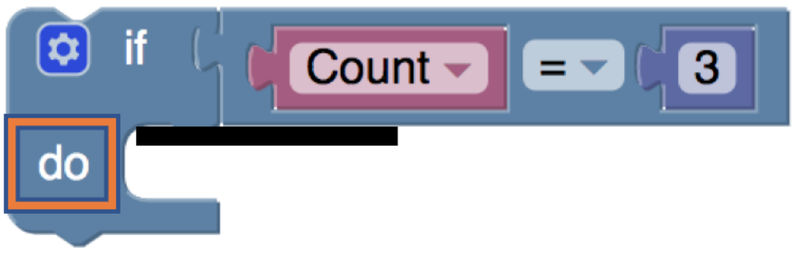


Figure 3.7: A block of code with the following text representation: *if (count = 3) do blank.*

3.5.3 Participant 3: High School Student with minimal programming experience.

Observation: The participant seemed to use a combination of mouse, zoom function and keyboard since she had moderate low vision. The participant experienced a bit of learning curve trying to understand the different connections and markings but once got used to it, was able to write code using blocks. She also had some understanding issues with the cursor as did participant 2. She did understand when a new line had started due to her moderate vision. However, same as participant 2, needed some help to navigate to the input connection of the next line as pressing the D key moved to the first element on the next line, and not its input connection.

Feedback: The participant said the software was easy to understand and the tasks helped her understand the basics of computer programming. She preferred having this synthetic voice than a screen reader because she felt that screen readers repeat words and read out unnecessary information. This can also be due to the fact she had low screen reader proficiency. She also mentioned she would have liked a custom zoom feature to zoom on individual blocks.

3.5.4 Participant 4: High School Student with no programming experience.

Observation: Due to technical difficulties arising from a Bluetooth Keyboard, we shifted our testing approach to a purely audio based one. Using our audio feedback solution, we went over a few programs and asked the participant to answer what the program printed. The programs contained if-do, logic compare and repeat- while blocks. After the participant got used to do the synthetic speech, he was able to answer most of the questions correctly. However, this was purely based on the researcher operating the keyboard and the participant only listening to the audio feedback. The participant had keyboarding skills and we concur with practice the participant should be able to use the keyboard commands successfully as well.

Feedback: We did not get any feedback on the Keyboard Navigation. However, the participant mentioned that they believed with practice they could get a strong understanding of the application.

3.6 Recommendations for Improvements

1. Personalization: Add features to allow users to control the speed and verbosity of speech to match their audio comprehension needs.

2. Currently, there is no functionality to move blocks across the workspace. A block can be connected to another block only through toolbox insertion. We will modify the cursor to add this functionality.

3. Improved Audio Feedback: We will add audio feedback to the creating variable experience as explained in Observation data of Participant 2. Alternatively, we could: (a) move the create variable workflow to the Blockly workspace, as is in Open Roberta [39]. (b) ask users to turn on screen-reader so content outside Blockly workspace is accessible to them.

4. We will also make granular aspects of the workspace accessible. For example, adding audio feedback after pressing the Enter Key to change the math block value. We will also add audio feedback on some hierarchical information of the toolbox. An example of that could be: Logic, category 1 of 9...

5. Improved Cursor: We will constrain how the user can navigate through the blocks so that they are not able to go next or previous lines of code without using their respective keys

specifically. This should fix most of the navigation issues such as not knowing if new line of code has begun as found in usability testing.

6. Other: We hope to add screen-reader support to blocks itself so experienced screen-reader users can benefit. This can be done by using the Aria Live region to communicate where the cursor is. Ludi and Spencer have already done work on this [8]. In addition to this, we will add a zoom feature to assist users with low vision.

   After we make the following changes above, we will re-evaluate our solution with more users who are BLV.

3.7 Progress on Improvements

1. Improved Cursor: We modified the existing cursor to better suit the navigation needs of the participants. The earlier recommendation for line cursor was to not let users navigate to the next or previous lines of code without using those keys specifically. However, upon further research, it was decided that it might not be the best approach. Users will always reach the end of a block line and an additional keystroke will be wasted as the cursor will not move forward. Hence the following approach was taken:

   (a) In our previous line cursor, as described earlier, users could navigate up and down lines/connections of code using keys W and S respectively and could navigate in and out a line of code by using keys A and D respectively. However, if users reach the start or end of a line using keys A and D, the cursor will automatically reach the first element of the block on new line, *but not the new line's top or input connection. The ability for users to reach the new line using the keys A and D was retained, but functionality where it will reach the new line's top or input connection eventually was added.* This way, users will not have to utilize separate keys to mark top or input connection.

   (b) The above new functionality mentioned is the behavior of Google's Blockly Basic Cursor. However, in their Basic Cursor approach, using the previous and next keys, W and S respectively, will also present the same functionality as using keys A and

D. This was tackled by retaining our previous functionality of using W and S keys. Users can still navigate up and down lines/connections of code using keys W and S respectively. Hence, users can save time navigating up and down through connections of code if they wish, while they also have the ability to reach to reach those connections via using keys A and D respectively.

2. Our adapted cursor previously did not provide functionality to move blocks across the workspace. However, with an improved cursor, users now have the ability to move blocks across the workspace.

3. We have added hierarchical information to the toolbox and are working to make the granular aspects discussed in previous section accessible.

4. We are also working to add screen-reader support to the blocks. We are exploring Ludi's approach to utilize Aria Live region to communicate where the cursor is, and also directly talking to screen-readers using their respective APIs.

Chapter 4

Accessible Block-Based Programming using Speech to Text for K-12 Students who are Blind or Low Vision and those with motor disabilities.

4.1 Introduction

In the earlier chapter, Text to Speech(TTS) approaches were discussed for making Block-Based programming accessible. In this chapter, a way of controlling blocks through speech is introduced to aid BLV individuals who may not have Keyboarding skills or those with motor disabilities.

Voice Programming for Scratch, a popular block-based programming app for K-12 students, has been researched before. Myna, a tool developed in Java runs parallel to Scratch as a voice recognition system [40]. When Myna is started and Scratch is opened, users can create programs within Scratch solely through voice [40]. There are also several voice tools for text-based programming mentioned in Chapter 2.

4.2 Methodology: Building Upon Google's Keyboard Navigation

This Speech to Text(STT) approach utilizes Google's Keyboard Navigation feature to control blocks. A JavaScript library called Annyang [41] is utilized which allows for users to add voice functionality to their applications. Annyang receives the voice commands from the user, and dispatches a Google Keyboard Navigation Event mapped to that command. Once this event is successful, the TTS feature described in last chapter provides the necessary information about the event to the user. Figure 4.1 illustrates this methodology.

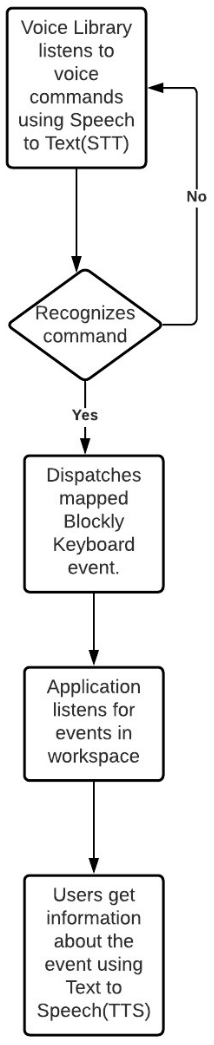

Voice Library listens to voice commands using Speech to Text(STT)
No
Recognizes command
Yes
Dispatches mapped Blockly Keyboard event.
Application listens for events in workspace
Users get information about the event using Text to Speech(TTS)

Figure 4.1: STT and TTS Methodology

4.3 Supported Voice Commands

Currently, the application supports the voice commands displayed in Table 4.1 to control various coding blocks.

Table 4.1: Supported Voice Commands

Command	Action
Toolbox	Enters the Toolbox
in	cursor proceeds to next block element
out	cursor proceed to previous block element
next	cursor proceed to next line
previous	cursor proceed to previous line
select	selects a block from the toolbox
mark	marks a connection on a block
insert	inserts selected block to a marked connection
detach	detaches a block from another block
tooltip	displays the tooltip for a selected block
delete	deletes a block
information	displays text representation of a block

4.4 Workflow Demonstration

Below is a short workflow of how the STT feature works with the TTS feature. It demonstrates marking the *do connection* of a *repeat block.*

(a) Initial Blockly Workspace

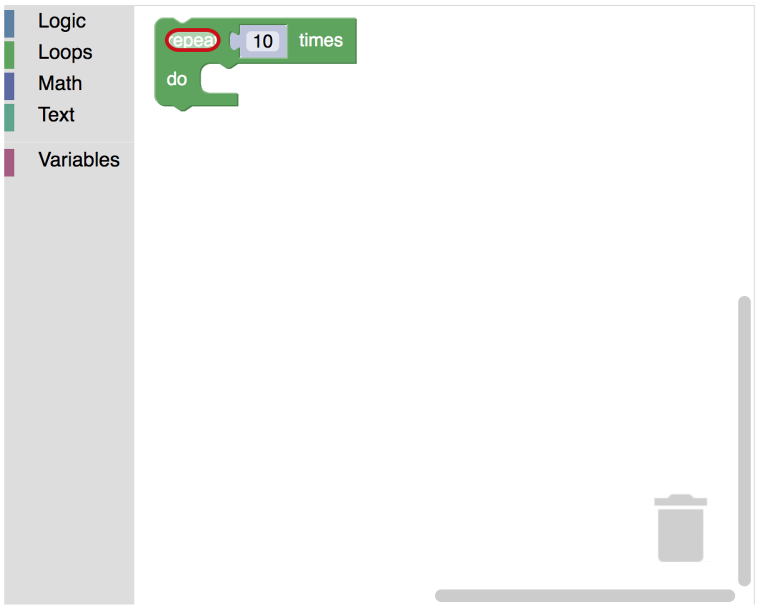


Figure 4.2: A Blockly environment with a *repeat 10 times* block on its workspace. The repeat field is circled in red

(b) Blockly Workspace after the voice command, ”Next”. This moves the cursor to the *do connection* which is also conveyed via TTS, as shown in the speech bubble.

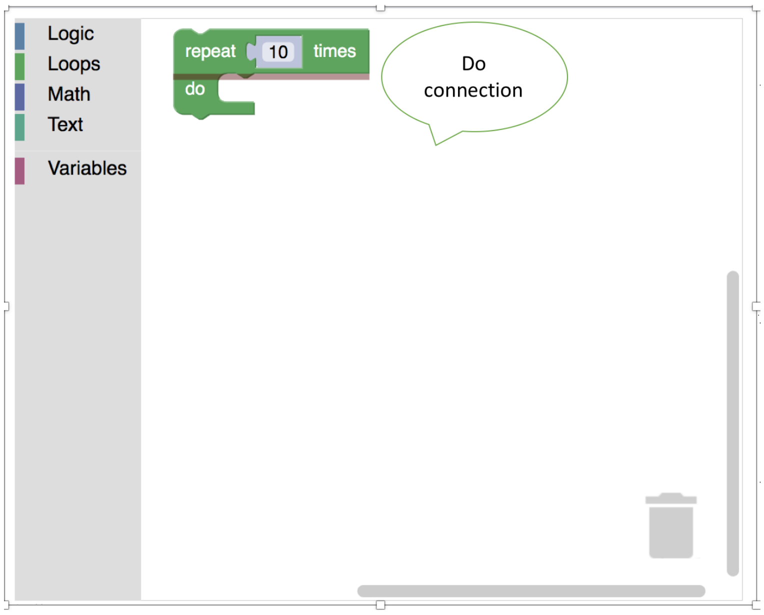


Figure 4.3: A Blockly environment with a *repeat 10 times* block on its workspace. The *do connection* is illustrated with a red line.

(c) Blockly Workspace after the voice command, ”mark”. This marks the do connection which is also conveyed via TTS, as shown in the speech bubble.

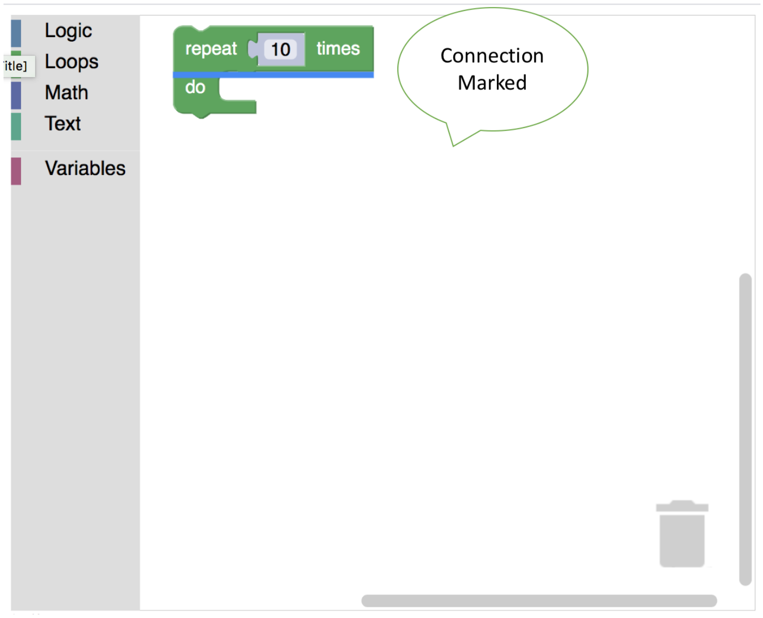


Figure 4.4: A Blockly environment with a *repeat 10 times* block on its workspace. The marked *do connection* is illustrated with a blue line.

4.5 Future Work

(a) Conversational Interface: A more conversational interface using voice commands will be created. Currently, the system only supports commands which can perform one action at a time. However, it would be useful for the system to execute multiple actions given a single command. For example, if a user wants to create a variable in the current system, they would have to first say the necessary commands to enter the toolbox, navigate to the variables category, choose the create variable block, name the variable in the alert box, and then press Enter. An improved version would be where the user asks the system to create a variable, and the system immediately displays the alert box to create the new variable. This functionality requires usage of Natural Language Processing models.

(b) Utilizing Natural Language Processing(NLP): Training on a variety of different utterances and accents is important to ensure a robust experience. For example: a task may be, “Delete the top-most block on the workspace”. One user may choose to say, “Delete the first block on the workspace” and other may say, “Delete the top block on the workspace”. These tasks are the *intents* and the two examples above are known as *user utterances*. These intents will be added to our model and then trained, tested and validated. This will ensure STT is accurate with a variety and diversity of users. This will also ensure personalized TTS voices are created depending on the users. In addition to this, active learning will also be utilized to improve the model accuracy. When users interact with the system with their commands, their utterances will be fed into the existing model and trained to further improve the personalization experience.

Chapter 5

Accessible Computer Science for K-12 Students who are Deaf/Hard of Hearing

Section 5.1-5.3 material first accepted for publication in HCII 2020 Conference: Das M., Marghitu D., Jamshidi F., Mandala M., Howard A. (2020) Accessible Computer Science for K-12 Students with Hearing Impairments. In: Antona M., Stephanidis C. (eds) Universal Access in Human-Computer Interaction. Applications and Practice. HCII 2020. Lecture Notes in Computer Science, vol 12189. Springer, Cham. https://doi.org/10.1007/978-3-030-49108-6_13

5.1 Introduction

American Sign Language (ASL) is a language very distinct from English and contains "all the fundamental features of language, with its own rules for pronunciation, word formation, and word order" [42]. A study by Heunerfauth and Hanson revealed that Deaf individuals often acquire sign language as their first language, and they are most fluent and comfortable in it [43]. They stated that sign language interfaces are a necessity for such D/HH individuals. Websites such as YouTube provide closed-captioning features for videos, but the captions are often inaccurate, and the reading level of this text is sometimes too complex and difficult for deaf individuals. Half of deaf students pass from secondary school with a fourth-grade reading level or less [44]. Another study stated that the frequently reported low literacy levels among students with severe hearing disability were partly due to the discrepancy between their "incomplete spoken language system and the demands of reading a speech-based system" [45]. This illustrates the need of providing coding instruction in ASL. D/HH students are at a high risk of unemployment or chronic underemployment at a rate of 70% or more nationally [46]. The number of tech job openings in the country is growing exponentially and hence, by encouraging D/HH students to learn CS concepts and tackle programming challenges from the middle school level using accessible coding instruction, we can help to lower the unemployment rate.

5.2 Related Work

Despite the fact that human signing videos are expensive and hence difficult to update, there is a need to produce sign language content explaining core computer programming concepts which don't require frequent updating. For example, in the CS field, ASLCORE has produced signs and definitions for vocabulary and concepts such as "Recursion", "Debugger", "Linked List" and "Variable". In this way, CS jargon is being made accessible to older students who are D/HH, who have to otherwise rely on fingerspelling while communicating with this vocabulary. Apple has also recently released seven videos [47] explaining computer programming concepts in ASL. On the other hand, Drag and Drop coding applications such as MIT's Scratch [48] are popularly used to teach younger students to code. Our project aims to make CS principles, using Block-based coding interfaces, a more inclusive experience for D/HH students since many students' first experience with programming is through such coding interfaces.

A number of ambitious projects (e.g. automatic sign language recognition, animated 3D humanlike characters who perform sign language, and reading assistance technologies) are being developed to make web content more accessible to D/HH students. Although substantial work [43] is being done in creating this animated sign language using avatars, which automatically converts a body of text into avatars signing them, the majority of the work is still in the research phase. Our project does not aim to create sign language videos for every content related to CS on the web; automated virtual human sign technologies, when fully developed, are a better fit for this purpose. Instead, our objective is to create engaging human signing videos for core computing science concepts and common data structures in the context of a block-based programming environment. There are two primary reasons for doing this:

(a) To increase pre-lingual deaf students' excitement about CS: According to the Communication Services for the Deaf, "98% of deaf people do not receive education in sign language" [49]. Lisa Harrod, a former sign language interpreter, mentions that captioning is simply the written form of spoken words which can be very difficult for D/HH who struggle with English as a second language [50]. Pre-lingual individuals (born with deafness or became deaf very young) who learned ASL as their first language will more likely be interested to learn coding through ASL, rather than just reading paragraphs of text or captioned videos explaining computer programming concepts. Therefore, this project tries to provide a more promising and engaging learning environment for these students that possibly their second language could not. In addition, we hypothesize that a student's sense of inclusion will increase with the knowledge that computer programming is being taught in a language that addresses their needs.

(b) To improve post-lingual deaf students comfort level with common CS ASL signs: For post-lingual students who learned ASL later in life and are not very comfortable with it, web applications should provide captioning and/or transcripts for every ASL video, so that they do not have to solely depend on ASL. However, our plan is to familiarize post-lingual deaf students with the basic ASL signs for programming concepts, so they may use it to communicate with other deaf students, in addition to providing captioning in videos.

5.3 Conceptual Details

This project is part of a larger research initiative which aims to develop an accessible block-based web application for middle school students with disabilities. The app aims to provide accessibility features for students who are either blind/visually impaired, D/HH, have physical, and/or cognitive disabilities. This paper particularly focuses on the accessibility considerations for the D/HH audience, which is further divided into two parts. The first part discusses the development considerations for videos explaining core CS concepts in ASL, which will be embedded as part of the curriculum for the accessible web application project. The signs for STEM terminologies will be based on ASLCORE to ensure technical accuracy. Our ASL videos explain the following programming concepts with real-life application examples: a. Introduction to CS & Programming b. Pseudocode & Algorithms c. Commands d. Conditional Statements & Event Driven Development. e. Loops f. Variables g. Functions h. Debugging. i. Object-Oriented Programming. These concepts are inspired by materials from CS Teachers Association's K-12 CS Standards, and the CS4ALL curriculum developed for an inclusive robot camp at Auburn University [51].

It is proven that D/HH individuals are visual learners [52], hence every video will demonstrate application of the particular programming concept through a block-based coding interface. Videos for other CS concepts which explain the social aspect of computing, such as cybersecurity, bias in computing, accessibility, and careers in computing, are also being filmed. We have received a SIGCSE special projects grant [53] to produce these videos. Hence, we have partnered with the non-profit organization Deaf Kids Code [46] to recruit D/HH STEM teachers and successful deaf technologists to star in these videos. Another key reason to specifically produce these computing videos in ASL is to avoid the denotation and connotation confusion of English words [54], which D/HH individuals sometimes experience. For example, the word 'call' in English has the most common meaning of contacting someone or giving someone a name. However, in the CS context, it means to execute a subroutine or a function. Young D/HH kids, especially those in middle school, may not have the contextual knowledge to compensate for these informational gaps [54], hence explaining and clarifying these contextual differences through ASL videos would be extremely helpful for D/HH middle-schoolers. Additionally, research shows that the majority of teachers for the D/HH are hearing. 2,575 K-12 D/HH teachers were surveyed and it was found that less than 22% were deaf [55]. Only 38% of D/HH teachers believed that "their expressive sign skills were on par with their expressive spoken English" [55]. This suggests that only a few D/HH K-12 individuals have access to teachers who communicate with them using ASL, let alone using appropriate signs for STEM terminologies. Hence, our videos can be used by STEM teachers in their classrooms to introduce students to CS concepts.

The second part of this project is developing built-in accessibility features for D/HH audience for the Block-based coding web application. The web application consists of some programming tutorials similar to Activity Guides [56] in MIT's Scratch. Additionally, it consists of an extensive robot platform where students can use programming blocks to control a robot. The benefits of learning CS through robotics are well researched [57]. In this sense, several robotic environments are being proposed to facilitate programming teaching, such as Lego Mindstorms [58] and NaoBlocks [59]. The Finch robot designed to align with the learning goals and concepts taught in introductory CS courses is extremely popular [60]. Our robot, which is designed at the Georgia Institute of Technology, is cost-effective and visually appealing to middle schoolers. It also has the capability to produce sounds which can serve as feedback for blind/visually impaired individuals. The robot has a wireless connection and commands to control it, via blocks, can be sent through Bluetooth from the web application.

Figure 5.1 is a picture of the top-view of our robot.

Figure 5.2 is a picture of the bottom-view of our robot.

The guidelines for the heuristic evaluation for Deaf Web User Experience (HE4DWUX) were used to develop a product requirements list for D/HH accessible web application [61]. This heuristic evaluation is based on the well-known usability heuristics developed by Jakob Nielsen [62]. The HE4DWUX consists of similar heuristics as Nielson's and provides additional context with regards to accessibility for the D/HH.

We discuss the following accessibility features of our app with regards to seven relevant heuristics described in the HE4DWUX.

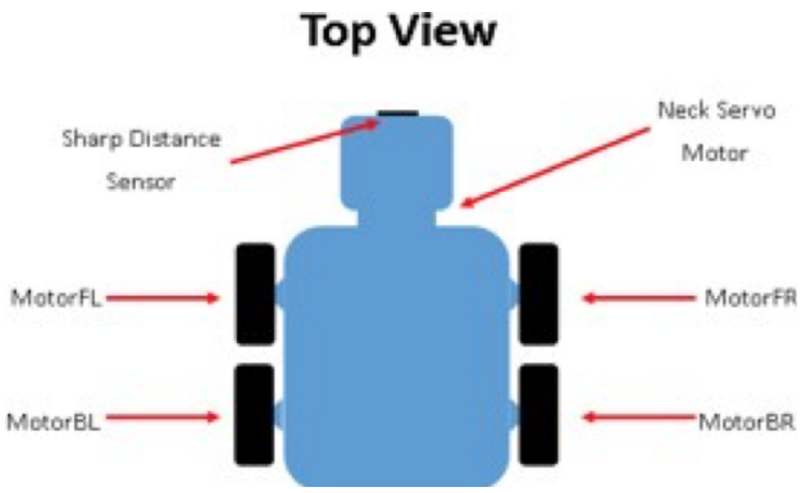


Figure 5.1: Top View of the robot showing the Sharp Distance Sensor, Neck Servo Motor, Front-left Motor, Back-left Motor, Front-right Motor & Back-right Motor

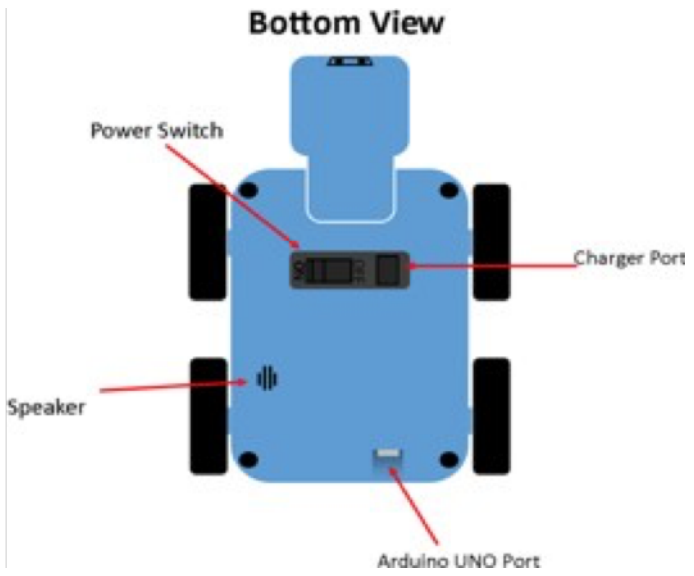


Figure 5.2: Bottom View of the robot showing the Power Switch, Speaker, Charger Port and the Arduino UNO Port.

(a) Sign language video content—The website should support captioned video content, alternative text and make use of signers in videos.

The ASL videos will be embedded in the web application. Figure 5.3 is a prototype of this feature in action. The ASL video for this particular tutorial would explain the concept of commands such as ‘turn left’, ‘turn right’, and ‘move forward’.

(b) Visibility of status and actions—The website should provide users who are deaf with timely visual-based feedback on actions.

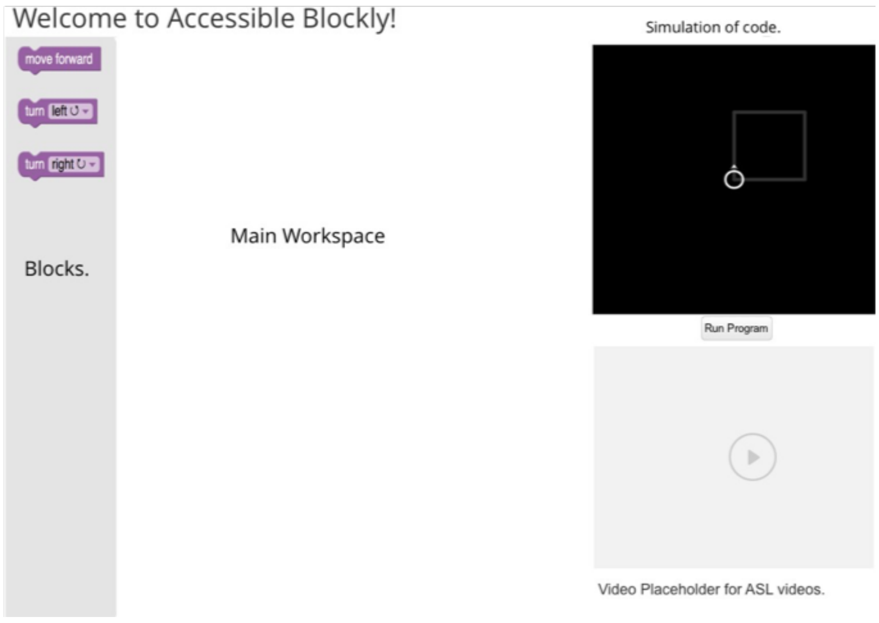


Figure 5.3: Under-development block coding interface showing Placeholder for ASL videos. Block and simulation images taken from Blockly Games.

Good visual feedback is very important to ensure web accessibility for D/HH. For example, when a user clicks on the run button to execute their code, it is necessary to provide clear visual cues such as text or graphics on screen, which can convey that the program has been run, as opposed to just audio cues. Some other examples of visual-based feedback in consideration are ensuring that content changes are indicated near the area of interest, using alternatives to static and textual forms of feedback, and text with icons for better navigation [59].

(c) Match between website and the world of the Deaf—The website should be aligned with language and concepts from Deaf culture.

Our signs for programming concepts are heavily based on the work done by ASLCORE [63] since they have done extensive research in creating signs for technical programming terms such as “IF”, “WHILE” and “FOR”. Our partnership with Deaf Kids code allows us to understand the needs and expectations of D/HH students. ASLCORE recruits a 4-person all Deaf translation team who have strong and intuitive knowledge of ASL [64]. Similarly, only ASL experts will participate in the making of our videos to ensure authenticity and correctness. The web application also provides a feedback form so that users may let us know of any potential issues in the videos.

(d) User control and freedom—The website should make users who are deaf feel in control of their experience.

Users are able to control the speed of ASL videos and increase the size of captions. ASL signs/videos for every drag and droppable ‘block’ like “IF-ELSE” or “WHILE”, are displayed as tooltips for a block. Tooltip is a common graphical user interface element. It is used jointly with a cursor, commonly a pointer [65]. A research study conducted with 15 deaf users found that sign language and picture tooltips were very positively rated as opposed to only text-based tooltips or a human speaking-based tooltip (used for lip-reading) [66]. In our web application, once a user hovers their cursor over a block, it will play the sign for the block in ASL. Figure 5.4 is a prototype of this feature in action.

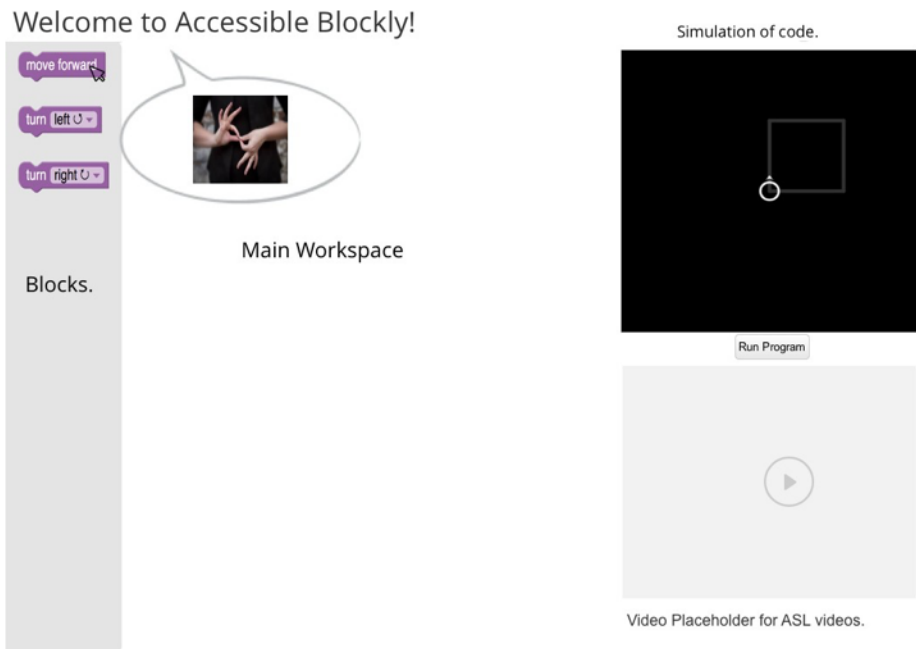


Figure 5.4: Under-development block coding interface displaying tooltips in ASL as a user hover over a block. Block and simulation images taken from Blockly Games. Handshape image taken from Calliope Interpreters.

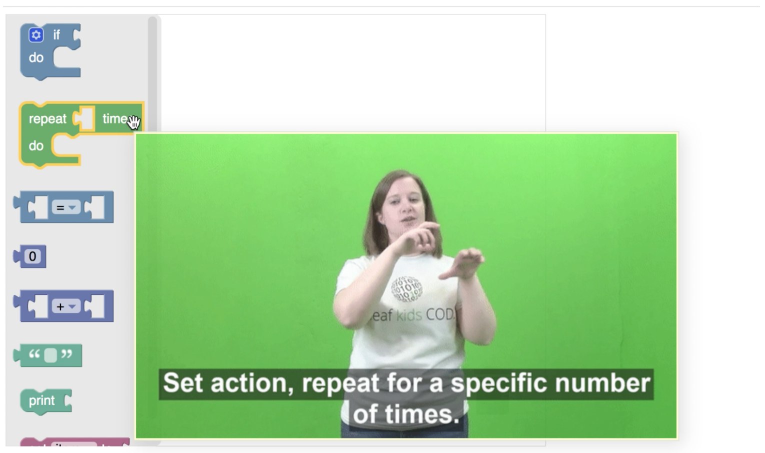


Figure 5.5: Blockly Interface displaying a developed tooltip in ASL as mouse hovers over a *repeat block*.

However, if the user would prefer text as tooltips instead of ASL, they have the option to change to text view as well. This ensures users have control on the features of the web application. Figure 5 is a prototype of this feature in action.

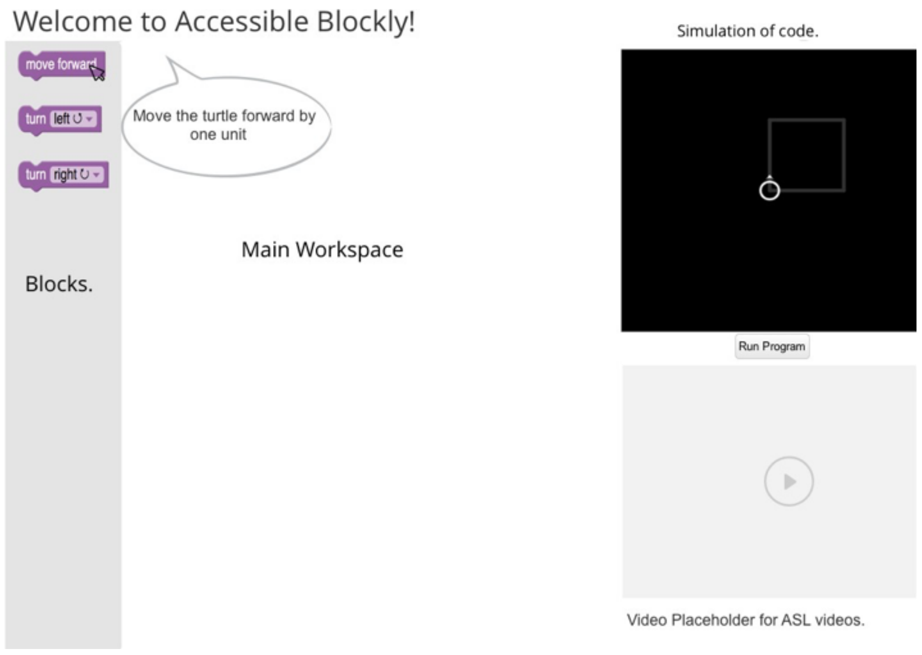

Welcome to Accessible Blockly!
move forward
turn left
turn right
Move the turtle forward by one unit
Main Workspace
Blocks.
Simulation of code.
Run Program
Video Placeholder for ASL videos.

Figure 5.6: Under-development block coding interface displaying tooltips in English as opposed to in ASL, if user prefers English instead. Block and simulation images taken from Blockly Games.

(e) Captions—The website should describe dialogue and sound effects (audio content). Lisa Harrod also mentions, "Captioning is perfect for the post-lingual deaf or hard-of hearing audience; it presents content in an accessible format, in the primary language of the user" [50]. Every ASL video in our web application includes captions to account for post-lingual D/HH individuals who prefer captions.

(f) Aesthetic and minimalist design—The website should contain the most essential elements and any non-relevant or rarely needed content that will be competing with other units of content for visibility and attention must be removed.

    Our application aims for web content accessibility conformance (e.g. using sufficient color contrast, allowing zooming, minimizing distracting and cluttering content, and using proper headings). Hence, visual noise is eliminated to reduce memory overload. This will also be beneficial to individuals with cognitive disabilities.

(g) Personal skills—The website should support the skills and background knowledge of users who are deaf and not replace these.

    We incorporate multimodal learning in our app which is beneficial to D/HH individuals. Multimodal approaches "involve the use of multimedia along with the content text and encourage the use of interactive literacy practices between the instructor and the students". Lacina and Mathews researched how multimodal learning tools affect the learning process, and "found that it provided a pleasant experience, captured students' attention, increased their motivation of reading, and enhanced their literacy skills" [67]. Deaf explore the world mostly by the sense of sight and vision. "Graphics, pictures, videos, and the different forms of multimedia can facilitate learning of complex theories and generalizations, which are usually challenging for deaf students that do not know how to use sign language" [68]. Involving multiple representations of the educational materials can help in decreasing the loss of the received data. "These multimodal representations for deaf students can combine text and sign language, or visual displays (e.g., pictures or videos) along with sign language subtitles" [66]. There is a diversity of learning styles among normal students, they could be verbal, visual or a different type of learners. Therefore, it's crucial to include all possible learning styles for deaf students' e-learning techniques [69] [70].

5.4 Evaluation

Eight computer science videos in ASL were filmed in partnership with Deaf Kids Code. The videos were about 2-3 minutes long in length. Figure 5.7 displays a shot from the *Introduction to Computer Science* ASL Video. The following were the topics:

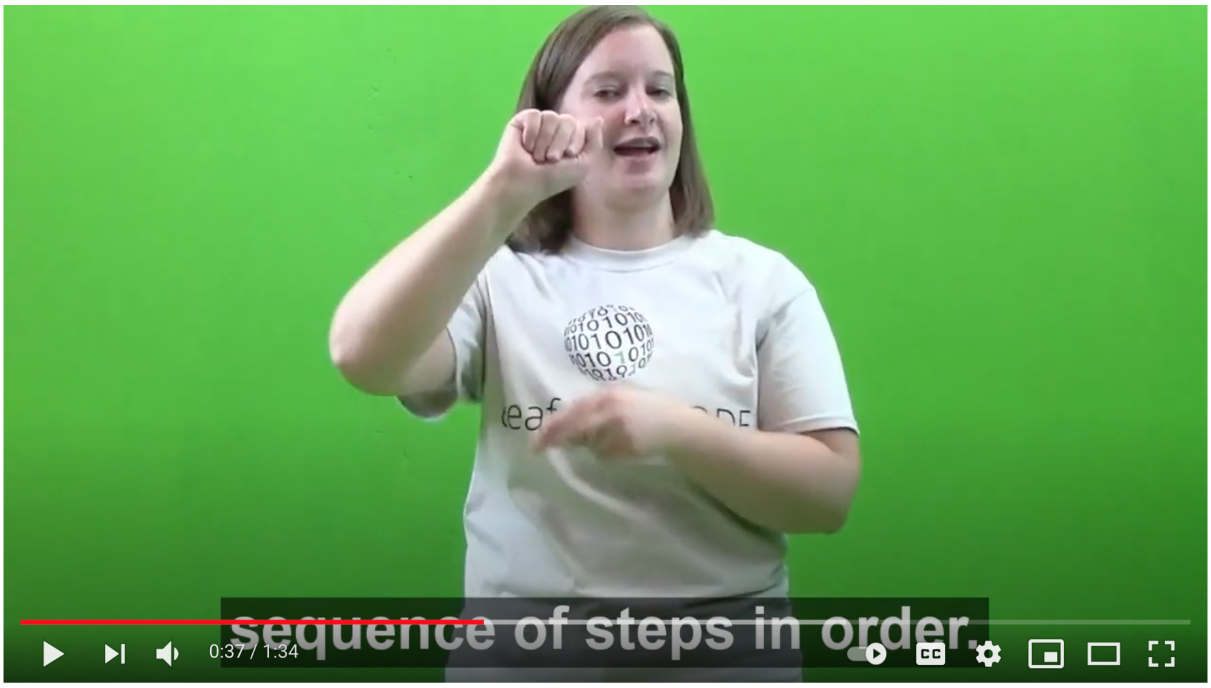


Figure 5.7: A shot from the *Introduction to Computer Science* ASL video.

(a) Introduction to Computer Science: This video explains the basics of computational thinking which involves breaking down actions into sequence of steps. It also provides an example of this by explaining the process to make a PB&J Sandwich.

(b) Programming Language Commands: This videos explains how computers take input, process it, and generate output. It also provides an example of a pizza in the microwave, where we set the timer, press the start button, and the pizza is ready.

(c) Conditional Statements: This video explains the basics of IF, AND, OR conditional statements which help us decide between situations.

(d) Loops: This video explains the basics of Loops which repeats a certain situation until a condition is met. It provides an example of looping to finish the food on the plate until the plate is empty.

(e) Variables: This video explains the basics of Variables which store the values in the computer memory and can be referenced by using their names. It also provides an example of setting a variable named *score* and how to add to or subtract from it.

(f) Functions: This video explains the basics of Functions which are blocks of code that represent one action and take input and produce output. It also provides an example of a cow that eats grass and produces milk.

(g) Debugging: This video explains the basics of Debugging which is finding and fixing problems in a block of code. It also provides an example of how we solve car problems by filling the gas tank or getting a new battery.

(h) Object-Oriented Programming: This video explains the basics of Object-Oriented Programming which allows to create objects and assign them details. For example, a car object with brakes, wheels, and high speed.

Although a professional signer with computer science background advised and starred in these videos, we further evaluate these videos with two adults with moderate ASL proficiency and a computer science background. Table 5.1 presents the participant information.

Table 5.1: Participant Information

Age	Experience with Programming	ASL Proficiency	Assistive Technologies/Accommodations Used
18-22	Extensive Experience: Wrote Many Programs	Level 3: Able to sign ASL with sufficient structural accuracy and vocabulary to participate effectively in most formal and informal conversations pertaining to practical, social, and professional needs.	lip-reading, cochlear implant, ASL.
27-35	Substantial Experience: Wrote several small to medium-sized programs	Level 3: Able to sign ASL with sufficient structural accuracy and vocabulary to participate effectively in most formal and informal conversations pertaining to practical, social, and professional needs.	lip-reading, cochlear implant, ASL, CART(Real-Time captioning)

The following five findings were distilled from this evaluation:

(a) Captions and spoken audio in sync with ASL: In some places, the ASL lagged behind the captions and the spoken audio. It is difficult to synchronize ASL and

English since they are structurally different languages, however we will improve this further.

(b) Producing ASL-only versions: It was found that the participants found it distracting to focus on both ASL and spoken audio/captions at the same time. We will make available ASL only versions of the videos.

(c) Considerations for ASL-English Bilingual audience: Participant suggested that for a ASL-English Bilingual audience, it would help to describe a key concept in ASL and also spell or sign the English form. We will explore this further.

(d) Use of Graphics: As evidenced in literature before, multimedia such as images are helpful for D/HH individuals to learn better. We will explore adding images so that the audience can visualize what the signer is saying.

(e) Use of appropriate ASL signs: Participant gave an example of one instance in the videos. "If a condition is met" in English does not equate to the TO-MEET sign used in the video. They recommended to sign IF TRUE, which is more accurate.

Apart from their suggestions, the two participants had positive things to say about the ASL CS curriculum. Participant 1 said that the videos made her feel *"Good, excited to learn more about CS in ASL."* and *"taught me more CS vocab in ASL."* Participant 2 said *"The ASL videos aided me in further understanding and appreciating computer science concepts - I loved the analogies and I wish that my computer science education included more visual examples."* Although statistical conclusions cannot be drawn from this evaluation, it is evident from past literature and these participants that D/HH individuals like learning about computer science in ASL.

Chapter 6

Conclusion

The following were the goals established at the beginning of this research: Create Accessible Block-Based Programming tools for the following:

1. BLV population using interactive Text to Speech(TTS), Speech to Text(STT) and screen-reader approaches.
2. Motor Disabilities population by using STT and exploring the case for Language Understanding using Artificial Intelligence to enhance STT solution.
3. D/HH population by creating computer science concept videos in American Sign Language, and informational mini-videos called tooltips for every block-based coding block.

The research was successful in meetings these goals. Evaluation with four Blind/Low Vision(BLV) and two Deaf/Hard of Hearing(D/HH) individuals was also done. Their feedback is being incorporated to better our solution for the needs for the disabled population. Future work with regards to designing Natural Language Processing(NLP) techniques for the Speech to Text(STT) feature is actively being worked on. The codebase for the solution will also available on GitHub under a certain modification license so that other researchers can actively utilize and modify this work. This is a great step towards ensuring access to computer science for people with disabilities.

References

[1] Computer and Information Research Scientists: Occupational Outlook Handbook:: U.S. Bureau of Labor Statistics. https://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm. Accessed 30 Mar. 2021.

[2] Ladner, R. (2009). Persons with Disabilities: Broadening Participation and Accessibility Research. Computing Research News, 21(2). Retrieved from http://cra.org/resources/crn-archive-view-detail/persons_with_disabilities_broadening_participation_and_accessibility

[3] WebAIM: PDF Accessibility - Defining PDF Accessibility. https://webaim.org/techniques/acrobat/. Accessed 30 Mar. 2021.

[4] Wilson, J. D. (2017). Reimagining disability and inclusive education through universal design for learning. Disability Studies Quarterly, 37(2).

[5] Mealin, S., & Murphy-Hill, E. (2012, September). An exploratory study of blind software developers. In 2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 71-74). IEEE.

[6] Huff, E., Koushik, V., Ladner, R., Ludi, S., Stefik, A., Perkoff, M., . . . Milne, L. (2020). Accessible Tools and Curricula for K-12 Computer Science Education. Retrieved March 30, 2021, from https://www.microsoft.com/en-us/research/uploads/prod/2021/01/Accessible-Tools-and-Curricula-for-K-12-Computer-Science-Education-1.21.pdf

[7] Hansen, J. P., Rajanna, V., MacKenzie, I. S., & Bækgaard, P. (2018, June). A Fitts' law study of click and dwell interaction by gaze, head and mouse with a head-mounted display. In Proceedings of the Workshop on Communication by Gaze Interaction (pp. 1-5).

[8] Ludi, S., & Spencer, M. (2017). Design Considerations to Increase Block-based Language Accessibility for Blind Programmers Via Blockly. Journal of Visual Languages and Sentient Systems, 3(1), 119-124.

[9] Milne, L. R., & Ladner, R. E. (2018, April). Blocks4All: overcoming accessibility barriers to blocks programming for children with visual impairments. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-10).

[10] Are There Hour of Code Activities That Are Accessible to Students with Visual Impairments? — AccessComputing. https://www.washington.edu/accesscomputing/are-there-hour-code-activities-are-accessible-students-visual-impairments. Accessed 30 Mar. 2021.

[11] "Learn Today, Build a Brighter Tomorrow." Code.Org, https://code.org/hourofcode/overview. Accessed 30 Mar. 2021.

[12] Game Accessibility Guidelines — Ensure Screenreader Support, Including Menus & Installers. http://gameaccessibilityguidelines.com/ensure-screenreader-support-including-menus-installers/. Accessed 30 Mar. 2021.

[13] Ladner, R. E., & Israel, M. (2016). For all" in" computer science for all. Communications of the ACM, 59(9), 26-28.

[14] Israel, M., Dey, S., Dimitriadi, Y., Feldner, H., Lsvik, A., Kuriakos, N., . . . India, G. (2020). Reimagining Accessibility and inclusion in K-12 CS Education through curriculum and professional development. Retrieved March 30, 2021, from https://www.microsoft.com/en-us/research/uploads/prod/2021/01/Reimagining-Accessibility-and-Inclusion-in-K12_Making-Higher-Ed-in-CS-Accessible-Group-B.pdf

[15] Huff Jr, E. W., Boateng, K., Moster, M., Rodeghero, P., & Brinkley, J. (2021, March). Exploring the Perspectives of Teachers of the Visually Impaired Regarding Accessible

K12 Computing Education. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (pp. 156-162).

[16] Stefik, A., Ladner, R. E., Allee, W., & Mealin, S. (2019, February). Computer science principles for teachers of blind and visually impaired students. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 766-772).

[17] Stefik, A., & Siebert, S. (2013). An empirical investigation into programming language syntax. ACM Transactions on Computing Education (TOCE), 13(4), 1-40.

[18] Brown, N. C., Mönig, J., Bau, A., & Weintrop, D. (2016, February). Panel: Future directions of block-based programming. In Proceedings of the 47th ACM technical symposium on computing science education (pp. 315-316).

[19] "Keyboard Navigation — Blockly." Google Developers, https://developers.google.com/blockly/guides/configure/web/keyboard-nav. Accessed 30 Mar. 2021.

[20] National Association of the Deaf - NAD. https://www.nad.org/resources/american-sign-language/community-and-culture-frequently-asked-questions/. Accessed 30 Mar. 2021.

[21] NAD: A Promising Practice in Streaming Captioned Educational Video — AccessComputing. https://www.washington.edu/accesscomputing/nad-promising-practice-streaming-captioned-educational-video. Accessed 30 Mar. 2021.

[22] Kheir, R., & Way, T. (2007, June). Inclusion of deaf students in computer science classes using real-time speech transcription. In Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (pp. 261-265).

[23] Huenerfauth, M. (2006). Generating American Sign Language classifier predicates for English-to-ASL machine translation (Doctoral dissertation, University of Pennsylvania).

[24] ASL-STEM Forum — About Us. https://aslstem.cs.washington.edu/info/about. Accessed 30 Mar. 2021.

[25] Nowogrodzki, A. (2018). Speaking in code: how to program by voice. Nature, 559(7712), 141-143.

[26] "Serenade." Serenade, https://serenade.ai. Accessed 30 Mar. 2021.

[27] UASpeech Database. http://www.isle.illinois.edu/sst/data/UASpeech/. Accessed 30 Mar. 2021.

[28] Kouroupetroglou, G. (Ed.). (2013). Disability informatics and web accessibility for motor limitations. IGI Global.

[29] Google/Blockly-Experimental. 2019. Google, 2020. GitHub, https://github.com/google/blockly-experimental.

[30] Koushik, V., & Lewis, C. (2016, October). An accessible blocks language: work in progress. In Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility (pp. 317-318).

[31] "Everything you need to know about screen readers." Canadian Assistive Technologies Ltd., https://canasstech.com/blogs/news/everything-you-need-to-know-about-screen-readers. Accessed 19 Feb. 2021.

[32] Lewis, C. (2014). Work in Progress Report: Nonvisual Visual Programing. In B.duBoulay and J.Good (Eds) Proc. PPIG 2014 Psychology of Programming Annual Conference, 25th Anniversary Event. Brighton, England, 25th -27th June 2014.

[33] Kane, S. K., Morris, M. R., Perkins, A. Z., Wigdor, D., Ladner, R. E., & Wobbrock,J. O. (2011, October). Access overlays: improving non-visual access to large touch screens for blind users. In Proceedings of the 24th annual ACM symposium on User interface software and technology (pp. 273-282).

[34] Dorsey, R., Park, C. H., & Howard, A. (2014). Developing the capabilities of blind and visually impaired youth to build and program robots.

[35] Brewster, S. A. (1997). Using non-speech sound to overcome information overload. Displays, 17(3-4), 179-189.

[36] Murphy, E., Bates, E., & Fitzpatrick, D. (2010, October). Designing auditory cues to enhance spoken mathematics for visually impaired users. In Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility (pp. 75-82).

[37] Ludi, S., Wang, J., Chapati, K., Khoja, Z., & Nguyen, A. (2019). Exploring the Use of Auditory Cues to Sonify Block-Based Programs.

[38] Events — Blockly. Google Developers, https://developers.google.com/blockly/guides/configure/web/events, Accessed 10 Feb. 2021.

[39] Open Roberta Lab. https://lab.open-roberta.org/. Accessed 10 Feb. 2021.

[40] Wagner, A., Rudraraju, R., Datla, S., Banerjee, A., Sudame, M., & Gray, J. (2012). Programming by voice: A hands-free approach for motorically challenged children. In CHI'12 Extended Abstracts on Human Factors in Computing Systems (pp. 2087-2092).

[41] "Annyang! Easily Add Speech Recognition to Your Site." Annyang, https://www.talater.com/annyang/. Accessed 30 Mar. 2021.

[42] The National Institutes of Health, American Sign Language, https://www.nidcd.nih.gov/health/american-sign-language, last accessed 2019/12/28

[43] Huenerfauth, M., & Hanson, V. (2009). Sign language in the interface: access for deaf signers. Universal Access Handbook. NJ: Erlbaum, 38.

[44] Traxler, C. B. (2000). The Stanford Achievement Test: National norming and performance standards for deaf and hard-of-hearing students. Journal of deaf studies and deaf education, 5(4), 337-348.

[45] Geers, A. E. (2006). Spoken Language in Children With Cochlear Implants.

[46] Deaf Kids Code Mission Homepage, https://www.deafkidscode.org/mission , last accessed 2019/12/28.

[47] Apple ASL Videos Webpage, https://developer.apple.com/asl-videos/, last accessed 2019/12/28

[48] MIT Scratch Homepage, https://scratch.mit.edu/, last accessed 2019/12/28

[49] ASL Day 2019: Everything You Need To Know About American Sign Language,https://www.newsweek.com/asl-day-2019-american-sign-language-1394695, last accessed 2019/12/28.

[50] Deafness and the User Experience, https://alistapart.com/article/deafnessandtheuserexperience/, last accessed 2019/12/28.

[51] Marghitu, D., Brahim, T. B., Weaver, J., & Rawajfih, Y. (2013, March). Auburn university robo camp K12 inclusive outreach program: a three-step model of effective introducing middle school students to computer programming and robotics. In Society for Information Technology & Teacher Education International Conference (pp. 58-63). Association for the Advancement of Computing in Education (AACE).

[52] Luckner, J., Bowen, S., & Carter, K. (2001). Visual teaching strategies for students who are deaf or hard of hearing. Teaching Exceptional Children, 33(3), 38-44.

[53] SIGCSE Special Projects Grants 2019 Webpage, https://sigcse.org/sigcse/programs/special/2019.html, last accessed 2019/12/28.

[54] Reis, J., Solovey, E. T., Henner, J., Johnson, K., & Hoffmeister, R. (2015, October). ASL CLeaR: STEM education tools for deaf students. In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (pp. 441-442).

[55] Beal-Alvarez, J. S., & Huston, S. G. (2014). Emerging evidence for instructional practice: Repeated viewings of sign language models. Communication Disorders Quarterly, 35(2), 93-102.

[56] MIT Scratch Ideas Homepage, https://scratch.mit.edu/ideas, last accessed 2019/12/28.

[57] Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. Journal of Teacher Education, 69(4), 386-407.

[58] Patterson-McNeill, H., & Binkerd, C. L. (2001, March). Resources for using lego mindstorms. In Proceedings of the twelfth annual CCSC South Central conference on The journal of computing in small colleges (pp. 48-55).

[59] Sutherland, C. J., & MacDonald, B. A. (2018, June). NaoBlocks: A Case Study of Developing a Children's Robot Programming Environment. In 2018 15th International Conference on Ubiquitous Robots (UR) (pp. 431-436). IEEE.

[60] Lauwers, T., & Nourbakhsh, I. (2010, July). Designing the finch: Creating a robot aligned to computer science concepts. In First AAAI Symposium on Educational Advances in Artificial Intelligence.

[61] Yeratziotis, A., & Zaphiris, P. (2018). A heuristic evaluation for deaf web user experience (HE4DWUX). International Journal of Human–Computer Interaction, 34(3), 195-217.

[62] Ten Usability Heuristics, https://www.nngroup.com/articles/ten-usability-heuristics/, last accessed 2019/12/28.

[63] https://aslcore.org/computerscience/

[64] ASLCORE CS FAQ Webpage, https://aslcore.org/faq/, last accessed 2019/12/28.

[65] Wikipedia Tooltip Webpage, https://en.wikipedia.org/wiki/Tooltip, last accessed 2019/12/28.

[66] Petrie, H., Fisher, W., Weimann, K., & Weber, G. (2004, April). Augmenting icons for deaf computer users. In CHI'04 Extended Abstracts on Human Factors in Computing Systems (pp. 1131-1134).

[67] Lacina, J., & Mathews, S. (2012). Using online storybooks to build comprehension. Childhood Education, 88(3), 155-161.

[68] Kourbetis, V., Boukouras, K., & Gelastopoulou, M. (2016, July). Multimodal accessibility for deaf students using interactive video, digital repository and hybrid books. In International Conference on Universal Access in Human-Computer Interaction (pp. 93-102). Springer, Cham

[69] Marschark, M., Paivio, A., Spencer, L. J., Durkin, A., Borgna, G., Convertino, C., & Machmer, E. (2017). Don't assume deaf students are visual learners. Journal of developmental and physical disabilities, 29(1), 153-171.

[70] AlShammari, A., Alsumait, A., & Faisal, M. (2018, November). Building an interactive e-learning tool for deaf children: interaction design process framework. In 2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e) (pp. 85-90). IEEE.

Appendices

AUBURN UNIVERSITY INSTITUTIONAL REVIEW BOARD for RESEARCH INVOLVING HUMAN SUBJECTS

RESEARCH PROTOCOL REVIEW FORM
FULL BOARD or EXPEDITED

For Information or help contact **THE OFFICE OF RESEARCH COMPLIANCE (ORC),** 115 Ramsay Hall, Auburn University
Phone: 334-844-5966 **e-mail:** IRBAdmin@auburn.edu **Web Address:** http://www.auburn.edu/research/vpr/ohs/index.htm

Revised 5.19.2020 **Submit completed form to IRBsubmit@auburn.edu or 115 Ramsay Hall, Auburn University 36849.**

***Complete this form using Adobe Acrobat** Writer **(versions 5.0 and greater).** Hand written copies not accepted.*

1. PROPOSED START DATE of STUDY: 11/15/2020 **Today's Date:** 10/26/2020

PROPOSED REVIEW CATEGORY (Check one): ☐ **FULL BOARD** ☒ **EXPEDITED**

SUBMISSION STATUS (Check one): ☐ **NEW** ☒ **REVISIONS (to address IRB Review Comments)**

2. PROJECT TITLE: Accessible Block-Based Programming for Blind and Low Vision Students.

3. Meenakshi Das	Graduate Research Assistant	Computer Science	mzd0107@auburn.edu
PRINCIPAL INVESTIGATOR	**TITLE**	**DEPT**	**AU E-MAIL**
215 S Gay Street Apt 107 Auburn AL 36830		6623128045	meenakshidas1112@gmail.com
MAILING ADDRESS		**PHONE**	**ALTERNATE E-MAIL**

4. FUNDING SUPPORT: ☐ N/A ☐ Internal ☒ External Agency: National Science Foundation ☐ Pending ☒ Received

For federal funding, list agency and grant number (if available). NSF Grant #18420922.

5a. List any contractors, sub-contractors, other entities associated with this project:

b. List any other IRBs associated with this project (including Reviewed, Deferred, Determination, etc.):
Auburn IRB Protocol #18-296, Georgia Tech IRB Protocol #H18260

PROTOCOL PACKET CHECKLIST

All protocols must include the following items:

- ☒ **Research Protocol Review Form** (All signatures included and all sections completed)
  (Examples of appended documents are found on the OHSR website: http://www.auburn.edu/research/vpr/ohs/sample.htm)
- ☒ **CITI Training Certificates** for all Key Personnel.
- ☒ **Consent Form or Information Letter** and any Releases (audio, video or photo) that the participant will sign.
- ☒ **Appendix A**, "Reference List"
- ☒ **Appendix B** if e-mails, flyers, advertisements, generalized announcements or scripts, etc., are used to recruit participants.
- ☒ **Appendix C** if data collection sheets, surveys, tests, other recording instruments, interview scripts, etc. will be used for data collection. Be sure to attach them in the order in which they are listed in # 13c.
- ☐ **Appendix D** if you will be using a debriefing form or include emergency plans/procedures and medical referral lists (A referral list may be attached to the consent document).
- ☐ **Appendix E** if research is being conducted at sites other than Auburn University or in cooperation with other entities. A **permission letter** from the site / program director must be included indicating their cooperation or involvement in the project. NOTE: If the proposed research is a multi-site project, involving investigators or participants at other academic institutions, hospitals or private research organizations, a letter of **IRB approval** from each entity is required prior to initiating the project.
- ☐ **Appendix F** - Written evidence of acceptance by the host country if research is conducted outside the United States.

The Auburn University Institutional Review Board has approved this Document for use from
10/20/2020 to ---------------
Protocol # 20-461 EP 2010

Version Date (date document created): 09/18/2020

COMPUTERSCIENCE
AND
SOFTWARE ENGINEERING

AUBURN UNIVERSITY

SAMUELGINN COLLEGE
OFENGINEERING

(NOTE: DO NOT AGREE TO PARTICIPATE UNLESS AN APPROVAL STAMP WITH CURRENT DATES HAS BEEN APPLIED TO THIS DOCUMENT.)

INFORMED ASSENT
for a Research Study entitled
"Accessible Block-Based Programming for Blind and Low Vision Students."

You are invited to participate in a research study to evaluate a solution for Blind/Low Vision individuals to learn block-based computer programming. The study is being conducted by Meenakshi Das, a graduate student, under the direction of Dr. Daniela Marghitu, a faculty member in the Auburn University Department of Computer Science.

What will be involved if you participate? If you decide to participate in this study, you will be asked to perform 10-15 computer programming tasks by interacting with our platform. A few days before the study, you will be given an instruction sheet to familiarize yourself with the Keyboard commands of the platform. The study will take around 45 minutes and will be conducted over Zoom. We will answer any questions, while simultaneously noting down our observations and your feedback while you perform the programming tasks. In the last 10 min, we will ask you for overall feedback of our solution.

Are there any risks or discomforts? The risks associated with participating in this study are minimal. It is fully on-line over zoom video conferencing platform. The meeting and computer screen will be recorded. You will need access to a computer and a stable Internet connection. Any risks, discomforts, and concerns will be addressed immediately by the instructor and faculty advisor. You may stop at any time during the study. Our observations and your feedback are confidential.

Are there any benefits to you or others? If you participate in this study, you can learn the basics of navigating block-based programming as a blind/low vision individual. In addition, you can help us better our solution by giving feedback and observational data. We/I cannot promise you that you will receive any or all the benefits described.

Will you receive compensation for participating? You will be given an Amazon Gift card of 10$ for your participation.

Are there any costs? The research study is completely free of cost.

If you change your mind about your participation, you can be withdrawn from the study at any time. Your decision about whether to participate or to stop participating will not jeopardize your relations with Auburn University or the Department of Educational Foundations, Leadership and Technology, the Department of Computer Science and Software Engineering.

Participants Initial ____________________

The Auburn University Institutional Review Board has approved this Document for use from
__10/20/2020__ to __----------------__
Protocol # ____20-461 EP 2010___

Your privacy will be protected, Any information obtained in connection with this study will be confidential. Information obtained through his/her participation may be published in a professional journal, conference paper or thesis.

If you have questions about this study, you are encouraged to ask questions at any time. Feel free to contact Meenakshi Das at mzd0107@auburn.edu or Dr.Daniela Marghitu at marghda@auburn.edu. A copy of this document will be given to you to keep.

If you have questions about your rights as a research participant, you may contact the Auburn University Office of Research Compliance or the Institutional Review Board by phone (334)-844-5966 or e-mail at IRBadmin@auburn.edu or IRBChair@auburn.edu.

HAVING READ THE INFORMATION PROVIDED, YOU MUST DECIDE WHETHER OR NOT YOU WISH TO PARTICIPATE IN THIS RESEARCH STUDY. YOUR SIGNATURE INDICATES YOUR WILLINGNESS TO PARTICIPATE.

Participants Signature	Printed Name	Date
Investigator obtaining consent	Printed Name	Date

The Auburn University Institutional Review Board has approved this Document for use from __10/20/2020__ to __---------------__
Protocol # __20-461 EP 2010__

COMPUTERSCIENCE
AND
SOFTWARE ENGINEERING

AUBURN UNIVERSITY

SAMUELGINN COLLEGE
OFENGINEERING

(NOTE: DO NOT AGREE TO PARTICIPATE UNLESS AN APPROVAL STAMP WITH CURRENT DATES HAS BEEN APPLIED TO THIS DOCUMENT.)

PARENTAL PERMISSION/MINOR ASSENT
for a Research Study entitled
"Accessible Block-Based Programming for Blind and Low Vision Students."

Your child is invited to participate in a research study to evaluate a solution for Blind/Low Vision individuals to learn block-based computer programming. The study is being conducted by Meenakshi Das, a graduate student, under the direction of Dr. Daniela Marghitu, a faculty member in the Auburn University Department of Computer Science. Since he/she is age 18 or younger we must have your permission to include him/her in the study.

What will be involved if your child participates? If you decide to allow your child to participate in this study, your child will be asked to perform 10-15 computer programming tasks by interacting with our platform. A few days before the study, your child will be given an instruction sheet to familiarize themselves with the Keyboard commands of the platform. The study will take around 45 minutes and will be conducted over Zoom. We will answer any questions, while simultaneously noting down our observations and your child's feedback while they perform the programming tasks. In the last 10 min, we will ask them for overall feedback of our solution.

Are there any risks or discomforts? The risks associated with participating in this study are minimal. It is fully on-line over zoom video conferencing platform. The meeting and computer screen will be recorded. Your child will need access to a computer and a stable Internet connection. Any risks, discomforts, and concerns will be addressed immediately by the instructor and faculty advisor. Your child may stop at any time during the study. Our observations and their feedback are confidential.

Are there any benefits to your son/daughter or others? If your child participates in this study, he/she can learn the basics of navigating block-based programming as a blind/low vision individual. In addition, they can help us better our solution by giving feedback and observational data. We/I cannot promise you that he/she will receive any or all the benefits described.

Will you or your son/daughter receive compensation for participating? Your child will given an Amazon Gift card of 10$ for their participation.

Are there any costs? The research study is completely free of cost.

If you (or your son/daughter) change your mind about his/her participation, he/she can be withdrawn from the study at any time. Your decision about whether to allow your son/daughter to participate or to stop participating will not jeopardize your or his/her future relations with Auburn University or the Department of Educational Foundations, Leadership and Technology, the Department of Computer Science and Software Engineering.

Parent/Guardian Initials____________
Participants Initials______________

The Auburn University Institutional Review Board has approved this Document for use from
10/20/2020 to ---------------
Protocol # 20-461 EP 2010

Your son's/daughter's privacy will be protected. Any information obtained in connection with this study will be confidential. Information obtained through his/her participation may be published in a professional journal, conference paper or thesis.

If you (or your son/daughter) have questions about this study, you are encouraged to ask questions at any time. Feel free to contact Meenakshi Das at mzd0107@auburn.edu or Dr.Daniela Marghitu at marghda@auburn.edu. A copy of this document will be given to you to keep.

If you have questions about your son's/daughter's rights as a research participant, you may contact the Auburn University Office of Research Compliance or the Institutional Review Board by phone (334)-844-5966 or e-mail at IRBadmin@auburn.edu or IRBChair@auburn.edu.

HAVING READ THE INFORMATION PROVIDED, YOU MUST DECIDE WHETHER OR NOT YOU WISH FOR YOUR SON OR DAUGHTER TO PARTICIPATE IN THIS RESEARCH STUDY. YOUR SIGNATURE INDICATES YOUR WILLINGNESS TO ALLOW HIM OR HER TO PARTICIPATE.

Parent/Guardian Signature	Printed Name	Date
Investigator obtaining consent	Printed Name	Date
Minor Signature	Printed Name	Date

The Auburn University Institutional Review Board has approved this Document for use from 10/20/2020 to ---------------
Protocol # 20-461 EP 2010

VIDEO RELEASE - MINOR

During your child's participation in this research study, "Accessible Block-Based Programming for Blind and Low Vision Students", the online meeting, including the computer screen while your child performs the tasks, will be recorded. Your signature on the Informed Consent gives us permission to do so.

Your signature on this document gives us permission to use the videotape(s) for the additional purposes of reports to project funders, presentations, publications, beyond the immediate needs of this study. These videotapes will not be destroyed at the end of this research but will be retained until June 2021.

In addition, the following persons or groups will have access to the tapes:
Dr. Daniela Marghitu, Department of Computer Science and Software Engineering, Faculty & Director Education and Assistive Technology Laboratory

Your permission:

I give my permission for videotapes produced in the study , "Accessible Block-Based Programming for Blind and Low Vision Students", which contain images of my child, to be used for the purposes listed above, and to also be retained until June 2021.

______________________	______________________
Parent/Guardian's Signature Date	Investigator's Signature Date
______________________	______________________
Parent/Guardian's Printed Name	Investigator's Printed Name

____
Minors Printed Name.

The Auburn University Institutional Review Board has approved this Document for use from __10/20/2020__ to __---------------__
Protocol # __20-461 EP 2010__

____
Minor's Signature /Date

Version Date (date document created): ______________

VIDEO RELEASE

During your participation in this research study, “Accessible Block-Based Programming for Blind and Low Vision Students”, the online meeting, including your computer screen while you perform the tasks, will be recorded. Your signature on the Informed Consent gives us permission to do so.

Your signature on this document gives us permission to use the videotape(s) for the additional purposes of reports to project funders,
presentations, publications, beyond the immediate needs of this study. These videotapes will not be destroyed at the end of this research but will be retained until June 2021.

In addition, the following persons or groups will have access to the tapes:
Dr. Daniela Marghitu, Department of Computer Science and Software Engineering, Faculty & Director Education and Assistive Technology Laboratory

Your permission:

I give my permission for videotapes produced in the study, “Accessible Block-Based Programming for Blind and Low Vision Students” to be used for the purposes listed above, and to also be retained until June 2021.

____________________	____________________
Participant’s Signature Date	Investigator’s Signature Date
____________________	____________________
Participant’s Printed Name	Investigator’s Printed Name

Version Date (date document created): 09/21/2020

Appendix B

E-MAIL INVITATION FOR A SOLUTION EVALUATION FOR SCHOOL ADMINISTRATORS AND EDUCATORS IN ALABAMA TO RECRUIT TWO BLIND MIDDLE/HIGH SCHOOL STUDENTS.

Our Research Team at Auburn University has designed a solution to make Block-based programming accessible to individuals who are blind or low vision. **We are looking for your help to recruit two blind individuals who have little to no experience with computer programming (middle or high school students), to participate in a study.**
Aa participant, they will be asked to perform 10-15 computer programming tasks by interacting with our platform. A few days before the study, they will be given an instruction sheet to familiarize themselves with the Keyboard commands of the platform. The study will take around 45 minutes and will be conducted over Zoom. We will answer any questions, while simultaneously noting down our observations and their feedback while they perform the programming tasks. In the last 10 min, we will ask them for overall feedback of our solution. The study is completely confidential. They will be compensated with an Amazon Gift Card of 10$ for their time.

Please feel free to send it to students who might be interested. They can contact us at Meenakshi Das at *mzd0107@auburn.edu* or Dr.Daniela Marghitu at *marghda@auburn.edu.*
Thank you for your consideration.

E-MAIL INVITATION FOR A SOLUTION EVALUATION TO RECRUIT EXPERIENCED PROGRAMMERS VIA ACCESSCOMPUTING LIST OR DIRECT EMAIL TO COLLEAGUES.

Our Research Team at Auburn University has designed a solution to make Block-based programming accessible to individuals who are blind or low vision. **We are looking for two experienced blind programmers (individuals who have or are majoring in computer science and/or working professionally as a software developer), to participate in the study.**
As a participant, you will be asked to perform 10-15 computer programming tasks by interacting with our platform. A few days before the study, you will be given an instruction sheet to familiarize yourself with the Keyboard commands of the platform. The study will take around 45 minutes and will be conducted over Zoom. We will answer any questions, while simultaneously noting down our observations and your feedback while you perform the programming tasks. In the last 10 min, we will ask you for overall feedback of our solution. The study is completely confidential. You will be compensated with an Amazon Gift Card of 10$ for your time.

If you would like to participate in this research study or have questions, please contact Meenakshi Das at *mzd0107@auburn.edu* or Dr.Daniela Marghitu at *marghda@auburn.edu.* If you know someone who might be interested, please feel free to forward this message to them.
Thank you for your consideration.

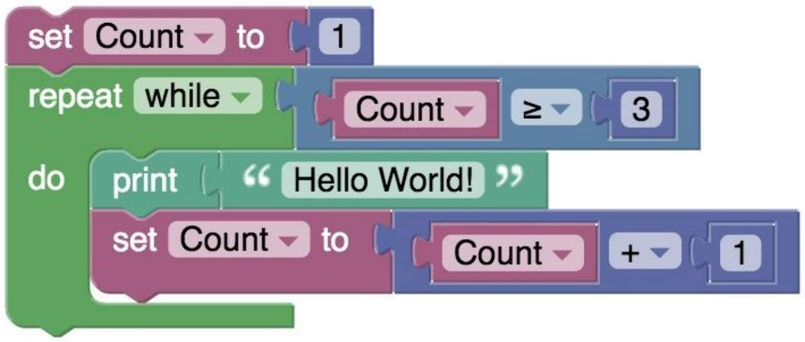
Appendix C
Data Collection Sheet.

Participants do the following tasks on our web platform. We record our observations and their feedback for each task. Tasks may be slightly modified on the day of study.

Tasks for the 2 experienced blind programmers:

Do the following using the Keyboard and listen to audio output:

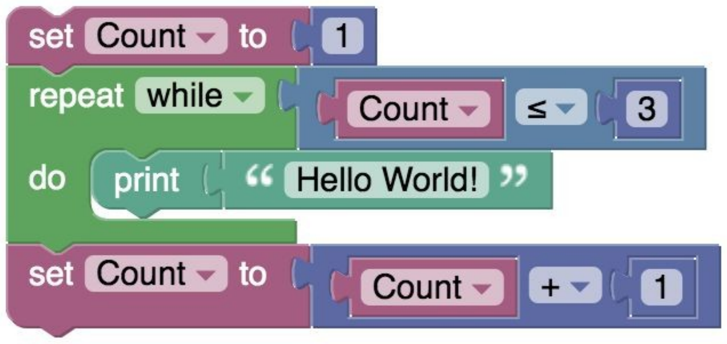
1. **Task:** Enter the toolbox.
   Observation: Were they able to successfully enter the toolbox?
   Feedback:
2. **Task:** Navigate through all blocks in the toolbox.
   Observation: Were they able to successfully navigate the toolbox?
   Feedback:
3. **Task:** Select 'create variable' block from toolbox, name it 'test' and set it's value to 3.
   Observation: Were they able to successfully create the variable?
   Feedback:
4. **Task:** Insert the 'if-do' block into the workspace.
   Observation: Were they able to successfully insert the block into the workspace?
   Feedback:
5. **Task:** Navigate through all of the 'if-block' connections
   Observation: Were they able to successfully navigate through all connections?
   Feedback:
6. **Task:** Produce the following code: Write code using the 'if-do' which print's "Hello world" if the value of variable 'test' is equal to '3'
   Observation: Were they able to successfully produce the code? Any issues they ran into?
   Feedback:
7. **Task:** Produce the following code: Write code using the 'repeat blank times' block which prints 'Hello world' until the value of a variable named 'apple' reaches 10.
   Observation: Were they able to successfully produce the code? Any issues they ran into?
   Feedback:
8. **Task:** The following code is supposed to print 'Hello world' until the value of 'count' reaches 3. However, there is a bug on one line. Can you identify the bug and fix it?



```
set Count to 1
repeat while Count ≥ 3
do  print "Hello World!"
    set Count to Count + 1
```

Observation: Were they able to successfully debug the code? Any issues they ran into?
Feedback:

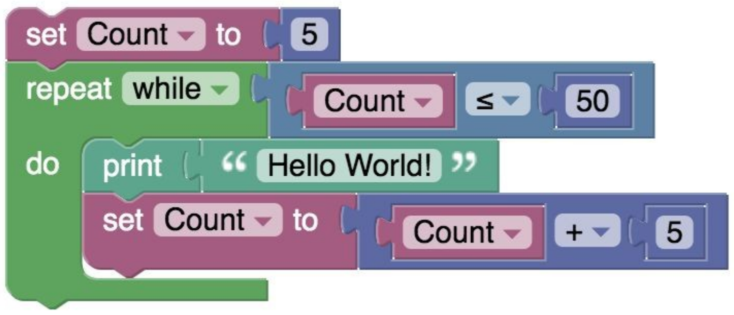
9. **Task:** The following code is supposed to print 'Hello world' until the value of 'count' reaches 3. However, there is a bug on one line. Can you identify the bug and fix it?



```
set Count to 1
repeat while Count ≤ 3
do  print "Hello World!"
set Count to Count + 1
```

Observation: Were they able to successfully debug the code? Any issues did you run into?
Feedback:

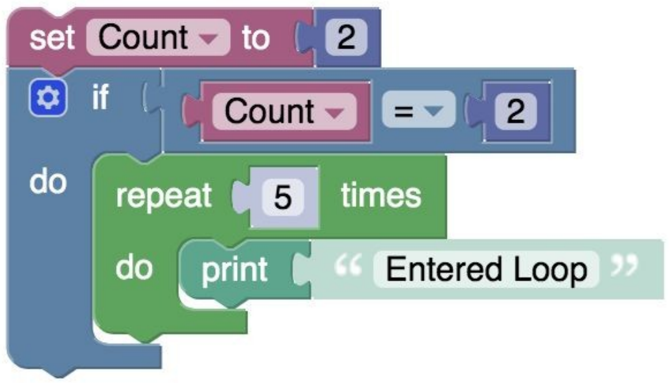
10. **Task:** Translate this block-based code into a programming language of your choice. Type it on the Google Doc shared.



```
set Count to 5
repeat while Count ≤ 50
do  print "Hello World!"
    set Count to Count + 5
```

Observation: Were they able to successfully translate the code? Any issues they ran into? Were they able to identify the nested statements?
Feedback:

11. **Task:** What is the output of the following code?



```
set Count to 2
if Count = 2
do  repeat 5 times
    do  print "Entered Loop"
```

Observation: Were they able to successfully get the output of the code? Any issues did you run into?
Feedback:

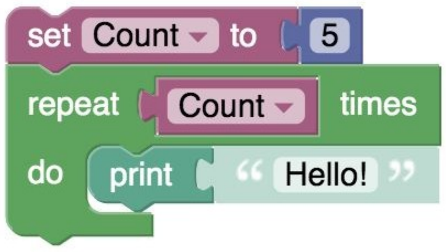
Tasks for the 2 low-experienced blind programmers(middle-high schoolers):

Do the following using the Keyboard and listen to audio output:

1. **Task:** Enter the toolbox.
   Observation: Were they able to successfully enter the toolbox?
   Feedback:
2. **Task:** Navigate through all blocks in the toolbox.
   Observation: Were they able to successfully navigate the toolbox?
   Feedback:

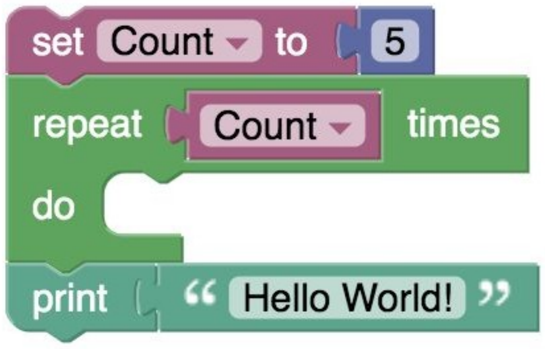
3. **Task:** Select ‘create variable’ block from toolbox. Use the Keyboard Command to read its tooltip.
   Observation: Were they able to successfully find and understand the tooltip? Was the tooltip helpful?
   Feedback:
4. **Task:** Name the variable ‘test’ and set it’s value to 3.
   Observation: Were they able to successfully create the variable?
   Feedback:
5. **Task:** Insert the ‘if-do’ block into the workspace.
   Observation: Were they able to successfully insert the block into the workspace?
   Feedback:
6. **Task:** Find the ‘if’ connection in the ‘if-do’ block. Then attach the ‘is equal to’ block to it.
   Observation: Were they successfully able to find the ‘if’ connection and attach the ‘is equal to’ block to it?
   Feedback:
7. **Task:** Find the first input connection in the ‘is equal to block’. Then insert the variable ‘test’ into it. Find the second input connection in the ‘is equal to block’. Then insert the number ‘3’ into it.
   Observation: Were they successfully able to find the two input connections and insert the respective blocks into it?
   Feedback:
8. **Task:** Find the ‘do’ connection of the ‘if block’. Insert the ‘print’ block into it. Find the ‘right’ connection of the print block and insert the ‘string’ block. Type ‘Hello world’ into the ‘string’ block.
   Observation: Were they successfully able to find the do connection and insert the print block and the string block?
   Feedback:
9. **Task**: Find the ‘bottom’ connection of the ‘if-do’ block. Attach the ‘repeat blank times’ block to it.
   Observation: Were they successfully able to find the bottom connection and insert the repeat block?
   Feedback:
10. **Task**: Find the ‘blank times’ input connection of the ‘repeat times’ block. Insert any number you want into it.
    Observation: Were they successfully able to find the input connection and insert the number block?
    Feedback:
11. **Task:** Find the ‘do’ connection of the ‘repeat times block’. Insert the ‘print’ block into it. Find the ‘right’ connection of the print block and insert the ‘string’ block. Type ‘Hello world’ into the block.
    Observation: Were they successfully able to find the do connection and insert the print block and the string block?
    Feedback:

12. **Task:** Go to the top and navigate through the entire program. Explain in your words what the program does.
Observation: Were they able to successfully understand the code?
Feedback:
13. **Task:** What is the output of the following program?
Observation: Were they able to successfully get the output of the code? Any issues they ran into? Were they able to identify the nested statement?
Feedback:



```
set Count to 5
repeat Count times
do  print "Hello!"
```

14. **Task:** The following code is supposed to print 'Hello world' five times.. However, there is a bug on one line. Can you identify the bug and fix it?
Observation: Were they able to successfully debug the code? Any issues they ran into?
Feedback:



```
set Count to 5
repeat Count times
do
print "Hello World!"
```

AUBURN UNIVERSITY INSTITUTIONAL REVIEW BOARD for RESEARCH INVOLVING HUMAN SUBJECTS

RESEARCH PROTOCOL REVIEW FORM
FULL BOARD or EXPEDITED

For Information or help contact **THE OFFICE OF RESEARCH COMPLIANCE (ORC)**
Phone: 334-844-5966 **e-mail:** IRBAdmin@auburn.edu **Web Address:** http://www.auburn.edu/research/vpr/ohs/index.htm

Revised 10.08.2020 **Submit completed form to IRBsubmit@auburn.edu**

***Complete this form using Adobe Acrobat** Writer **(versions 5.0 and greater).** Hand written copies not accepted.*

1. PROPOSED START DATE of STUDY: 11/15/2020 **Today's Date:** 10/26/2020

PROPOSED REVIEW CATEGORY (Check one): ☐ **FULL BOARD** ▣ **EXPEDITED**

SUBMISSION STATUS (Check one): ☐ **NEW** ▣ **REVISIONS (to address IRB Review Comments)**

2. PROJECT TITLE: Accessible Block-Based Programming for Blind and Low Vision Students.

3.

Meenakshi Das	Graduate Research Assistant	Computer Science	mzd0107@auburn.edu
PRINCIPAL INVESTIGATOR	**TITLE**	**DEPT**	**AU E-MAIL**
215 S Gay Street Apt 107 Auburn AL 36830		6623128045	meenakshidas1112@gmail.com
MAILING ADDRESS		**PHONE**	**ALTERNATE E-MAIL**

4. FUNDING SUPPORT: ☐ N/A ☐ Internal ▣ External Agency: National Science Foundation ☐ Pending ▣ Received

For federal funding, list agency and grant number (if available). NSF Grant #18420922.

5a. List any contractors, sub-contractors, other entities associated with this project:

b. List any other IRBs associated with this project (including Reviewed, Deferred, Determination, etc.):
Auburn IRB Protocol #18-296, Georgia Tech IRB Protocol #H18260

PROTOCOL PACKET CHECKLIST

All protocols must include the following items:

- ☐ **Research Protocol Review Form** (All signatures included and all sections completed)
  (Examples of appended documents are found on the OHSR website: http://www.auburn.edu/research/vpr/ohs/sample.htm)
- ☐ **CITI Training Certificates** for all Key Personnel.
- ☐ **Consent Form or Information Letter** and any Releases (audio, video or photo) that the participant will sign.
- ☐ **Appendix A**, "Reference List"
- ☐ **Appendix B** if e-mails, flyers, advertisements, generalized announcements or scripts, etc., are used to recruit participants.
- ☐ **Appendix C** if data collection sheets, surveys, tests, other recording instruments, interview scripts, etc. will be used for data collection. Be sure to attach them in the order in which they are listed in # 13c.
- ☐ **Appendix D** if you will be using a debriefing form or include emergency plans/procedures and medical referral lists (A referral list may be attached to the consent document).
- ☐ **Appendix E** if research is being conducted at sites other than Auburn University or in cooperation with other entities. A **permission letter** from the site / program director must be included indicating their cooperation or involvement in the project. NOTE: If the proposed research is a multi-site project, involving investigators or participants at other academic institutions, hospitals or private research organizations, a letter of **IRB approval** from each entity is required prior to initiating the project.
- ☐ **Appendix F** - Written evidence of acceptance by the host country if research is conducted outside the United States.

The Auburn University Institutional Review Board has approved this Document for use from 02/27/2021 to -------- Protocol # 21-112 EP 2102

Version Date (date document created): 09/18/2020

COMPUTERSCIENCE AND SOFT
WARE ENGINEERING

AUBURN UNIVERSITY

SAMUELGINN COLLEGE
OFENGINEERING

(NOTE: DO NOT AGREE TO PARTICIPATE UNLESS AN APPROVAL STAMP WITH CURRENT DATES HAS BEEN APPLIED TO THIS DOCUMENT.)

INFORMED CONSENT
for a Research Study entitled
"Accessible Computer Science for Deaf K-12 Students"

You are invited to participate in a research study to evaluate computer science concept videos in American Sign Language(ASL). The study is being conducted by Meenakshi Das, a graduate student, under the direction of Dr. Daniela Marghitu, a faculty member in the Auburn University Department of Computer Science.

What will be involved if you participate? The study will involve you watching eight computer science concept videos being delivered via ASL. The study will be completed over Microsoft Teams where you watch the videos and fill out written surveys. The study will take one hour, and we will answer any questions or troubleshoot technical difficulties over Microsoft Teams.

Are there any risks or discomforts? The risks associated with participating in this study are minimal. It is fully on-line over Microsoft Teams conferencing platform. You will need access to a computer and a stable Internet connection. Any risks, discomforts, and concerns will be addressed immediately by the instructor and faculty advisor. You may stop at any time during the study. The study is confidential.

Are there any benefits to you or others? If you participate in this study, you can learn the basics of computer science delivered via ASL. In addition, you can help us better our videos by giving survey and interview data. We cannot promise you that you will receive any or all the benefits described.

Will you receive compensation for participating? You will be given an Amazon Gift card of 25$ for your participation.

Are there any costs? The research study is completely free of cost.

If you change your mind about your participation, you can be withdrawn from the study at any time. Your decision about whether to participate or to stop participating will not jeopardize your relations with Auburn University or the Department of Educational Foundations, Leadership and Technology, the Department of Computer Science and Software Engineering.

Participants Initial ____________________

The Auburn University Institutional Review Board has approved this Document for use from
02/27/2021 to --------
Protocol # 21-112 EP 2102

Your privacy will be protected, Any information obtained in connection with this study will be confidential. Information obtained through your participation may be published in a professional journal, conference paper or thesis.

If you have questions about this study, you are encouraged to ask questions at any time. Feel free to contact Meenakshi Das at mzd0107@auburn.edu or Dr.Daniela Marghitu at marghda@auburn.edu. A copy of this document will be given to you to keep.

If you have questions about your rights as a research participant, you may contact the Auburn University Office of Research Compliance or the Institutional Review Board by phone (334)-844-5966 or e-mail at IRBadmin@auburn.edu or IRBChair@auburn.edu.

HAVING READ THE INFORMATION PROVIDED, YOU MUST DECIDE WHETHER OR NOT YOU WISH TO PARTICIPATE IN THIS RESEARCH STUDY. YOUR SIGNATURE INDICATES YOUR WILLINGNESS TO PARTICIPATE.

Participants Signature	Printed Name	Date
Investigator obtaining consent	Printed Name	Date

COMPUTERSCIENCE AND SOFT
WARE ENGINEERING

AUBURN UNIVERSITY

SAMUELGINN COLLEGE
OFENGINEERING

(NOTE: DO NOT AGREE TO PARTICIPATE UNLESS AN APPROVAL STAMP WITH CURRENT DATES HAS BEEN APPLIED TO THIS DOCUMENT.)

PARENTAL PERMISSION/MINOR CONSENT
for a Research Study entitled
"Accessible Computer Science for Deaf K-12 Students."

Your child is invited to participate in a research study to evaluate computer science concept videos in American Sign Language(ASL). The study is being conducted by Meenakshi Das, a graduate student, under the direction of Dr. Daniela Marghitu, a faculty member in the Auburn University Department of Computer Science. Since he/she is age 18 or younger we must have your permission to include him/her in the study.

What will be involved if your child participates? The study will involve your child watching eight computer science concept videos being delivered via ASL. The study will be completed over Microsoft Teams where they will watch the videos and fill out written surveys. The study will take one hour, and we will answer any questions or troubleshoot technical difficulties over Microsoft Teams.

Are there any risks or discomforts? The risks associated with participating in this study are minimal. It is fully on-line over Microsoft Teams conferencing platform. Your child will need access to a computer and a stable Internet connection. Any risks, discomforts, and concerns will be addressed immediately by the instructor and faculty advisor. They may stop at any time during the study. The study is confidential.

Are there any benefits to your son/daughter or others? If your child participates in this study, they can learn the basics of computer science delivered via ASL. In addition, they can help us better our videos by giving survey and interview data. We cannot promise you that your child will receive any or all the benefits described.

Will you or your son/daughter receive compensation for participating? Your child will be given an Amazon Gift card of 25$ for their participation.

Are there any costs? The research study is completely free of cost.

If you (or your son/daughter) change your mind about his/her participation, he/she can be withdrawn from the study at any time. Your decision about whether to allow your son/daughter to participate or to stop participating will not jeopardize your or his/her future relations with Auburn University or the Department of Educational Foundations, Leadership and Technology, the Department of Computer Science and Software Engineering.

Parent/Guardian Initial -

Participant Initial -

Your son's/daughter's privacy will be protected. Any information obtained in connection with this study will be confidential. Information obtained through his/her participation may be published in a professional journal, conference paper or thesis.

If you (or your son/daughter) have questions about this study, you are encouraged to ask questions at any time. Feel free to contact Meenakshi Das at mzd0107@auburn.edu or Dr.Daniela Marghitu at marghda@auburn.edu. A copy of this document will be given to you to keep.

If you have questions about your son's/daughter's rights as a research participant, you may contact the Auburn University Office of Research Compliance or the Institutional Review Board by phone (334)-8445966 or e-mail at IRBadmin@auburn.edu or IRBChair@auburn.edu.

HAVING READ THE INFORMATION PROVIDED, YOU MUST DECIDE WHETHER OR NOT YOU WISH FOR YOUR SON OR DAUGHTER TO PARTICIPATE IN THIS RESEARCH STUDY. YOUR SIGNATURE INDICATES YOUR WILLINGNESS TO ALLOW HIM OR HER TO PARTICIPATE.

Parent/Guardian Signature	Printed Name	Date
Investigator obtaining consent	Printed Name	Date
Minor Signature	Printed Name	Date

Appendix B
E-MAIL INVITATION TO EVALUATE COMPUTER SCIENCE CONCEPT VIDEOS IN AMERICAN SIGN LANGUAGE(ASL)

Our Research Team at Auburn University has developed computer science concept videos in ASL in partnership with the non-profit organization Deaf Kids Code . We invite you to participate in evaluating the videos. The study is completely confidential and will involve you watching eight computer science concept videos being delivered via ASL. The study will be completed over Microsoft Teams where you will watch the videos and fill out written surveys. It will take about one hour, and we will answer any questions or troubleshoot technical difficulties over Microsoft Teams. You will be required to sign a consent form before the study. You will be compensated for 25$ for your time. Please contact us at Meenakshi Das at mzd0107@auburn.edu or Dr.Daniela Marghitu at marghda@auburn.edu. Thank you for your consideration.

Appendix C

Data collection - Middle or high schoolers with little to no experience with computer programming

1. **Pre-Survey**

Demographic Data:

- Name:

- How old are you today?
  a. 11 to 13 years
  b. 14 to 17 years
  c. 18 to 22 years
  d. 23 to 26 years
  e. 27 to 35 years
  f. over 35 years

- What is your gender?
  a. Male
  b. Female
  c. Other
  d. Non-binary
  e. Prefer Not to say

- Which category best fits your race?
  a. White
  b. Black or African American
  c. Asian
  d. Hispanic
  e. Other: Please enter:
  f. Prefer not to say

- What is your profession?

- Do you have a degree in computer science or a closely related field such as Electrical Engineering, Human Computer Interaction, Information Technology?
  a. Yes, I am currently pursuing or have a bachelor's degree
  b. Yes, I am currently pursuing or have a master’s degree
  c. Yes, I am currently pursuing or have a PhD.
  d. No, my degree is in ____________

- What is your level of experience with text-based programming such as with Python, C++ or Java etc.?
    a. I have never heard of it
    b. I have heard of it but have no experience.
    c. Minimal experience. Maybe compiled a test program.
    d. Some experience. Wrote one or two small programs.
    e. Substantial experience. Wrote several small to medium-sized programs.
    f. Extensive experience. Wrote many programs.

- What is your American Sign Language(ASL) proficiency?
    a. Level 0: Unable to function in the language.
    b. Level 1: Able to satisfy routine travel needs and minimum courtesy requirements.
    c. Level 2: Able to satisfy routine social demands and limited work requirements.
    d. Level 3: Able to sign ASL with sufficient structural accuracy and vocabulary to participate effectively in most formal and informal conversations pertaining to practical, social, and professional needs.
    e. Level 4: Able to use the language fluently and accurately on all levels pertaining to professional needs.
    f. Level 5: Language proficiency equivalent to that of a sophisticated native signer.

- Please check all which apply to you:
    a. I use lip-reading.
    b. I use cochlear-implant.
    c. I use American Sign Language(ASL).
    d. I use hearing aids.

Respond to the following questions on the answer sheet, using the following scale: a) strongly agree b) agree, but with reservations c) neutral, neither agree nor disagree d) disagree, but with reservations e) strongly disagree

1. I am confident I can learn basic computer science concepts.
2. I like to use computers to solve problems.
3. Knowing how to use computers will help me with my future job.
4. I like thinking about using technology to do new things.
5. Programming (creating new programs or tools on computers) is hard.
6. Girls can do computer programming.
7. Boys can do computer programming.
8. People with disabilities can do computer programming.
9. My hearing disability hinders me from learning computer programming.
10. My disability makes me feel unwelcome in computer programming.
11. Computer jobs are boring.

12. I know more than my friends about computers.
13. I like the challenge of programming computers.
14. I am interested in a career that uses computers to do new things.

Please check boxes next to the items you are confident about:

- ☐ Understand Algorithms (1)
- ☐ Understand User Input and Output (2)
- ☐ Understand Programming Commands (3)
- ☐ Declare a variable (4)
- ☐ Create a conditional statement (5)
- ☐ Create a conditional statement with AND, OR (6)
- ☐ Create a loop (7)
- ☐ Debugging code (8)
- ☐ Understand Object-Oriented Programming (9).
- ☐ Create a function (9)

2. While watching videos: written responses:

During meeting, watch every computer science video and answer the following:

1. Explain in 3-4 sentences what you understand from it.
2. What was confusing to understand?

After watching all videos: written responses:

1. What did you learn today?
2. How did the videos make you feel?
3. Did the ASL videos help you learn better? Explain why yes or no.
4. Would you like to see more computer science videos in ASL?

3. Post-survey:

1. I am confident I can learn basic computer science concepts.
2. I like to use computers to solve problems.
3. Knowing how to use computers will help me with my future job
4. I like thinking about using technology to do new things
5. Programming (creating new programs or tools on computers) is hard.
6. Girls can do computer programming.
7. Boys can do computer programming.
8. People with disabilities can do computer programming.
9. My hearing disability hinders me from learning computer programming.
10. My disability makes me feel unwelcome in computer programming.
11. Computer jobs are boring.
12. I know more than my friends about computers.
13. I like the challenge of programming computers.
14. I am interested in a career that uses computers to do new things.

Data collection - Experienced Adult computer programmers:

1. **Pre-Survey:**

Demographic Data:

- Name:

- How old are you today?
    g. 11 to 13 years
    h. 14 to 17 years
    i. 18 to 22 years
    j. 23 to 26 years
    k. 27 to 35 years
    l. over 35 years

- What is your gender?
  f. Male
  g. Female
  h. Other
  i. Non-binary
  j. Prefer Not to say

- Which category best fits your race?
  g. White
  h. Black or African American
  i. Asian
  j. Hispanic
  k. Other: Please enter:
  l. Prefer not to say

- What is your profession?

- Do you have a degree in computer science or a closely related field such as Electrical Engineering, Human Computer Interaction, Information Technology?
  e. Yes, I am currently pursuing or have a bachelor's degree
  f. Yes, I am currently pursuing or have a master's degree
  g. Yes, I am currently pursuing or have a PhD.
  h. No, my degree is in ____________

- What is your level of experience with text-based programming such as with Python, C++ or Java etc.?
  g. I have never heard of it
  h. I have heard of it but have no experience.
  i. Minimal experience. Maybe compiled a test program.
  j. Some experience. Wrote one or two small programs.
  k. Substantial experience. Wrote several small to medium-sized programs.
  l. Extensive experience. Wrote many programs.

- What is your American Sign Language(ASL) proficiency?
  g. Level 0: Unable to function in the language.
  h. Level 1: Able to satisfy routine travel needs and minimum courtesy requirements.
  i. Level 2: Able to satisfy routine social demands and limited work requirements.
  j. Level 3: Able to sign ASL with sufficient structural accuracy and vocabulary to participate effectively in most formal and informal conversations pertaining to practical, social, and professional needs.
  k. Level 4: Able to use the language fluently and accurately on all levels pertaining to professional needs.

l. Level 5: Language proficiency equivalent to that of a sophisticated native signer.

- Please check all which apply to you:
    a. I use lip-reading.
    b. I use cochlear-implant.
    c. I use American Sign Language(ASL).
    d. I use hearing aids.

2. While watching videos: written responses:

For each video:

1. Explain in 3-4 sentences what you understand from it.
2. What improvements would you make to the video?
3. Were the captions accurate with the ASL?
4. Was the ASL accurate with regards to computer science concepts?
5. What was confusing to understand?

After watching all videos: written responses:

1. What did you learn today?
2. How did the videos make you feel?
3. How or how not did the ASL Videos help you?
4. Would you like to see more computer science videos in ASL?