Project Overview

The main goal of the completed work was to develop a nonvisual interface for the Scratch programming language in order to enable BVI students to learn Scratch programming independently alongside their sighted peers. A critical component of the approach was to translate key aspects of Scratch that increased engagement and lowered hurdles to programming for sighted students to the haptic domain, which is accessible by all BVI students. To do so, it was necessary to consider the significant differences between how haptics and vision process information to avoid adding new barriers due to these differences. Visual design techniques for low vision were also used in tandem to ensure access for BVIs who may still use their residual vision, as well as sighted students. Components of the system were developed with stakeholder feedback, including two full design iterations for the main components of the system with objective user testing involving BVIs.

A pilot test involving BVI middle school and high school students using the complete system was performed through multiple offerings of a 2-day camp for BVI students at VCU. Each camp simulated a classroom setting with 1-2 BVI students. A hands-on learning curriculum, consisting of lecture notes, accessible multi-media support materials and accessible quizzes, was created for this purpose. A subset of the Scratch code blocks were used, including most of the blocks from the Motion and Control categories, as well as some code blocks from the Events, Operators and Sensing categories. Audio-visual data of student behavior, as well as an exit survey, was collected and analyzed using qualitative content analysis methods. Participating BVI students did use the system as expected and were enthusiastic about the camp. Improvements for subsequent design iterations were noted.

A picture of a student work station with the developed system is shown in Figure 2 (left). The student using the system sits in the chair shown. The three tables surrounding the chair correspond to the three main panes in “virtual” Scratch. The code editor work surface, where the student assembles their code blocks, is located directly in front of them (and shown in more detail on the right side of Figure 2). To the student’s right is the block palette consisting of a storage drawer organizational system for the tangible code blocks. To the student’s left is the physical “stage” with a mobile robot sprite that executes the commands programmed. A camera underneath the code editor work surface is used to track ArUco markers placed on the backs of each of the tangible code pieces (Figure 4, c4); the results can be used to translate the built program into executable code for the mobile robot sprite. As with “virtual” Scratch, tinkering is facilitated by allowing users to leave code blocks and code fragments on the code editor worksurface: only complete code is executed. This and other consistencies with “virtual” Scratch, in addition to providing some of the same advantages as “virtual” Scratch, are also intended to facilitate the simultaneous teaching of BVI and sighted students by having both interfaces behave in a similar manner.

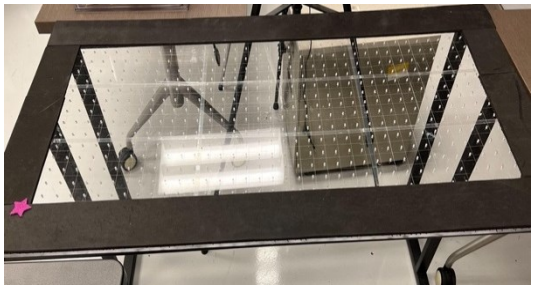

Figure 2: Left, Overview of system. Below, The code editor work surface.

Unlike the “virtual” code editor workspace in Scratch, the design of the physical workspace (Figure 2, right) for assembling code blocks had significant physical and perceptual limitations to consider. For ease of use, physically the workspace was limited to the reach of most middle school students (target age group); this, together with the desire to be able to construct moderately sized programs, helped define the size of the code blocks (Figure 4). Perceptually, the work surface needed additional design elements to facilitate when solely non-visual access was used due to the difficulties created by the much more limited field of view of touch as compared to vision. This made searching for and manipulating blocks on the

workspace more serial in nature and, thus, more time consuming and cognitively demanding. Our user studies found that an etched grid (with grid spacing equivalent to a standard code block size) improved performance and ease of use for typical tasks performed when learning programming; however, constraining the movement of the code blocks within the workspace (through the use of physical channels) did not. To address the surprisingly poorer performance in the center of the workspace for some tasks, raised segment lines (the 3x3 grid) were added as reference lines. The segmentation, together with the raised star, also created a high-level coordinate system that made it easier to communicate the location of code between individuals in the computer camps

Figure 3: Code blocks palette. Variables and constants were kept in a smaller drawer system (left) and all other blocks were organized by function in a larger drawer system (right).

A system of pull-out drawers, labelled with raised profiles of their top surfaces, was used to store the code blocks for easy retrieval (Figure 3). The blocks themselves were design to be accessible to blind, low vision and sighted users (Figure 4). Raised relief symbols (black against a light background for high contrast), representing each code block's function, were created in a size and shape to be easily accessible by touch and vision. Commands were represented by pictorial symbols (e.g., Figure 4a), typically with functional (e.g., Figure 4d) or mathematical (e.g., Figure 5) meaning. Numeric values were represented by both large text roman letters and raised braille characters (Figure 4b). Variables were represented by single raised roman letters (e.g., Figure c2). All blocks had a raised black dot in the upper left-hand corner so that the block can be oriented correctly for use.

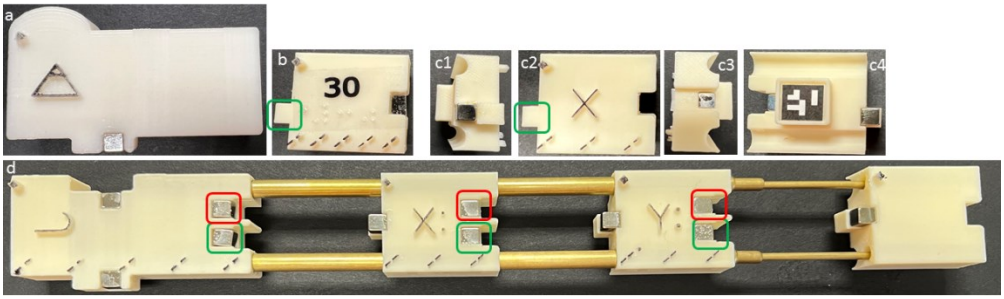

Figure 4: Top, example code blocks. Bottom, example code block assembly (glide in T sec to X,Y).

As in "virtual" Scratch, code blocks were designed to only be able to connect in ways that made syntactic sense. For example, the start event block (Figure 4a) can only have command blocks connected below the event block and only through matching the tab with a notch (e.g., Figure 4d); the only blocks with a matching notch are the command blocks, which can then be sequenced together to form a program. In the same vein, connections were also used to define syntax for parameters and conditional expressions, as in "virtual" Scratch. In those cases, (Figure 4 bottom and Figure 5), the location of a connection(s) on the left hand side of a parameter slot imposed whether a block or block sequence that evaluates to a Boolean (red boxes) and/or numeric (green boxes) value can be inserted. The design uses localized abrupt discontinuities instead of overall shape (as in "virtual Scratch") due to the difficulties in perceiving overall shape differences with touch. Through a sequence of user studies, magnets were added to the connections to allow increased haptic and audio feedback when a connection was successful to increase ease of use. Slots were physically defined for parameters/operands/command subsequences similar to "virtual" Scratch so that users knew what was needed to complete the commands (e.g., Figure 4d, Figure 5). Telescoping tubing was used to allow the slots to fit a single block (variable or value) or expand to include nested operator expressions, again similar to "virtual" Scratch.

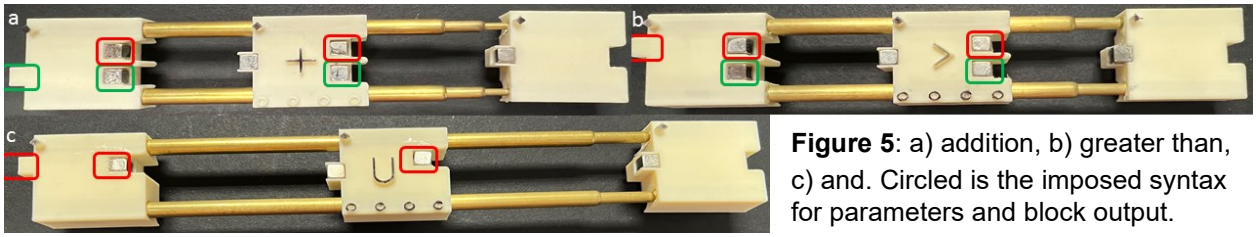

Figure 5: a) addition, b) greater than, c) and. Circled is the imposed syntax for parameters and block output.

The final component of the system developed was creating an accessible alternative for "virtual" Scratch's visual stage to show the program output. A key requirement for the design of our BVI accessible stage was that users needed to be able to accurately follow the program output non-visually to determine whether the program was correct; this task is crucial for successful program debugging. Two auditory methods (descriptive audio feedback and 3D sonification of sprites) and one haptic method (mobile robots that could be followed by the hand) were evaluated for this requirement as well as their ability to engage users. Mobile robots were considered as they were known to spark interest and curiosity in children (Eguichi, 2022). The auditory methods were considered as they were expected to be able to more easily handle multiple sprites and Scratch Looks commands (i.e., which would allow a closer parallel between sighted and BVI use of Scratch), although their ability to engage learners was unknown. User studies with BVIs found that users were more engaged and could follow the program execution better when mobile robots were used. Key to these conclusions was the use of a mobile robot (Figure 6, right: a Pololu turtle coupled to a "cap" where the user's hand rests) that could move accurately and the use of stage "backgrounds" with raised axes and an incised grid to haptically determine distances while avoiding visual clutter (Figure 6, left: note that the incised grid is not visible visually). In Figure 6, removable haptic features were added to a standard background for a story project.

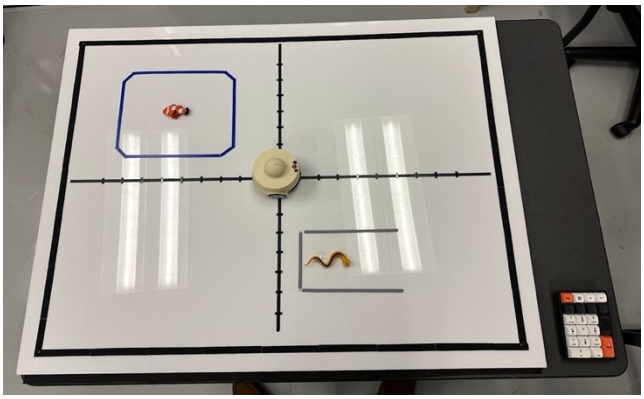

Figure 6: Left, physical stage with the background used for one of the programming projects, the mobile robot sprite, and numeric key input pad. Below, the mobile robot sprite.

Instructional material was developed for a 2-day camp that targeted BVI middle school students based on resource materials and curriculums available on both the Scratch website (scratchmit.edu/educators) and the Computer Science for all in SF website (csinsf.org), as well as the PI's experience teaching computer science and K12 students. Feedback and guidance was also provided by Janice Johnson (TVI, retired) and two BVI adults, unfamiliar with programming, who piloted the 2-day camp. The programming concepts covered in the 2-day camp included sequencing, events, if-else statements, variables, relational operators and logical operators, along with the concepts of incremental development and program debugging. The camp was a hands-on experience with the instructor describing what was to be done aurally while seated in front of the participants. Students had a project at the end of each day; a story on Day 1 and a game on Day 2. The material was provided at the pace of the student(s) present, although the same material was covered for all students. Quizzes and reference materials were created with a combination of Braille and raised line drawings.

BVI students were recruited for the camps from Virginia and surrounding areas through organizations for the blind, parent groups for BVI children, teachers of BVI children and schools for the blind. Seven middle school/high school BVI students, unfamiliar with programming, participated in the camps. The main objective of the camps was to determine whether BVI participants used the system in the way we designed it be used. The second objective was to determine if the BVI participants enjoyed the experience and if it made them excited about programming. The camps were too short to effectively evaluate learning of computer science concepts. To assess our objectives, an audio-visual recording of

each of the participants interacting with the system was made, as well as a post camp interview. Qualitative content analysis was used to assess the data collected. For the most part, students used the system as expected. However, there were a few issues that suggest the need for some improvements including better readability of some of the haptic symbols, eliminating the occasional sticking of the telescoping tubing, better organization of currently unused code segments on the work surface, and use of a robot that could handle a larger variety of textures. Student enjoyment of the camp and interest in programming was clear from student comments both during the camp and in the post-camp interview. In addition, many students strongly requested (and were allowed) to save the programs that they constructed to show their parents when they came at the end of the day to pick them up.

Future Work will focus on increasing meaningfulness by increasing the diversity of project types and personalization, as well as increasing the social participation through a website that will provide information and resources for students, parents and teachers and act as an accessible conduit to the official Scratch website. In the next phase, we will also work with Science Centers across the national to expand our reach through 5 day computer camps for mixed group of sighted and visually impaired students.