Computer Science Accessibility for Students with Visual Disabilities

Joan M. Francioni
Computer Science Department
Winona State University
Winona, MN 55987
jfrancioni@winona.edu

Ann C. Smith
Department of Computer Science
Saint Mary's University
Winona, MN 55987
asmith@smumn.edu

Abstract

Students with visual disabilities face unique challenges in learning to be computer scientists. These challenges can be overcome, however, with the use of specialized software tools and hardware equipment, collectively called assistive technology. In this paper, we discuss the environment we are using for three students with visual disabilities who are starting in our programs this year. This environment includes a collection of commercial assistive technology and a programming tool that we have developed in-house.

1 Introduction

Computer Science is a popular major choice for high school students with disabilities planning on going to college [1]. This is true for the same reasons that computer science is a popular major choice for high school students in general. However, there are two other significant reasons for why we are seeing an increase in the number of students with disabilities wanting to major in computer science. For one, many students with disabilities use computer technology in their day-to-day lives for communication and/or access to information that would not otherwise be available to them. This exposure often makes computers less intimidating to these students than to others. The second reason, however, is what makes majoring in computer science actually a viable choice for many students with disabilities today. That reason is the American Disabilities Act of 1990 (ADA). Since the ADA ruling, it is no longer acceptable to waive the course requirements of students with disabilities in high school. In particular, students with disabilities have the same mathematics requirements as everybody else. Therefore, any student who wishes to major in computer science, and who has the aptitude for studying computer science, will have the necessary high school background to start the program.

Paper published in proceedings of *33rd SIGCSE Technical Symposium on Computer Science Education*, Northern Kentucky, February 2002, pp. 91-95.

How, then, can we make our programs more accessible to students with disabilities? Different aspects of college and the computer science curriculum are more or less challenging for particular students, depending on their disability. For students with visual disabilities, studying mathematics and navigating a programming environment present two unique challenges. As part of our Computer Science Curriculum Accessibility Project [2], we are working with three first-year students with visual disabilities who are majoring in computer science. In this paper we will discuss our strategies for making our programs accessible to these students.

2 General Assistive Technology

A number of commercial assistive technology tools are available that enable people with visual disabilities to access electronic and printed information. The most basic piece of assistive technology for someone who is blind or who has a severe visual disability is what is called a *screen reader*. These tools speak the information that is on the screen out loud. Information can either be read as characters or as words. When reading sentences, a screen reader uses the grammar rules of the natural language to determine the appropriate inflection and intonation to use. Directing the tool as to what part of a document to read, or to start and stop reading, can always be done using the keyboard alone. In addition to reading text, screen readers are also designed to navigate a Windows environment and to browse Web pages in an intelligent way.

For people with limited visibility, tools that magnify the image on the screen are sometimes useful. *Screen magnification* tools have the capability of increasing the size of the screen image from slightly larger all the way to large enough so that one letter takes up the entire screen.

Screen readers provide one way of accessing information that is available on-line. But it is also useful and highly desirable sometimes to be able to read text in Braille. For on-line information, this can be done either by printing the material to a Braille printer, or *Braille embosser*, or by using a *refreshable Braille display* device. The latter is a device that represents the current line on the screen in

Braille and that is able to change letters in real-time as the cursor is moved to another line.

Information that is in printed form or that is non-textual needs to be handled differently. Printed text does not pose any real problems, however, since it is possible to scan the text in and translate it into characters using a good OCR (optical character recognition) program. From there it can either be read out loud or displayed in Braille.

For images and diagrams, there are two concerns: one is generating a tactile version of the image or diagram; the other is conveying the "information" of the image or diagram. Just doing the first one, doesn't always imply the second one. In Sections 3 and 4 we discuss problems in this area specific to Mathematics and Computer Science. As for actually generating a digital image or diagram in a tactile form, this can be done by using certain kinds of Braille embossers or by using a *thermal embosser*. The first is a type of impact printer and the second uses special *swell paper* that reacts to heat.

Given this basic set of software and hardware, students with visual disabilities are able to access most of the same information that is provided for sighted students. There are a few caveats, though. For one, although the information in a text book can generally be scanned in and read using a screen reader, the text of a Mathematics text book has too many equations to make this a viable option. (Screen readers do not recognize many mathematical symbols.) For another, depending on how a Web page is designed, the information content of the page may not be accessible. A third example is that PDF files are not yet accessible with today's technology. Things are changing for the better, however. Recent Federal legislation [5] requires that all government Web sites, as of June 21, 2001, be accessible. This is helping people become more aware of how to make a Web page accessible and it is also encouraging the development of Web page composer software that can help a designer generate accessible pages from the start. (See [15] for a complete set of guidelines for making a web page accessible.) As for accessing PDF files, to date, only one screen reader can directly access PDF files created with Adobe 5.0. How soon other screen readers and Braille embossers will be able to access PDF files, and how this all plays out with the Digital Millenium Copyright Act [3], remains to be seen.

3 Mathematics

The study of Mathematics presents particular challenges to people with visual disabilities. Our students have taped copies of their textbooks. In addition to this, we are using specialized assistive technology in three basic ways to support our students in their Math classes: (1) to generate tactile versions of Math equations for notes and exams; (2) to represent mathematical functions and graphs; and (3) to support direct communication between the teacher and student during office hours.

Currently, there are two viable ways of representing mathematical equations in a tactile format: one is called Nemeth code and the other is called DotsPlus. Nemeth code is a special Braille code, developed by Dr. Abraham Nemeth in 1946, specifically for mathematics and other technical subjects. Nemeth code is capable of rendering all mathematical and technical documents in six-dot Braille. This is accomplished by letting the same six-dot pattern mean different things in different contexts. Also, multiple six-dot patterns are used for many mathematical symbols. Some blind students find Nemeth code intuitive, but others have a hard time learning and remembering the code. The main problem with Nemeth code at the college level, however, is not about the students as much as it is about the teachers. For one, most college Mathematics teachers do not know Nemeth code and therefore cannot introduce the appropriate codes to the students as the topics are introduced in class. Secondly, at the present time, Nemeth code does not translate into print with the click of a button, as with literary Braille. Therefore, even though there are tools to translate documents containing mathematics to Nemeth code [9], there is no easy way for the student to solve a problem in Nemeth code and relay that solution back to a sighted teacher in print [11].

An alternative to Nemeth code is DotsPlus [6], which is not really a code but rather a set of tactile fonts. When printed by an embosser that understands DotsPlus, text and numbers appear in standard Braille formats, while common punctuation marks and mathematical symbols appear as small graphical symbols. These symbols are distinguishable out of context and, together with the number and letter formats, include all characters represented within computer screen fonts. As such, any computer document written in a language based on the Roman alphabet can be converted to DotsPlus. Therefore, it is possible to generate one document, say using Word and MathType, and print it out in either print form for a sighted reader or tactile form for a non-sighted reader. At this point, although it is possible, it is still cumbersome for the non-sighted student to generate the mathematical equations directly.

For our project, we are using the Tiger [14] embosser from ViewPlus Technologies, which is capable of printing out DotsPlus documents. With this set-up, we are able to generate notes and exams using Word and MathType that are then converted to a Tiger font and printed on the Tiger. Although this strategy is working fine so far, the Tiger is a relatively new machine and we are two of only four universities that are using this printer for students currently taking college-level mathematics courses. Along with Oregon State University and M.I.T., we are working to determine the most effective way to use the printer and to educate the students.

In addition to being able to represent mathematical equations, students also need to be able to represent and understand functions and graphs. We are using both the Tiger and the Accessible Graphing Calculator (AGC) [14]

for this. The Tiger uses a standard Windows graphics-capable printer driver. Therefore, it is capable of taking the graphical content of a document and rendering it directly to produce a tactile representation on the paper. AGC is a Windows-based graphing calculator. In addition to the standard features of a hand-held scientific graphing calculator, it is capable of displaying graphs or other sets of X vs. Y data both visually and audibly, as a tone graph.

For direct communication between the teacher and student, we are experimenting with various hand-drawing tactile tools. We have also found it to be very useful for the students to have a tactile copy of the notes for class before class. This, of course, requires the cooperation of the Math instructor.

4 Computer Science

4.1 Diagrams

Since the Tiger embosser has the capability of generating tactile diagrams, we are using a strategy similar to the one above for generating tactile representations of diagrams in computer science. The other problem that has to be solved, however, has to do with being able to interpret a tactile diagram. A sighted person sees the whole picture first and then figures out what the parts are that make up the whole. Conversely, a blind person must figure out the parts first and then combine them together to construct the whole. So, for example, in the diagram in Figure 1, a sighted person can quickly deduce that there is a square centered on the X-Y axis. Even if you were not thinking the X-Y axis, it is still visually apparent that the long horizontal line is all one line, and similarly for the long vertical line. Conversely, a blind person feeling along the long horizontal line starting from the left can not tell in the beginning how long the line is. Therefore, starting from the left of the horizontal line, there is a question of "which way to go" as soon as you get to the left edge of the square.

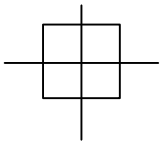

Figure 1. Sample Tactile Diagram

To display images (rather than just line diagrams) on a tactile device, algorithms to identify object boundaries are used. Research in this field focuses on edge detection and image segmentation methods that simplify an image enough to generate an accurate and effective tactile representation of the image [7]. Along these same lines, our initial observations indicate that it would be better to present line diagrams to someone who is blind in a "layered" sequence of diagrams. Using the above example, one layered sequence would be as shown in Figure 2.

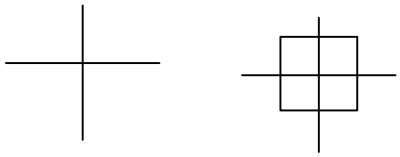

Figure 2. Sample Layered Diagram

In this way, the blind student can identify the "parts" of the diagram more easily and, hopefully, gain a better and quicker understanding of the whole diagram.

Working with the students with visual disabilities in our programs, we are experimenting with different schemes for our diagrams. These include varying the size of the diagrams, the thickness of the lines, and the sequence ordering of different "layers" of the diagrams.

4.2 Programming

Learning to program, of course, is a major part of being a computer science student. It is a challenging task for many students, but for students with visual and certain learning disabilities, this task takes on an extra level of difficulty. For someone who already knows how to program and is familiar with a Unix environment, Emacspeak [12] provides an effective, eyes-free programming environment. For *learning* to program, however, students need a different kind of tool. To this end, we have developed a specialized programming environment for Java, called *JavaSpeak*, to help students with visual disabilities learn how to program.

JavaSpeak Functionality.

Basically, JavaSpeak is an integrated development environment (IDE) with aural feedback and keyboard navigation control that is designed to provide a user with useful information about a program's structure and semantics. It parses the program and then "speaks" the program's structure to a user, in much the same way that separate lines and indentation and color all help to "show" the structure of a program to a sighted user. It is also designed to be easily configurable, so that we can adapt it as we learn more from working with the students who are using it.

Version 2.0 of JavaSpeak is designed to offer students an auditory rendering of their program at one of the following eight levels:

1. *Basic compilation unit*. This level depicts the package declaration, any import declarations, and the class and interface declarations of the program.
2. *Class composition.* This level presents the instance and class variables and methods that make up the class.

3. *Method composition.* The names, parameters, and throws clauses of each method of a class are given.
4. *Nesting levels.* Information depicting the nesting levels within methods is given.
5. *Block composition.* This level depicts the conditionals, the number of statements, and the variable declarations within each block, as well as the instance and class variable declarations for the class.
6. *Block details.* This level combines levels 1, 3, 4, and 5 with details of the block statements added.
7. *Token by token.*
8. *Character by character.*

The intent is to help students group individual tokens of the language together as syntactic units, as they are developing their programs.

In addition to giving a student different "views" of a program, JavaSpeak is also designed to incorporate gestures of spoken English to convey semantics about the program. By speaking the code in different ways and with different intonations, information is given about the organizational structure of the code as well as the logic. For example, consider the code segment in Figure 3. Three different aural renderings related to this code are described in Table 1.

We use English punctuation in Table 1 as a mechanism to depict how the code would sound. For example, in the Block Composition column, the colon after the while condition indicates an expectant pause, rather than, say, the completion of a sentence. Actual sound file examples can be found at the following web site: http://cs.winona.edu/CSCap/javaspeak.html .

```
public class PrimeFactors {
 static boolean continue = true;
 public static void main (String [] args) {
  . . .
  while (continuePlaying) {
   while (testNumber < 0) {
     System.out.print("Enter number:");
     try {
       numString = in.readLine();
     }
     catch (Exception e) {
       System.out.println("Input Error");
     }
     testNum = Integer.parseInt(numString);
   }
   if (!isPrime(testNumber))
     . . .
```

Figure 3. Code Segment

JavaSpeak Implementation.

A simpler prototype version of JavaSpeak was introduced at ACM's *Assets 2000* conference [13]. This version included a basic editor and was able to parse a program to generate the speech renderings, but it did not offer all the current levels nor operate as a full compiler. Since then, a fully integrated programming environment has been developed as Version 2.0. This version of JavaSpeak makes extensive use of existing, open source code. In this way, we were able to put together a tool that has the appropriate functionality for students to do serious programming.

For the front-end of the system, we use NetBeans [10]. NetBeans is a modular, standards-based IDE, written in Java. It has support for building client- and server-side applications in Java, with a wide range of features including debugging. All of the pieces of functionality are implemented in the form of modules that plug into the NetBeans core. Therefore, we are able to choose which features we want to include so as to keep the complexity level appropriate to the level of the course. NetBeans is available from Sun and is the base tool platform for their Forte IDE.

Class Composition	Nesting Levels	Block Composition
Begin class declaration. Public class prime-factors. Class variable: static boolean, continue = true. Method: main. Method: is-prime. Method: print-factors. Method: restart. End class declaration.	Begin while 1? Begin while 2? Begin try 3? Try. Catch. End try 3. End while 2. Begin if 2?	Begin while 1. While, continue-playing: Begin while 2. While, test-Number is-less-than 0: Statement. Begin try 3. Statement. Catch: Statement. End try 3. Statement. End while 2. Begin if 2. If, not is-prime of test-number:

Table 1. Descriptions of Sound Renderings

The back-end of the system is a modified version of the Kopi compiler [4]. Kopi contains a set of tools to edit and generate Java class files, including DIS (Java disassembler), KSM (Java assembler) and KJC (Kopi Java compiler). KJC compiles from Java source code to bytecode and is freely available under the terms of the GNU Public License.

Two other pieces of the puzzle are the Java Speech API (JSAPI) and the Java Speech Markup Language (JSML) [8]. JSAPI defines a standard, cross-platform software interface to speech technology. Two core speech technologies are supported through the JSAPI: speech recognition and speech synthesis. We are using the speech synthesis part to produce synthetic speech from text.

The Java Speech Markup Language (JSML) is used by applications to annotate text input for JSAPI speech synthesizers. The JSML elements provide a speech synthesizer with detailed information on how to say the text. JSML includes elements that describe the structure of a document, provide pronunciations of words and phrases, and place markers in the text. JSML also provides prosodic elements that control phrasing, emphasis, pitch, speaking rate, and other important characteristics, however, we are not currently using these elements.

To generate the speech patterns that we want, we have the compiler generate text renderings of the program that include JSML tags and that are based on how the code was parsed. We then use an implementation of the JSAPI to actually have the renderings spoken.

Current Status.

Our three students with visual disabilities are using Version 2.0 of JavaSpeak this semester. Lessons learned from this semester's use will be presented at the SIGCSE conference and will also be available at our web site (http://cs.winona.edu/CSCap). Information on downloading JavaSpeak will also be available at this site.

5 Concluding Remarks

All of the tools and techniques described in this paper are important for supporting students with visual disabilities in a computer science program. However, in our opinion, they are not sufficient. The other necessary component is the willingness of the faculty to experiment with and to be open to new and different ways of communicating ideas. We have been very lucky at Saint Mary's and Winona State to have such a supportive group of faculty for this first group of students. We are sure things will get easier as we learn how to better meet the needs of the students. But we are also sure it will always take support from the instructors to meet the students half way.

6 Acknowledgements

We wish to acknowledge Christopher Johnson, Anna Rouben and Ross Rosemark for their excellent work on Version 2.0 of JavaSpeak.

The Computer Science Curriculum Accessibility Project is supported, in part, by the National Science Foundation.

References

1. Blackorby, J., Cameto, R., Lewis, A., & Hebbeler, K., "Study of Persons with Disabilities in Science, Mathematics, Engineering, and Technology," SRI International, Menlo Park, CA, 1997.
2. Computer Science Curriculum Accessibility Project, http://cs.winona.edu/CSCap
3. Digital Millenium Copyright Act of 1998, www.loc.gov/copyright/legislation/dmca.pdf
4. DMS Decision Management Systems GmbH, Vienna, Austria, http://www.dms.at/kopi/
5. Federal Regulation, Section 508: www.section508.gov
6. Gardner, John A., *The Science Access Project*, Oregon State University, http://dots.physics.orst.edu
7. Hernandez, Sergio and Barner, Kenneth, "Tactile Imaging Using Watershed-based Image Segmentation," in proceedings of *Assets 2000*, Washington D.C., November 2000.
8. Java Speech API, http://java.sun.com/products/java-media/speech/
9. Karshmer, Arthur, *MAVIS (Mathematics Accessible to Visually Impaired Students)*, New Mexico State University, http://www.nmsu.edu/~mavis
10. Net Beans Integrated Development Environment, http://www.netbeans.org/
11. Osterhaus, Susan, *Teaching Math to Visually Impaired Students*, http://www.tsbvi.edu/math/math-nemeth.htm
12. Raman, T. V., "Emacspeak – Direct Speech Access," in *Assets '96*, April 11 - 12, 1996, Vancouver Canada, pp. 32-36. http://cs.cornell.edu/home/raman/emacspeak/publications/assets-96.html
13. Smith, Ann C., Francioni, Joan M., and Matzek, Sam D., "A Java Programming Tool for Students with Visual Disabilities," in proceedings of *Assets 2000*, Washington D.C., November 2000, available at http://cs.winona.edu/CSCap/assets2000paper.doc
14. ViewPlus Technologies, http://www.viewplusTech.com/
15. World Wide Web Consortium's (W3C's) Web Content Accessibility Guidelines: www.w3.org/TR/WCAG10