Inspiring Blind High School Students to Pursue Computer Science with Instant Messaging Chatbots

Jeffrey P. Bigham, Maxwell B. Aller, Jeremy T. Brudvik, Jessica O. Leung,
Lindsay A. Yazzolino, Richard E. Ladner
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195 USA
{jbigham, maxaller, jtbrudvi, joyleung, lindsaym, ladner}@cs.washington.edu

ABSTRACT

Blind students are an underrepresented group in computer science. In this paper, we describe our experience preparing and leading the computer science track at the National Federation of the Blind Youth Slam. As part of this workshop, fifteen blind high school students created and personalized instant messaging chatbots, a project designed to be completely accessible to blind students. Chatbots enable students to infuse their own personalities into a socially-oriented program that incorporates ideas from artificial intelligence, natural language processing, and web services. We first outline the chatbots project and curriculum, which has wide appeal for all students, and then offer general design principles used to create it that can help ensure the accessibility of future projects. Students created their chatbots using a real programming language and were guided by both blind and sighted mentors. By *programming from the start* in a supportive environment, our students will gain the confidence to persevere in computer science in the future.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]; K.4.2 [**Social Issues**]: Assistive Technologies for persons with disabilities

General Terms

Design, Human Factors

Keywords

Accessibility, Blind students, Chatbots

1. INTRODUCTION

Blind students are underrepresented in computer science [8]. Factors that discourage blind students from pursuing computer science include a) exaggeration of the difficulties that they may face, b) a lack of mentors knowledgeable about assistive technology and c) inaccessible course projects and material [2]. Better understanding, encouragement and development can help inspire blind students to pursue computer science, and, to that end, we led a four-day, computer science workshop for fifteen blind high school students as part of the National Federation of the Blind Youth Slam. The Youth Slam attracted two hundred blind and low vision high school students and enabled them to explore disciplines falsely believed to be too difficult or impossible for blind people to pursue.

Figure 1: One of the authors demonstrates screen reader usage to a workshop participant. Both are wearing headphones to hear the screen reader.

We wanted this workshop to be not only exciting but also illustrative of the problem solving and creativity that is core to computer science. Introductory workshops in computer science often target producing visual artifacts because such projects are immediately appealing to sighted students [10]. Other workshops interest students with computer-controlled models, such as LEGO Mindstorm robots [5]. Both types of projects allow students to exercise their creative problem solving skills, but currently require using software that does not work well with screen reading or magnifying software. While such projects could be made accessible to blind students, their largely visual appeal may be lessened.

The project we chose centered on instant messenger chatbots, which leverage social and technological appeal to interest students. Chatbots are software robots with which students and their friends can interact. Students can pro-

SIGCSE'08, March 12–15, 2008, Portland, Oregon, USA.

gram their chatbots to carry on simple conversations that reflect their personality or interests. Instant messenger is an increasingly popular medium for communication among high school students, which awards it social appeal. These factors make an instant messenger chatbot an ideal project to inspire high school students to pursue computer science.

The following criteria were used to select the chatbots project and represent important aspects of a project for an introductory workshop for blind students:

1. The project must illustrate an exciting application of computer science that allows students to infuse their personality, imagination and intelligence.

2. The project must utilize the problem-solving skills that form the core set of computer science and engineering without undue time spent learning programming language syntax, complex tools or jargon.

3. Project components must be designed to be easily accessible to our student population. In our case this meant that all components were easily accessed using a screen reader.

As part of the project, students created personal chatbots endowed with unique personas. Examples of chatbots that we suggested to students included the following: a psychologist, a fortune teller and Yoda from Star Wars. No students chose to implement these examples. Instead, they created chatbots with personas that reflected their own personalities and interests, such as an anti-Bush conservative, an insult-giving smart alec and a poetic mathematician. Chatbots can also exhibit technological sophistication. They can draw information from the web to answer useful questions like "What's the weather in Baltimore?", "Who won the Orioles game last night?", or "Where's the nearest Chinese restaurant?" The creativity exhibited in the chatbots created by our students exemplified the wide range of possibilities made available by the chatbots project.

Many computer science projects involve working with a text terminal. Chatbots make such text input/output seem intelligent and exciting, perhaps because students are accustomed to talking to human friends using instant messenger. From the programming perspective, it is no different than creating a console program. Students were initially captivated even by their "Hello World" chatbot because they could type to it via instant messenger and it would respond. The excitement seemed to stem from the fact that most students had previously used instant messenger programs to talk to other people. Their chatbots immediately seemed more interesting than interacting with a terminal because it was easier to think of them as being intelligent. Students quickly realized that their chatbots were not smart and most were excited to learn how to make them appear smarter. As they did so, they learned the important lesson that computer programs that appear intelligent base their behavior on clever but deterministic programming. The instant messaging window served the function of a simple terminal window, but students found it exciting.

The remainder of this paper is organized as follows: Section 2 overviews the workshop and Section 3 outlines the principles followed to ensure an accessible curriculum. Section 5 summarizes the chatbots created by our students and Section 6 reports on lessons learned from the workshop.

2. WORKSHOP OVERVIEW

2.1 Assistive Technology

Blind individuals use screen readers and magnification software to access computers (Figure 1). Screen readers, such as JAWS [1] and Window-Eyes [3], convert both visual interfaces and information to speech. They provide keyboard shortcuts that enable efficient use of programs with the serial voice interface provided by screen readers. For instance, screen readers provide a shortcut to skip from heading to heading in web pages. This shortcut does not already exist because sighted users easily skip between visual headings. Screen readers need to be customized for each new program and not all programs are supported. Many blind individuals prefer to use magnification software, such as ZoomText[4], to enlarge the contents of their screens. These programs scale elements to high magnification while preserving important components and relationships.

A vital aspect of using a computer when blind is learning to use a screen reader. All of our students that used a screen reader had used one prior to our course, although each student's proficiency varied. Several mentors in the Youth Slam program were experienced screen reader users and, while this level of experience was incredibly valuable, it was not necessary to assist students. Each computer was equipped with a screen reader and a screen magnifier. An audio splitter that enabled instructors to hear what students were hearing was invaluable in assisting students.

While our focus was on designing a program for blind high school students without prior experience programming, many of the lessons learned are informative for ensuring usability in computer science education whenever populations with different access requirements are involved.

2.2 Programming from the Start

Our workshop was heavily influenced by the success of the Game of Life Workshop [10] which is a week-long, computer science workshop for disabled students. The workshop stresses problem-solving skills that form the core of computer science. Students new to programming wrote Java programs that visually simulated the complex interactions of cellular automata. Each student was paired with a computer scientist who could help them implement their ideas and debug their programming. The goal of this program was to guide students through the process of computer programming in a time frame that made it impossible to teach them everything one would teach a typical CS1 student.

Our workshop followed a similar model of "programming from the start." Even though the workshop lasted only four days, we introduced students to real programming in a supportive environment. This approach serves two valuable purposes. First, much of the initial frustration experienced by students is caused by learning confusing programming syntax. Introducing programming in a supportive environment may help students avoid this. Second, students who choose to pursue computer science will face real programming assignments from the start. By enabling them to successfully complete a realistic programming assignment, we hope to instill confidence in their abilities.

One student summarized the workshop for other Youth Slam participants as follows: "After the first day, we realized that programming is hard, but after we got into it, it was really fun!" This is a positive review of our approach.

```
class HelloWorldBot : BasicBot {
  public override string HandleMessage(
    string message,
    string user,
    BotMemory bm)
  {
    return "Hello World!";
  }
}
```

Figure 2: Source code for HelloWorld chatbot.

Students experienced first the struggle and then reward of computer programming, which they will certainly face in future computer science courses but may not have experienced with a simplified project. This workshop offered readily available assistance for students to help them avoid most of the frustration of learning to program. Instructors helped students turn their ideas into code and assisted them in quickly debugging their programs, but students programmed independently.

2.3 Provided Framework

We provided an instant messaging chatbot framework that allowed chatbot programs to be relatively short and understandable. Students were provided with a method that is called by the framework each time a new chat message arrives. Parameters hold the received message, the name of the user who sent it and a data structure that can persistently store data tied to each user. Figure 2 contains code sufficient to produce a "Hello World" chatbot which will respond to any message that it receives with "Hello World."

The project is written in C#, whose syntax is virtually indistinguishable from Java in the context of the chatbots project. The C# compiler is readily available on most Windows PCs and produces easy-to-understand error messages. The framework uses third-party instant messaging libraries written for the .NET platform and supports the AOL, Yahoo, Windows Live and Jabber protocols. This flexibility enables students to demonstrate their chatbots to friends regardless of which protocol they are using, helping to extend the excitement of our program beyond the workshop.

3. ACCESSIBLE DESIGN

The software used in the chatbots project was both accessible and easy to learn. To encourage students to continue participation after the workshop and to enable other groups to easily use our curriculum, the software chosen is freely available. The best way to ensure accessible design is to actively engage consultants from your target population in the process of developing, testing and publishing your curriculum. We developed our curriculum with close consultation of two blind computer users and with the support of the National Federation of the Blind.

3.1 Instant Messaging Client

Our blind consultants chose the Windows Live Messenger (formerly MSN Messenger) as being the most accessible instant messaging client. While Windows Live Messenger is not the most popular instant messenger client among the general population, it is quite popular with blind individuals because of its accessibility. The AOL Instant Messenger client came in a close second in terms of accessibility, but the AIM servers prevent users from logging off and on too quickly. This would be a burden during the usual software development cycle of write, compile and test. Such balances of accessibility and technical concerns is common.

3.2 Programming Environment

Students used the shareware text editor TextPad[1] for programming. This lightweight application is well-suited for beginning programmers because of its straightforward interface and extensive ability to be customized. This program is used in our CS1 course and was recommended by our blind consultants. Preliminary testing indicated that the latest version (5+) of the program did not work well with the JAWS screen reader because non-intuitive shortcut keys were used to switch between the editing and command-result panes of the program. Instead, we used version 4.7.3, which did not exhibit the problem.

We printed examples of chatbot code in braille for students, but few students used these when coding, preferring instead to refer to code examples on the web page. Most students had not previously read computer braille. The code samples were written in computer braille, which includes braille characters for additional symbols often used in computing. Listening to the code samples sounded like the code that they would later be editing and did not require them to read computer braille. These braille code samples enabled students to appreciate the structure of code, which is not apparent when listening to it with a screen reader.

3.3 Tutorial

With only four instructors and fifteen students, each student did not have a dedicated instructor. To enable independent learning when an instructor was not available, we created an online tutorial that provides a series of exercises focusing on concepts useful for creating chatbots. After a brief, high-level discussion of what programming is, the tutorial introduces strings because messages are received and sent as strings. Conditionals come next because they enable chatbots to appear intelligent as the mechanism by which chatbots make decisions. Basic regular expression matching is introduced early because chatbots seem more realistic when they can flexibly match incoming messages. Concepts related to state provide chatbots with memory that enables more intelligent chatbot dialog (Figure 3). Finally, we covered methods provided by the framework that allow information to be retrieved from web services. The tutorial covered a small subset of C# necessary to code chatbots in order to keep students from being overwhelmed by the syntax.

Most students were accustomed to browsing the web with a screen reader and the tutorial was made available on the web for this reason. The tutorial web pages utilize heading and list HTML tags extensively to enable students to efficiently skip between exercises. In-page links are used to enable students to quickly navigate within each page. A common method of navigating a web page with a screen reader is to move from link to link and all links are assigned descriptive anchor text to support this. Others have offered tips for creating accessible web pages [9, 7].

Our student-to-instructor ratio was low (approximately 4 to 1) and enabled us to provide timely attention to students as they completed the tutorial. Absent specific requests for

[1] www.textpad.com

```
class HowAreYouBot : BasicBot {
  public override string HandleMessage(
    string message,
    string user,
    BotMemory bm)
  {
    if(bm["asked"] == "yes") {
      bm["asked"] = "no";
      return "That's great!";
    } else {
      bm["asked"] = "yes";
      return "How are you today?";
    }
  }
}
```

Figure 3: Storing and using state in a chatbot. When a message is first received, the chatbot replies with a question. When it receives another message, it replies acknowledging an answer. It then resets.

assistance, instructors moved from student to student, working through the tutorial with them, answering questions and explaining concepts.

3.4 Summary

The chatbots project was designed with a focus on accessibility for blind students, but was designed with inclusive universal design principles to make it appropriate for most students [6]. We distilled the following design principles for creating accessible computer science projects:

- Prepare resources that can be easily converted into an accessible form. An electronic format is often best because it can be transformed into large-print, electronic speech, braille and others.
- Consider both technical requirements and accessibility when choosing software. Both can be achieved.
- Test early and extensively with members of your target population. If no such consultants are available, test using the tools that this population will use.

4. WORKSHOP EXPERIENCE

Students in our class were enthusiastic. The biggest challenge they faced was signing up for an instant messenger name for their chatbot because doing so required solving a CAPTCHA and none of the students could solve the difficult audio version. Sighted instructors ended up solving the visual CAPTCHA instead. Initially, programming was difficult for students, but by the end of the week many were creating new chatbots independently. All students successfully created chatbots.

All fifteen students in the workshop were described as blind, but blind individuals exhibit a great diversity of visual acuity and skill sets. Six students preferred to use their limited vision in preference to a screen reader. Two students preferred to use the ZoomText [4] screen magnification software. Several of our students did not know braille.

Coding difficulties often resulted from the lack of structure conveyed by screen readers which read code serially. Students would often type new code statements beyond the brackets of the main method or even within the quotes of

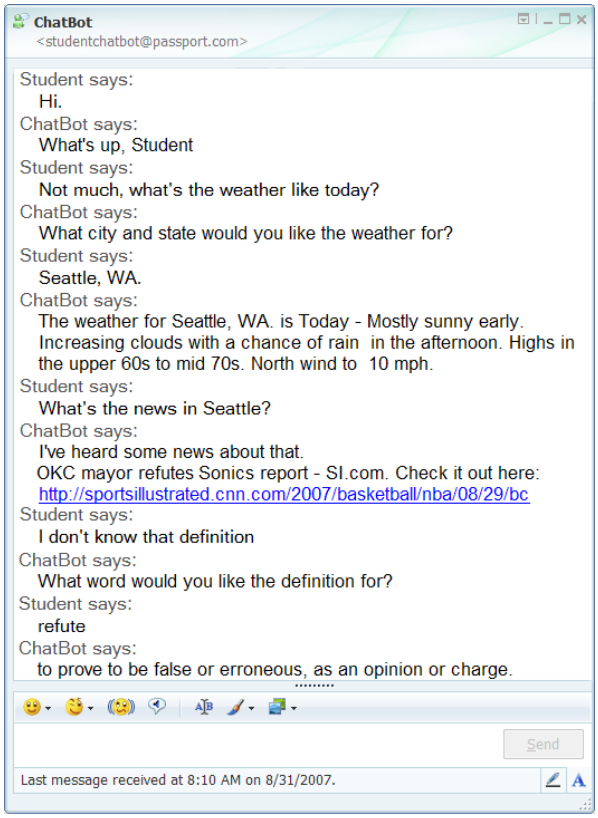

Figure 4: An example conversation with a chatbot produced by a workshop participant that illustrates the ability of chatbots to interact with web services.

an existing string. These errors are different than we have observed with new sighted programmers. A technique successfully used to improve their sense of the code's structure was to frequently move the cursor within the code they were editing. The screen reader speaks the character beneath the cursor when it is moved and by moving around quickly students could effectively hear a small, two-dimensional window around their cursor. The most successful screen reader users would also check for typos using this method. This is analogous to when a sighted person glances over what they have typed. Screen readers can be set to read aloud each character as the user types it, but most found the feature to be bothersome and to slow their typing.

Some students continued to work on their chatbots after the workshop. The projects were easy for students to use on their home computers because the required software is free and the setup and tutorial documents are on our web page.

5. CHATBOTS CREATED

Students created an impressive variety of chatbots. Figure 4 shows an example dialog. Chatbots generally expressed elements of three themes: technology, dialog and personality:

Technology Many chatbots focused on the technological capabilities that they could provide to their chatbots. Chatbots could access web services to retrieve up-to-date news and sports information and chatbots that could retrieve the definitions and synonyms of words.

Dialog Some chatbots were fluent with dialog in a specific area, such as math or small talk. One student built a chatbot

with the goal of replicating his own dialog. He signed his chatbot onto instant messenger using his usual IM name and a friend didn't realize that it wasn't him for eight messages.

Personality Some chatbots reflected the unique personalities of their creators. Many students spent as much time perfecting the clever responses that their chatbots would give as they spent programming. For example, one student had her chatbot respond "Melissa loves clothes, shoes, and hair accessories, but only if they're cute" by default.

6. LESSONS LEARNED

6.1 A Diversity of Abilities

All of our students were blind, but each student's visual acuity and experience using the assistive technology varied greatly. All of our students that required the use of a screen reader had previously used one, which is unfortunately atypical in the general blind population. Unsurprisingly, those students who were most proficient at using a screen reader were initially most successful. Programming is challenging using a screen reader, but most students quickly adapted and increased their screen reader proficiency as they programmed. They will carry this broadly applicable skill with them in the future. Despite differences in abilities, all students were able to program their chatbots with the functionality that they wanted with variable amounts of coaching.

6.2 Instructor Experience

Our instructors were not experienced screen reader users and we found that this was not a requirement to teach students who use screen readers, although it could have helped. Instructions based on direction (e.g., move the cursor to the left or bring focus to the control above the current one) were effective in assisting students navigating through the TextPad editor. Describing the type of element that needs to be used was enough to enable our students to find it. Instructors found value in listening to the output of the screen reader along with students. Learning to use a screen reader can be difficult, but gaining a sense of what students are doing is possible by both listening and watching the screen.

As previously mentioned, all of our students had prior experience using a screen reader. If they had not, then it would have been valuable to spend time initially overviewing basic screen reader usage. Most commands are fairly intuitive and existing Windows shortcuts still apply. For instance, the arrow keys still move the cursor in the direction expected, the TAB key moves from element to element in menus and ALT-TAB switches between programs. Using these basic commands will allow users to accomplish much of what is possible. The screen reader help dialog and experienced users helped students increase their shortcut repertoire.

Several blind mentors also assisted students in our track. Although all were not knowledgeable of computer science, most were experienced computer users and were invaluable in helping students increase their expertise in using their screen reader. We found that students often consulted one another to find more efficient ways to accomplish the tasks that they wanted. A few of the mentors were also familiar with the "sound of code," which is substantially different than the visual representation displayed on the screen and were able to help students become accustomed to it.

Instructor patience and approachability were paramount. Completing tasks for our students was tempting because screen readers can seem incredibly inefficient and difficult to use. Students do not learn how to accomplish tasks when they are performed using a mouse because they don't hear the process. Helping students to complete tasks using the keyboard will allow them to do so independently next time. Programming can be frustrating and we actively encouraged students to ask for help. We also moved from student to student asking if they had questions and asking them to explain what they were doing.

7. WEB RESOURCES

The chatbot curriculum, framework and accessible programming tools are available on our web page:

`http://webinsight.cs.washington.edu/chatbots/`

8. CONCLUSION

This paper has described our experience preparing and leading the computer science track at the National Federation of the Blind Youth Slam. We have presented both the curriculum used in the course and the principles followed to ensure that it would be accessible to our students. Students created impressive, personalized chatbots using a real programming language. We hope that this process will inspire these students to pursue computer science and contribute to the confidence that will enable them to persevere.

9. ACKNOWLEDGEMENTS

Our workshop has been supported by National Science Foundation grant IIS-0415273 and a Boeing Professorship. We thank the National Federation of the Blind for organizing the Youth Slam of which our workshop was a part.

10. REFERENCES

[1] Jaws 8.0 for windows. Freedom Scientific, 2006. http://www.freedomscientific.com.

[2] National center for blind youth in science concept paper. National Federation of the Blind, 2006. http://www.blindscience.org/ncbys/.

[3] Window-eyes. GW Micro, 2006. http://www.gwmicro.com/Window-Eyes/.

[4] Zoomtext. Ai Squared, 2007. http://www.aisquared.com.

[5] D. J. Barnes. Teaching introductory java through lego mindstorms models. In *Proc. of the Technical Symposium on Computer Science Education (SIGCSE '02)*, pages 147–151, New York, NY, USA, 2002.

[6] S. Burgstahler. Universal Design of Instruction. University of Washington, DO-IT, 2005.

[7] J. Clark. *Building Accessible Websites*. New Riders Publishing, Indianapolis, IL, USA, 2003.

[8] National Science Foundation. Women, minorities, and persons with disabilities in science and engineering. *NSF Report*, 04-317, 2004.

[9] J. Thatcher, P. Bohman, M. Burks, S. L. Henry, B. Regan, S. Swierenga, M. D. Urban, and C. D. Waddell. *Constructing Accessible Web Sites*. glasshaus Ltd., Birmingham, UK, 2002.

[10] T. VanDeGrift, S. Burgstahler, R. E. Ladner, and A. Poginy. The game of life workshop - reaching out to high school students with disabilities. In *Proc. of the 2006 ASEE Annual Conf. and Exposition*, 2006.