Robotics Programming Tools for Blind Students

Stephanie Ludi

Rochester Institute of Technology

salvse@rit.edu

Abstract

Despite advances in assistive technology, relatively few visually impaired students participate in computer science courses. Significant factors in this underrepresentation include lack of precollege preparation, access to resources, and the highly visual nature of computing. This paper describes the development of software that provides an accessible Lego Mindstorms NXT programming environment for teenage students who are visually impaired. With the popularity of robotics in both pre-college and introductory programming classes, such an environment has the potential to accommodate students who are visually impaired in classroom and outreach activities. Pilot testing results will be presented in addition to the design of the system.

Keywords

Visually impaired, outreach, teens, robotics

Introduction

In the US as the numbers of students pursuing computer science and related degrees has declined in recent years, efforts by the National Federation of the Blind (*Youth Slam*); Cannon, Panciera, and Papanikolopoulos; and Doerschuk, Liu, and Mann have been undertaken to increase interest and participation in computing degrees. Like other underrepresented groups, students who are visually impaired do not participate in computing careers to their full potential. Factors include inadequate preparation and awareness of potential career paths. Students who are visually impaired continue to encounter barriers in computing courses in terms of access to course tools and participation in activities in part due to highly graphical depictions within user interfaces.

As robotics has become popular as a means for engaging pre-college students in computing and engineering (including efforts by the National Federation of the Blind and Cannon, Panciera, and Papanikolopoulos), the need for accessibility persists. Work by Ludi and Reichlmayr has demonstrated how Robotics, such as Lego Mindstorms, are as appealing to students who are visually impaired as they are to sighted students. The default programming software available from Lego uses icons to represent commands. This software is not accessible, most notably in terms of screen reader compatibility. Whether for in-class activities or extracurricular outreach, the software needs to maximize accessibility in order to promote interest in computer science and related disciplines.

The underrepresented students of concern are those who are visually impaired, where the threshold is legally blind. The American Federation of the Blind defines the term “legally blind” as defined through federal law, with “central visual acuity of 20/200 or less in the better eye with

the best possible correction, as measured on a Snellen vision chart, or a visual field of 20 degrees or less" (*Statistics*).

The goal of the JBrick project is to devise accessible Lego Mindstorms programming software that can be used by those with or without sight. In the case of the ImagineIT workshops and future outreach, the target users are teens who are visually impaired. These teens are often novice programmers, as the focus of the outreach is to enable the participants to explore Computer Science via robotics.

Evaluating Mainstream Tools

This goal was derived over time, in part due to evaluating other environments that meet accessibility and other criteria. Research was conducted to find the best alternative programming software and language in terms of accessibility and cost. BricxCC, developed by Hansen, and the NXC (Not eXactly C) language were the initial choices for use in our student outreach workshops but the BricxCC software, developed in Delphi, is not entirely compatible with JAWS and thus requires the help of a sighted person at times. For example, code navigation can require help by a sighted person when the program is large since displayed code line numbers are not read. The alternative is for the user to count the lines, which can be frustrating and time consuming in a large program. A summary of the alternatives are as follows:

Table 1 Overview of the Pros and Cons of Using RobotC, Microsoft Robotics Developer Studio, and LejOS platform

Robotics Platform	Advantages	Disadvantages
Robot C	• Used in FIRST high school robotics programs • Has a simulator • Is relatively low cost • An entry point for beginners but good for advanced • Mac or PC	• Has many issues with JAWS • Simulator and help are not accessible
Microsoft Robotics Developer Studio	• Free • Can be used with many different robotics systems • Can work with the Kinect • Use of C# is nice for those who already know how to program	• PC only • Visual programming language is not accessible • Use of C# is not novice-friendly • Simulator not accessible
LejOS (for Java) uses NetBeans or Eclipse	• Free • Used in some universities • Use of industry tools	• Requires changing the NXT firmware • Not novice friendly • Buggy

The NXC language was chosen over Java, another popular alternative. The NXC language is easier to learn, and in order for the robot to work with Java programs, the robot's firmware needs to be changed. These factors will promote replication of developed materials in schools or at home. Each BricxCC feature was assessed with the JAWS screen reader and Zoomtext screen reader and magnifier, as described in work by Ludi and Reichlmayr. Persistent issues had some impact on participant satisfaction. The decision was made to migrate from BricxCC to a new system--JBrick.

JBrick Features

The desire to have an accessible Lego Mindstorms programming environment has been the main driver for JBrick. The accessibility assessment of BricxCC, combined with notes taken during workshops when BricxCC was used with teens, provided a set of accessibility and features needs that in turn were the foundation for JBrick's requirements. The primary assistive technology is assumed to be screen readers and magnification software programs, though Braille displays are also significant.

The core concerns are:

- Accurate reading of the source code
- Ability to recognize the NXT brick, compile code and download programs to the brick and indicate status of process with audio
- Ability to open and save source code files
- Ability to read errors from the compiler
- Ability to navigate to the line of code that the compiler error is referring to quickly

Some of the more refined requirements include the following:

- The user shall read/hear the source code, including keyword highlighting and line numbers
- The user shall view/hear errors from the compiler, where each has an associated line number
- The system's interface and code files are compatible with JAWS and Zoomtext
- The system shall be usable by sighted and visually impaired users alike, either working separately or collaboratively
- The system shall provide audio cues for status such as compiling, downloading to the brick, or finding the brick
- The system must be multiplatform
- The system must be compatible with the NXC compiler and not require firmware changes to the brick
- The user shall be able to change font size and font/background colors

The sample of the system features and quality attributes presents the fine-grained approach to requirements that include customizability of the user interface.

Design of JBrick

In order to ultimately deliver JBrick on multiple platforms, Java was selected as the development language. The NXC compiler is being reused, so programs coded in JBrick and BricxCC are interoperable. Here is a sample of an NXC program:

```
// sound THRESHOLD in decibels to trigger movement
#define THRESHOLD 10
// name for SENSOR_2 (sound sensor)
#define SOUND SENSOR_2

task main()
{
  SetSensorSound(IN_2);
   // loop forever
  while(true)
        {
  // keep sampling until threshold is exceeded
              until(SOUND > THRESHOLD);
        OnFwd(OUT_AC, 75);
        Wait(300);
  // resume sampling for same threshold
        until(SOUND > THRESHOLD);
  OnRev(OUT_AC, 75);
        Wait(300);
   }  }
```

Standard Java libraries have been used, including the Java Accessibility libraries.

The JBrick user interface is simplified, as shown in Figure 1. The drop down menu structure has been streamlined from 8 menus in BricxCC to 5 menus in JBrick. This streamlining

can help with recall when the menus are read. JBrick has fewer icons, each larger in size and omitting superfluous aesthetic detail and tooltips. Navigation can now be accomplished with the keyboard as the sole input device. Common commands such as compile and download can be accessed through keyboard shortcuts.

Line numbering is visible to the left of each line of source code. The numbering is managed separately in order to be displayed on the screen, read by the screen reader, and displayed on a printout. Such features help both new programmers and experienced programmers work with code regardless of visual acuity.

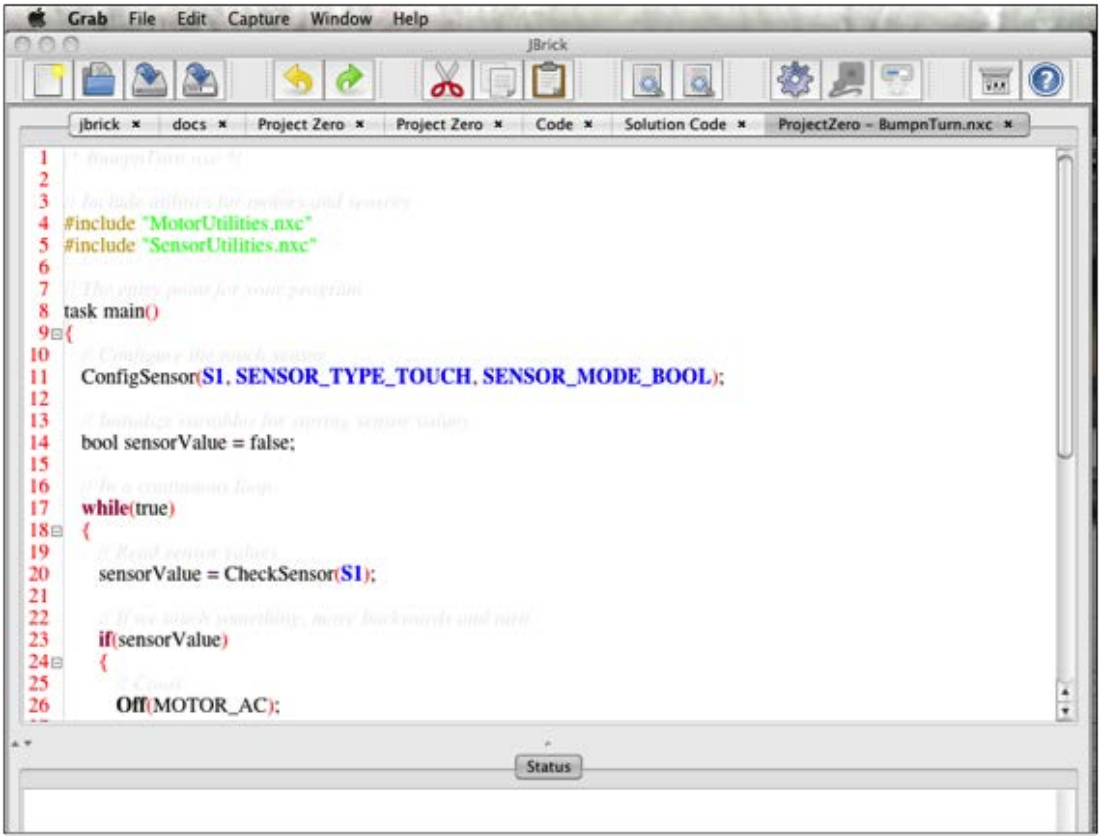

Fig. 1. Screenshot of JBrick Software-Depicting a Sample Code File List.

The large icons and streamlined menu are displayed.

Font legibility is also critical as default font sizes are often small. In JBrick, the default format for the font is Ariel Bold 12. The size can be increased. The color of the text, keywords, and the background can also be changed in order to accommodate the needs of the user. The default coloration is high contrast. The menu options and key sequences to compile and download files onto the brick can be accessed as a menu item, icon, or key sequence. The improvement has been in the form of a beep and dialog box to inform the user as to the status of the process rather than silence (as is the case with BricxCC). JBrick is also usable on Mac OSX and is compatible with VoiceOver. These features increase accessibility for users with visual impairments in addition to increasing usability for sighted users. For example, adjusting font sizing and screen coloring helps with legibility. Streamlined menus minimizes short-term memory overload for all users.

User Testing

After the technical and initial accessibility evaluations were complete, JBrick was pilot tested with a group of 5 undergraduate students with varying visual acuity, including 2 students who are blind. The students had no experience in programming Lego Mindstorms robots (or programming at all). Using a pre-designed NXC programming tutorial, each student worked through a set of tasks to learn how to use JBrick in addition to completing a simple programming activity using a stationary robot. The workflow tested consisted of the following tasks:

1. Open the program, and change the view to include the file tree and change the proportions of the other pane
2. With the program already open, open a new file and enter code
3. Save the file as test.nxc

4. Compile the file and download to robot
5. Open the Preferences pane and change the colors of text highlighting and any other preferences, except for the location of the compile
6. Open a new file then navigate back to test.nxc
7. In test.nxc, force an error by deleting a semicolon and try to compile. Use the error messages to fix the problem
8. Copy and paste part of the body of main in test.nxc
9. Fold the body of main in test.nxc
10. Close all open files, then close the program

Due to the small sample size, observational data included the number of errors made. Time to complete tasks was not as important since students work at their own pace.

Discussion

While all of the students ran into some issues with getting the test program to work, the issues were mostly in the area of text entry rather than JBrick itself. While the NXC language cannot be revised at this stage, the Future Work section discusses an approach to mitigate program entry (typing) issues. Excluding text entry issues, tasks 1 through 8 and task 10 were completed with 0 errors. Task 9 was thrown out due to a technical defect. However, in the course of the tasks' execution some minor defects were discovered. In Task 5 (Preferences), the tab order was incorrect, though users were able to complete the task. During Task 7 (Fixing the Defect), 1 student was a little frustrated by having to listen to multiple errors that were generated. This issue goes back to text entry/typing skills, given that text-based programming can generate numerous errors from typos. In addition, small technical (user interface) defects were found with

the GoTo command (misreading the ENTER key) as well as being able to fold the MAIN method in order to minimize the reading of some code (Task 9). The issue with the GoTo command did not prevent the user from completing the task, and most users did not notice the defect during the test. The defect in Task 9 caused issues and was thus thrown out of the user study. The results showed that the line numbering helps students find parts of their code when editing, though a change in voice for the number would give the features more value. In addition, no screen reader or magnification issues occurred with JAWS or ZoomText.

Conclusions and Future Work

JBrick has been successful in terms of providing an accessible foundation in Lego Mindstorms NXT programming for pre-college students with varying degrees of visual impairment. As such, students can collaborate with sighted peers or other students who are visually impaired during robotics activities that build technology skills. Moving forward, additional features will be added in order to provide students the ability to navigate code using audio cues, as well as debugging (a tool to help work through coding and logic defects). JBrick will be tested with high school students participating in the Computer Science Academy, starting in Summer 2013. In addition, Versions of JBrick for iOS and tablet PCs are underway that will provide new means of interaction. For example, JBrick for the tablet PC uses physical blocks as a way to help enter commands and edit programs. This user interface may help younger students and those who either lack keyboarding skills or who have some dexterity issues.

Works Cited

Cannon, Kelly, Katherine Panciera, and Nikolaos Papanikolopoulos. “Second Annual Robotics Camp for Underrepresented Students.” *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. Scotland: ACM Press, 2007: 14-18. Web. 10 Oct. 2011.

Doerschuk, Peggy, Jiangjiang Liu,and Judith Mann. “INSPIRED Computing Academies for Middle School Students: Lessons Learned.” *The Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations*. USA: ACM Press. 2009. 52-57. Web. 12 Oct. 2011.

Hansen, John. *BricxCC Command Center Homepage*. N.p. 2007. Web. 16 Sept. 2012.

Ludi, Stephanie, and Tom Reichlmayr. “Developing Inclusive Outreach Activities for Students with Visual Impairments.” *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. New York: ACM Press, 2008: 439-43. Print.

Ludi, Stephanie, and Tom Reichlmayr. “The Use of Robotics to Promote Computing to Pre-College Students with Visual Impairments.” *ACM Transactions on Computing Education*. 11.3 (2011): 1-20. Web. 10 Oct. 2011.

Statistics and Sources for Professionals. American Foundation for the Blind. 2010. Web. 6 Mar. 2011.

Youth Slam 2009 Homepage. National Federation of the Blind. 2009. Web. 27 Oct. 2011.