المجلة الأكاديمية للأبحاث والنشر العلمي | الإصدار الثاني | تأريخ الإصدار: ٥-٦-٢٠١٩
ISSN: 2706-6495

Teaching programming for the blind

Prepared by Researcher: Mohamed Abdel Latif Ismail

Ain Shams University – Cairo

Email: mnoman72@htomail.com

Abstract

This research illustrate advances in computer accessibility over the past few decades, and what we have resulted in many computing devices and applications that are accessible to blind and visually impaired individuals. However, computer programming tools still present significant accessibility challenges to blind people, and blind people are currently underrepresented in computer science.

Keyword: Teaching programming, blind, computer accessibility

Introduction:

Introductory programming activities for students often include graphical user interfaces or other visual media that are inaccessible to students with visual impairments. Digital fabrication techniques such as 3D printing offer an opportunity for students to write programs that produce tactile objects, providing an accessible way of exploring program output. This paper describes the planning and execution of a four-day computer science education workshop in which blind and visually impaired students wrote Ruby programs to analyze data from Twitter regarding a fictional ecological crisis. Students then wrote code to produce accessible tactile visualizations of that data. This paper describes outcomes from our workshop and suggests future directions for integrating data analysis and 3D printing into programming instruction for blind students.

Advances in computer accessibility over the past few decades have resulted in many computing devices and applications that are accessible to blind and visually impaired individuals. However, computer programming tools still present significant accessibility challenges to blind people, and blind people are currently underrepresented in computer science. This effect is likely due, at least in part, to a lack of compelling accessible instructional materials and tools for learning how to program . Developing a supportive environment in which blind students can learn to program presents several challenges. First, the programming tools must be accessible to the student and must work with the assistive technology that he or she uses, e.g., a screen reader, screen magnifier, or refreshable Braille display. Second, the student must be provided with programming tasks

Request permissions from networking data via the Twitter API, manually explored that data via their code, and visualized that data via an interactive iPad application and 3D-printed tactile models (Figure 1). We then report the results of this activity and identify additional opportunities for integrating data analysis and 3D printing into accessible computer science programs.

Literature review

Teaching programming for the blind: Challenges and opportunities

We will answered the following questions

- Is there a need for learning programming among the blind?
- What are the main challenges that may face them in learning programming?

Teaching Programming to Blind Students via 3D Printing, Crisis Management, and Twitter

The workshop took place over four days in July and August 2013, as part of the computer science track of the National Federation of the Blind's STEM-X program. STEM-X (formerly known as the NFB Youth Slam) is a week-long summer science camp for blind and visually impaired youth. In 2013, STEM-X had fifty attendees from across the United States. STEM-X students chose one disciplinary track (computer science, chemistry, engineering, robotics, or aerospace science) to explore during the week. For four days, students spent half the day working with instructors in their track, and half the day participating in science enrichment and social activities. On the final day, students reconvened to show off their work to their peers. Students in all subject tracks worked together to solve a shared problem: the (fictional) impact of Comet ISON. Students in the aerospace track built a working hovercraft, students in the chemistry track studied the science behind desalinization plants, and students in the robotics track built robots to help find people in trouble. Students in the computer science track monitored comet sightings using social media and wrote programs to produce tactile visualizations of the predicted impact zone. At the end of each day, students came together to share status updates about the impending comet impact, and to discuss next steps.

The computer science track featured 9 students (3 female) from grades 8 to 12. Students varied greatly in their prior computer and programming experience, and also varied in their visual abilities and preferred assistive technologies. The computer science track had 6 instructors: 2 faculty members, 2 graduate students, and 2 undergraduate students. The instructors were also assisted by 2 mentors, who helped students with any general issues that came up, including basic computer or screen reader problems. The high instructor-to-student ratio ensured that students did not have to wait for help for very long if they became stuck, which was especially beneficial as most students were programming novices. However, students were able to make independent progress even when instructors were unavailable, suggesting that this workshop could work well with fewer instructors.

Programming Tools

Choosing an environment for introducing students to computer programming can be challenging, especially for blind students, who might have difficulties working with standard programming tools. Thus, much of our preparation work focused on choosing appropriate programming tools and developing supporting library code for students to work with. Our primary goals in choosing programming tools were to create an environment that would be easy to start programming in, to use real programming languages if possible, and to enable students to try out several programming concepts over the course of the workshop. We chose Ruby as the programming language for this workshop, as it had a number of advantages over competing languages. First, Ruby is a mainstream programming language that is available on many systems, and which offers many standard libraries. Second, Ruby syntax is comprised largely of text symbols (e.g., “if”, “then”, “end”), and contains relatively few non-alphanumeric symbols that may be difficult to recognize via a screen reader. Third, Ruby provides an interactive interpreter, irb, that enables students to learn using a “read-eval-print loop” [13]. We considered using the Python programming language, which also satisfies these criteria, but Python uses whitespace to delimit code blocks, which could be confusing to navigate with a screen reader. Since students only spent about 16 hours total in the workshop, they would be unable to make sophisticated programs completely on their own.

To enable these novice students to interact with compelling applications, we created several library functions that students could call to interact with more advanced features. These functions were grouped into four major categories: Twitter: a wrapper for the Twitter API that enabled students to log in, search tweets, and post tweets; Geocoding: functions for reverse-geocoding tweets that contained location data; Data visualization: functions that added geocoded tweets to an accessible map visualization, which could be viewed interactively on an iOS-based device using VoiceOver; 3D printing: functions that assembled geocoded tweets into a 3D model of a tactile map, which could be 3D printed.

Workshop Curriculum

The programming workshop took place over four days, and met for four hours each day during this time. The majority of workshop time was spent working on programming activities. Each day also featured a guest speaker, who called in to discuss their work with the group. Guest speakers included a professor and crisis informatics researcher, a blind software engineer, and a blind computer science graduate student. Because students began with different levels of background knowledge, we expected that they would proceed through the workshop activities at their own pace. Thus, students generally worked individually on the activities, and sought guidance from instructors or their peers if they became stuck. All workshop activities were written out as a step-by-step tutorial and posted on a single web page. The instructors set an approximate schedule for the workshop, as described below, and worked with students to help them complete each day's tasks.

INSIGHTS FROM THE WORKSHOP

Overall, we considered the workshop to be successful. While not every student completed every activity, every student spent several days developing his or her programming skills. Spending four days teaching high school students to program in Ruby also provided valuable insights about the suitability of Ruby for blind programmers,

About teaching beginning programmers to explore and visualize data, and about keeping students engaged through an intensive programming course

Using Ruby for Blind Programming

Generally, students in our workshop were successful at writing Ruby programs, using the terminal to launch Ruby programs, and using irb to test code. There were, however, several usability issues relating to the interaction between Ruby and screen readers that created minor challenges.

3D Printing in Introductory CS

Students were clearly excited by the use of the 3D printer during the workshop. When the 3D printer was demonstrated, students paid careful attention and asked questions. Students were eager to touch the printer and observe its mechanics. We printed tactile graphics for each student, and most students were excited to take the tactile graphics home as a souvenir. When students had the opportunity to test both interactive touch screen graphics (presented on the iPad) and tactile graphics, students clearly seemed more interested in the tactile graphics. It is unclear whether students preferred the tactile graphics because they were unfamiliar, because they were accessible, or for some other reason, but even some students who were less enthusiastic about their programming activities were intrigued by the 3D printer hardware and its output. We also found that students who did not read Braille were eager to explore the Braille printed on the tactile graphics. As Braille literacy has declined in recent decades, and because Braille literacy seems closely related to employment [11], using 3Dprinted objects to motivate Braille learning presents an exciting opportunity for future work. In general, it seemed clear that the 3D printer was a valuable addition to the computer programming workshop curriculum. However, there were some challenges in using the 3D printer in the classroom. First, the printer is quite slow: the tactile graphics, which were approximately the size of a credit card (85mm × 54mm × 5mm) each took approximately one hour to print on the MakerBot Replicator printer. Faster printing settings are available, but result in a more brittle object.

As a result of the slow print time, we were not able to print each student's tactile graphics in class, but instead collected data at the second-to-last meeting, printed the tactile graphics overnight, and delivered them at the final meeting. Furthermore, even with the default settings used, the printed tactile graphics could be quite brittle. Some parts of the tactile graphics would easily wear or break off, including fine details such as the 3D-printed Braille. While the prints made with the current printer were usable, there remains room for improving the quality and durability of the 3D-printed tactile graphics, especially since tactile graphics are likely to be handled frequently.

CONCLUSION

Learning to program still presents many accessibility challenges for blind and visually impaired people. One major opportunity is to identify introductory programming experiences that are compelling to novice programmers, but that are also accessible to programmers with varied abilities. We argue that combining data analysis tools with visualization tools, and with the fabrication of tactile graphic-based visualizations, presents an ideal environment to teach programming to blind students. Our results from a four-day workshop show that blind students were motivated to learn about 3D printing technologies, and to use their programming skills to create 3D-printed artifacts. We also found that Ruby and its interactive interpreter offer a sufficient, if not perfect, environment for teaching blind students to program. We hope that this work will motivate the development of software tools and curricula to support blind programming students in the process of exploring, analyzing, and visualizing data.

Reference

Code Talk: Improving Programming Environment Accessibility for Visually Impaired Developers

National Federation of the Blind. 2006. National Center for Blind Youth in Science. Retrieved September 6, 2013 from http://www.blindscience.org/ncybs-concept-paper

https://www.researchgate.net

Shaun K. Kane Department of Information Systems UMBC Baltimore, MD 21201 skane@umbc.edu

Information literacy competency standards for higher education. USA : Association of college & research libraries. .- Available at :www.ala.org/acrl Esther Grassian and Susan E. Clark. Information literacy sites: background and ideas for program planning and development. In: college and Research Libraries. Vol. 60, no.2 (Feb. 1999).pp.78-81.

ODLIS: on-line dictionary of library and information science .- Available at : http://www.wcsu.edu/library/odlis.htm

Davis, Gillian . Bibliographic instruction : An overview.- Available at : www.suite101.com/article.cfm/9460/86846

Texas Information Literacy Tutorial.- Available at :http://tilt.lib.utsystem.edu/intro/internet2.htm

Dabbour, Katherine Strober. Applying active learning methods to the design of library instruction for a freshman seminar. In College and Research Libraries. Vol. 8, no.4 (July 1997. P.299-308.

Small, Ruth, Nasriah Zakaria, and Houria El-Figuigui. Motivational aspects of information literacy skills instruction community college libraries. In : college and Research Libraries (March 2004).pp.96-120.

Fleming, Hugh. User education in academic libraries. London: Library Association, 1990. P.44-45.

Fleming, Hugh. Ibid. P.44.

Ibid. P.42.

Ibid., p.45.

Fleming, Hugh. Ibid. p.43.

Dabbour, Katherine Strober. Applying active learning methods to the design of library instruction for a freshman seminar. In College and Research Libraries. Vol. 8, no.4 (July 1997. P.299-308.

Davis, Gillian.Ibid.p1

Objectives for information literacy instruction: a model statement for Academic librarians. American Library Association.2003.- Available at: http://www.ala.org/ala/acrlstandards/objectivesinformation.htm

Guidelines for instruction programs in Academic Libraries. American Library Association. 2003.- Available at :