Including Students with Disabilities in Computing Education for the 21st Century

Richard E. Ladner
University of Washington
10/18/2013

Approximately 10% of high school students in the United States have a recognized disability. The majorities of these students are college capable and could become computer professionals. Indeed, there are already some successful computer scientists who have disabilities, but the number is very small. According to the Survey of Earned Doctorates, only 1% of PhDs in computing fields are disabled. I contend that there should be a significant effort within the CE21 program at NSF to make sure that the curricula developed for computing at the high school level be accessible to students with disabilities and that the effective pedagogies for these computing courses be welcoming for these students. It is not merely a matter of social justice but for the health of the discipline. Research and/or development teams, especially those who are developing consumer products and services, will be more effective with engineers and scientists with disabilities on the team. They will look at designs from a different perspective, one that might make the final design more universal, that is, more likely to be usable by the maximum number of users.

The research question that we address in this white paper is:

- What are the best ways to include students with disabilities in computing courses?

Naturally, this would depend quite a bit on the nature of the disability. Nonetheless, this question can be broken down into two fundamental questions.

1. Inclusion: How do we make the course welcoming and inclusive for participants with disabilities?
2. Accessibility: How do we make the software and hardware tools used in the computing course accessible to participants with disabilities?

To begin answering these questions it is helpful to give an example from my own experience teaching high school students in an enrichment program.

Illustrative Example
I taught over 30 deaf or hard of hearing high school students in the Saturday Computing Experience (SCE), which has been held at the University of Washington for the past three years. The purpose of the SCE is for the students to learn using a project-based learning style that encourages students to learn
programming concepts because they need them to complete their projects. Arduino was used in 2011, Scratch in 2012, and Processing in 2013 as tools for learning programming. The SCE is about eight weeks long, held on Saturday mornings from 9:30 to noon. The program enrolled only deaf or hard of hearing high school students, mostly freshmen and sophomores who had no experience with programming. Each year there were about ten students in the program. A major feature of the program is one-on-one mentoring from experienced programmers: undergraduate students, graduate students, and professionals. Mentors helped guide the students in solving various programming projects. Each student had their own final projects that took the majority of the approximately thirty hours in the program. This pedagogical approach was used successfully from 1995 to 2005 in the Game of Life Workshop [1].

Although all the students were fluent in sign language, only a few of the mentors were. To help facilitate communication we employed a pair of sign language interpreters who would be on call in the room. We also provided tablets and pens so that students and their mentors could communicate in writing. We provided handouts that explained various concepts and provided short practice projects. There was virtually no lecturing in the program, as students learned what they need to learn to complete their individual projects. The program was inclusive in that we recruited at high schools in the area that had small deaf programs.

For the most part, the technologies we used were accessible to our deaf and hard of hearing students. Arduino, Scratch, and Processing are all visually oriented products. However, one deaf student who wanted to come the second year (Scratch) had other disabilities including low-vision and mobility disabilities caused by cerebral palsy. She also required a personal attendant for any bathroom needs during the program. Our first reaction to her asking to participate was to answer, “yes, of course.” Her parents were somewhat wary, but we invited them to all come to the University to talk about it and assess the program for themselves. We tried to be inclusive by being positive and welcoming. The student already had some assistive technology that enabled her to participate in school and could bring it with her to the SCE. The technology included a joy-stick to control the mouse, a special board with key guards for typing, and a large monitor to enable large print.

By the second week we learned that the Scratch interface was not as accessible as it first appeared. The model for drag-and-drop used by Scratch to move a block from the holding area to the program area was very different than more standard drag-and-drop interfaces that allow you to first select a block to be moved, and then move it with the mouse, touch pad, or with arrow keys. With the Scratch interface, blocks are already active in the holding area. If you click on a block, then the action defined by the block actually happens. For example, a block might say rotate right 90°, in which case accidentally clicking on the block rotates an active element, called a Sprite, 90°. Clicking does not select, but activates. Now imagine someone like our student who does not have smooth control of her arms and hands. When she tried to create her programs using the Scratch drag-and-drop interface she would most often accidentally activate a Sprite or drop a block in the wrong place. There was no way to select a block and move it with the arrow keys which she is very capable of doing. The bottom line is that Scratch was not accessible to her.
In our motivating example we saw illustrations of both inclusion and accessibility. Inclusion was exemplified by the welcoming attitude for all students regardless of disability and letting them and their parents know that whatever is needed for the student to participate will be provided. Accessibility, or the lack thereof, was demonstrated by one of the unforeseen limitations of Scratch. It was not very accessible for a student with a mobility disability.

**The Opportunity**
The computing education community is just now developing two relatively new K-12 curricular thrusts: Computer Science Principles and Exploring Computer Science. What if these efforts were infused now with an inclusive attitude and accessible tools, rather than making it an afterthought? To a degree this is already happening. The NSF-funded AccessComputing Alliance has as its mission to increase the number and success of students in computing fields. It has been doing this for the past 7 years by supporting effective direct interventions, promoting institutional change, and by disseminating up to date and accurate information to educators, students, and parents. AccessComputing helped create the group for Accessibility and Universal Design on the CS10K Community web site. AccessComputing partners and collaborators have created curricula and tools that support learning about programming, robotics, and other topics [2-6]. AccessComputing co-PI, Sheryl Burgstahler, wrote an excellent article for the ACM Transactions on Computing Education on universal design of instruction [7]. Several articles by AccessComputing staff have appeared in the CSTA Voice. The AccessComputing webpage get more than 100,000 hits per month, many of them to the Knowledgebase that contains useful information for those teaching students with disabilities. This is just the beginning. Much more needs to be done in two directions:

1. Building and evaluating more accessible software, tools, and curricula that support students with disabilities learning computing in K-12.
2. Augmenting the existing CS10k infrastructure so teachers of students with disabilities can get the information, software, tools, and curricula they need.

It is pretty clear that many of the current software, tools, and curricula are not very accessible. The most popular tools for learning programming concepts, Scratch, Snap, Alice, Processing, and Greenfoot, are not very accessible to blind students. All these platforms are inherently visual. There are other problems with Scratch that were mentioned earlier. The Quorum programming language and accompanying IDE, developed by Andreas Stefik and his colleagues is much more accessible than the current most popular tools and has been tested successfully with blind students. A teacher who happens to have a blind student in her class could potentially use Quorum with all or some of her students to teach programming concepts. Since Computer Science Principles and Exploring Computer Science are supposed to be programming language agnostic, there is no fundamental reason not to use Quorum in implementations of these courses. However, in the end it might be better to make the most popular tools more accessible rather than having a separate tool just for blind students.

The current CS10k infrastructure is really a loose network of projects that are tied together with various focal points such as the CSTA, Computer Science Principles organization, and Exploring Computer
Science organization. There is also the CS10K Community website that is beginning link together the multitude of K-12 education projects in computer science. This CE21 Conference is a major vehicle for engaging the community. Nonetheless, to the best of my knowledge there are no current CE21 grants that directly address the inclusion of students with disabilities. Several projects have connected with AccessComputing which is a good start, but more needs to be done to make sure that students with disabilities are included in computing education for the 21st century.

**Potential Actions**

1. More CE21 projects should collaborate with AccessComputing to help them with their broadening participation mission.
2. There should be at least one new CE21 project whose role is to be a focal point for making sure that teachers have the resources and training to include students with disabilities in their computing courses.
3. There needs to be an exploration of ways to sustain this project beyond the life of a single grant.

**Acknowledgement**

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1042260. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

**References**