Improving Student Engagement Through the Incoporation of Robotics into Introductory Computer Science Curricula

Allison Thompson
Department of Computer Science, Calvin College,
3201 Burton Street SE, Grand Rapids, MI 49546 Email: ajt5@calvin.edu

Abstract— **It is fairly well known that the number of students interested in majoring in Computer Science has been low for the past few years, and that retaining students in the field has been difficult. The Institute for Personal Robots in Education is in the middle of a three-year experiment to determine if adding robots to introductory Computer Science courses will increase interest in Computer Science, and assist in the retention of students in the major. The University of Tennessee is participating in that experiment by conducting one of their own, adding robots to part of their introductory Computer Science course. We assisted in this project by creating the labs that will be used in that course.**

I. INTRODUCTION

In the past few years the number of freshman interested in pursuing a Computer Science major has fallen. Students seem uninterested in the field. Those in the field have suggested a plethora of possible solutions to fix this problem. Several interest groups are working to implement some of those suggestions, to determine if they will actually work. One such group is the Institute for Personal Robots in Education (IPRE). They are conducting an experiment to see if the addition of robots to introductory Computer Science courses will attract more students to the Computer Science major, as well as assist in retaining more students in the program.

In this project, our objective was to design labs for the University of Tennessee's CS 102 course. This Fall, the University will be assisting with the IPRE experiment by testing the theory of adding robots to introductory courses. Our goal was to create labs that teach the introductory computing skills necessary for students' success in future Computer Science courses while simultaneously introducing students to the basic concepts of robotics, all in a way that would be more fun than frustrating.

II. RELATED WORK

There are two other examples of curricula designed with goals similar to ours. Both were created through IPRE, the organization that instigated our project.

The first example was a course designed for middle schoolers by students at Bryn Mawr College. Their goal was to generate interest in Computer Science in students at an earlier age, as a means of attracting more potential majors to the discipline. They designed stimulating and engaging [2] activities that used Python and the Parallax Scribbler robots, in hopes that these activities would increase interest in and awareness of the computing disciplines. They also added some functions to the existing Myro infrastructure (more details on Myro later) to simplify the more complex aspects of Python. By doing so, the Bryn Mawr students exposed their participants to actual programming without necessitating that they understand complicated syntax.

At the end of their experiment, the number of middle schoolers who thought that computing was fun, feasible, and useful had increased. Most of the participants were more interested in computing as a result of the course; however, the number of students interested in taking courses involving robots had decreased, as had the number who said they enjoyed the challenge of computing [2]. The Bryn Mawr students in charge of the experiment suspect these negative shifts of opinion were caused by the not infrequent malfunctions of the robots.

The Bryn Mawr students believed their approach was successful in its design and in communicating concepts; though, as they note in their conclusion, a larger, more robust dataset [2] would be needed for further improvements.

The second example is the basic IPRE curriculum, off of which, in theory, any course that makes use of personal robots in education could be based, though the authors of the report on IPRE's work do not make such claims directly. The IPRE curriculum was designed with the goal of presenting a motivating and relevant course that would excite students and get them hooked on computer science. Rather than create a robotics course, IPRE worked to create an introductory CS course based on robotics [1]." The IPRE curriculum is based on these key ideas:

- use of a personal robot
- tools that have a low floor and a high ceiling; i.e., that are easy for a novice to learn, but also that are of a high enough power that an expert could continue to use them
- expand students' perceptions of computing [1]

IPRE notes that it is essential to their approach that each student have his or her own robot [1]. IPRE regards low price, portability/convenience, and robustness as the most import features of the robots chosen for the course. The authors note that previous work found that students who are only able to use robots during assigned lab hours suffer when compared to peers who don't need access to the lab to work on their (nonrobot) programs [1]." In other words, IPRE discovered that the convenience of the robots' portability must extend also to the use of the robots: students needed to be able to take the robots with them and use them anywhere for the robots to be an effective motivational tool.

With these key ideas and concepts in mind, IPRE chose to use the Parallax Scribbler robots in their introductory course. They also created an upgrade dongle to upgrade and extend the functionality of the robots. To keep the robots affordable, IPRE curriculum designers considered robustness only after price, with their ultimate goal being to keep the robot/textbook cost below $150, roughly the equivalent of other introductory science textbooks.

IPRE also designed a programming infrastructure they call Myro to support their curriculum's goals. Myro's primary purpose is to allow novices to easily control a personal robot while learning basic programming concepts [1]." That is to say, Myro allows students to learn basic programming and computing concepts without forcing them to learn relatively complex robotics concepts at the same time. Myro is also a cross-platform tool, and works on all major operating systems. Myro was initially created in Python, a high-level interpreted scripting language that IPRE believes demonstrates many of their pedagogical goals on its own [1]. The authors note that Python's built-in development environment lets students start work quickly Python's interpreter allows students to explore easily, without the limit of the save-compile-run routine so well known to veteran programmers.

The IPRE curriculum designers also work to keep each programming assignment tied to the robot and a physical problem that it must solve [1]," to underscore the fact that CS is not merely the writing of code, but a tool that can be used in general problem solving. In fact, they believe the most important aspect of their design to be embedding robots in the introductory course in a way that seems natural and inviting for students. They note that doing so required rethinking of the traditional sequence of topics presented in CS1 [1]."

IPRE has begun using their designs in multiple classes, though at the time of the writing of this paper, they were not very far into the analysis stage of their work, and no results were publicized in this document. Despite this, the first IPRE Annual Report is available on the IPRE website, and the results it reports are mixed. Based on surveys issued throughout the course, both at Bryn Mawr College, in a section comprised mostly of Computer Science majors, and at Georgia Tech, in a section of non-CS majors, students found the robots easy to use, and were comfortable working with the robots. While computer science majors were intrigued by the robots, and became more excited about computing, non-majors were not as likely to find the course important, or to want to work with robots again. Non-majors in the robot course at Georgia Tech were less likely to discuss their assignments with other students not enrolled in the course than those students in non-robot sections of the course [3].

IPRE does not offer any firm analysis of their results, but it seems clear enough that their approach is at least somewhat successful in retaining and increasing the interest of students who intend to major in Computer Science. It is unfortunate that non-CS majors were less interested in computing upon completion of the course, but perhaps refinement of the software and hardware used in the course will change that. No real conclusions can be drawn until the end of the three-year experiment.

III. Approach

Although the goals of the Bryn Mawr students had much in common with ours – we both sought to increase awareness of and interest in computing – their approach and ours were different out of necessity. The Bryn Mawr students worked for only eight weeks, instead of a semester. Also, because middle schoolers and college students approach problems and learn differently, the method of teaching the two groups must be different. This does not mean that either approach was better. In some ways, Bryn Mawr did what we are attempting, but they worked on a smaller scale, with younger students, and as such, they had to approach their work in a different manner than we did.

There were several key differences and similarities between our approach and that of the IPRE course designers. Both approaches aim to increase the interest of students in Computer Science by using the Parallax Scribbler robots. The IPRE approach uses Python, but we use C++. Although C++ lacks the advantage of Python's interpreter, it does share the key feature defined by the IPRE approach of a low floor and a high ceiling. The merits of using Python or C++ in introductory Computer Science courses is an ongoing debate that did not, actually, figure in our decision to use C++ here. In this experiment, all CS 102 students will attend the same lectures. One lab section will work with the robots, while the other sections will work on the normal labs. The existing introductory courses at the University of Tennessee use C++, and so our labs must also be based in that language. It is simply not possible to change languages at this time without disrupting the entire structure of the courses in the UT EECS department.

As mentioned in the summary of related work, the designers of the IPRE course discovered it was necessary to restructure the introductory course, and changed the order in which topics are presented. Although that would make sense, and should be looked into as an option if this course is successful, it was not possible for us to radically change the order. As previously noted, all students will attend the same lectures, making it impractical to alter the current syllabus. We did add some additional explanation of basic computing concepts to the labs, if it was necessary to introduce a concept slightly ahead of schedule, but for the most part, we tried to avoid using any topics that had not been discussed during the lecture period. All explanation of robotics concepts and use of the C++ Myro infrastructure was confined to appropriate sections in the labs.

Because the initial experiment will not be complete until the end of the UT Fall 2008 Semester, we needed some way to evaluate the labs we had created before that point. With that end in mind, we designed several criteria and rules to use both in the creation and evaluation of the labs:

- opportunity: the labs must provide students with the opportunity to practice most of the basic CS concepts covered in lecture
- non-displacement: use of the robots/robotics concepts in the labs must not displace the CS concepts students will need in future courses
- utility: labs should avoid using the robots for the sake of using them – if there is not a logical way to use the robots to teach a computing concept, the robots should be left out of that lab
- clarity: labs should be easy to understand, and their instructions should be easy to understand

A. *Opportunity*

To truly understand a how a computing concept works, students need the chance to actually implement that concept. For example, it is one thing to be told that arrays are zero-indexed, and another thing to use arrays and remember that you cannot access an element at array[size_of_array], because that element does not exist. Students in non-robot sections of the lab will have the opportunity to practice most of the concepts they learn in class. As many students enrolled in CS 102 will go on to take the next course in the sequence, we must take steps to ensure students in the robot section will not be at a disadvantage compared to those in regular sections

B. *Non-Displacement*

This criterion is closely related to the first, Opportunity. To get students excited about CS by using robots, we must teach them some robotics concepts. They must also learn to interact with the robots using the C++ version of the Myro infrastructure (created by a UT grad student). Students in this section will have more to learn than other students, so it is important that we find a balance between learning C++ and basic computing concepts, and learning the quirks and concepts of the Scribbler robots and their interface.

C. *Utility*

While our aim was to use the robots in every lab and, as IPRE did, to incorporate a problem for the robot to solve into each lab, we also made it a goal not to use the robot in a stupid way. That is to say, if a situation arose in which we could not think of a good problem for the robot to solve, and we could not incorporate the robots in another logical way, we would leave the robot out of all or part of that lab.

D. *Clarity*

The professor in charge of CS 102 is rewriting several of the existing labs, partially because they are unclear. In their current state, it can take more than one reading of a set of instructions to determine what, exactly, is required. With that in mind, we set the criterion of clarity in place: students should be able to easily tell after one (careful) reading what is required, and what they will have to do. The style of the labs should be simple, and the writing should be clear. Multiple proofreaders should look at each lab, to keep spelling and grammar mistakes to a minimum. Students should not have to struggle with anything other than their code to complete a lab.

IV. Results and Analysis

We worked on our labs for eight weeks this Summer. We created nine labs that students will work on in the Fall Semester. One lab will span two weeks, the others are all intended to be completed within a one week period. We also created a cascading style-sheet (CSS) to display the labs online. What follows is an analysis of the labs according to the criteria listed above:

A. *Opportunity*

We believe we were successful in creating opportunities for students to practice almost all of the basic computing concepts covered during the lecture periods. We did not cover every topic directly for example, there is no lab that deals directly with vectors; however, the students will use vectors when they use the C++ version of Myro, as several functions have vector as a return type. The students get ample opportunities to practice most C++ concepts. The only thing they may discover they lack is practice in creating classes. They do create one class, but we had a great deal of difficulty finding ways for the students to use classes and robots while still conforming to our utility requirement. In the end, we chose to have them create one complicated class. This is not as negative an effect as it initially seems. Students do not use C++ again until their object-oriented programming course, where they will undoubtedly get more experience.

B. *Non-Displacement*

Following this criterion turned out to be more complicated than we anticipated, but not impossible. One example lab where we encountered a struggle was in lab 3, in which students implement several of Braitenberg's vehicles as part of an exploration of programming robots to emulate lifelike behavior giving the robots a low-level form of artificial intelligence. Figuring out how to explain the concepts of Braitenberg vehicles and remind students of the programming concepts they would need without overwhelming students with too much information was difficult. We solved the problem by giving a non-detailed explanation of Braitenberg's vehicles, with a link to a source of more information. In this way, students are not overwhelmed with a lot of new information. We also chose to give them detailed guidance for most of the lab, so that they could spend more time implementing the vehicles and learning programming concepts than struggling to understand a somewhat obtuse robotics concept. (This also complies with our clarity criterion.)

C. *Utility*

There was only one instance in which we did not use the robot for part of lab because we could not find a logical, intelligent way to do so. It occurred in the first part of lab two.

We thought it was important that students in the robot section get the same early introduction to doing arithmetic in C++, but could not think of a way to do so with the robots using the students' knowledge at that point. We chose to assign the same temperature conversion program that is in the existing labs. We initially thought we would have to do something similar to introduce students to the concept of 2D arrays, but we came up with the idea to use the robots' cameras and images as a logical vehicle. We created lab 7, the panorama lab, in which students must manipulate 2D arrays to edit images and produce a panoramic view based on several pictures taken by the robot. In every other instance, a way to use the robots in an intelligent problem was relatively easy to come up with.

D. Clarity

We read over each lab multiple times, and had the course's TA read through them all, as well. The lecturer for the course will read and correct each lab before assigning it to the students. We added a method of notation to each lab: at the end of each section, we have added a subsection labeled Do This. The idea behind this is that it makes instructions doubly clear. The Do This subsection summarizes in one or two sentences what the students must do for each part of the lab. As previously mentioned, we also created the cascading style-sheet to make displaying the labs easy and clear. The style-sheet also makes it easy to maintain a consistent appearance throughout the labs.

V. Conclusion

We believe that we have created labs that will provide students with the opportunity to put what they learn in lectures and from reading their textbooks into practice, with activities that will let them explore less boring aspects of Computer Science and of computing as a whole. We hope that the labs will show them that Computer Science is not just about cranking out lines of code, but about solving problems, big and small – in other words, labs that will show them that Computer Science is not a boring or useless discipline, but one that is exciting, interesting, and worth devoting a career to.

VI. Future Work

University of Tennessee Computer Science faculty will use surveys and other assessment tools to evaluate the use of robots in their introductory class over the course of the next two semesters. Based on the results from those tools, and comparisons between the sections of class that did not use robots and those results, the faculty will determine the success of the experiment. The University will publish the results. If the experiment is considered successful, the EECS department will begin work to add robotic elements to subsequent Computer Science and Engineering courses.

References

[1] Tucker Balch et al. Designing personal robots for education: Hardware, software, and curriculum. *IEEE Pervasive Computing*, 7(2):5–9, Apr-Jun 2008.

[2] Mansi Gupta, Marwa Nur Muhammad, and Shikha Prashad. Robots byte in: An exploration of computer science education in middle school. Found on IPRE website: http://roboteducation.org/files/CREU_Final_Report.pdf.

[3] The institute for personal robots in education 2007 annual report. Annual report, Institute for Personal Robots in Education, 2007.