Education & Training

Editor: Scott F. Midkiff ■ Virginia Tech ■ midkiff@vt.edu

Designing Personal Robots for Education: Hardware, Software, and Curriculum

Tucker Balch, Jay Summet, Doug Blank, Deepak Kumar, Mark Guzdial, Keith O'Hara, Daniel Walker, Monica Sweat, Gaurav Gupta, Stewart Tansley, Jared Jackson, Mansi Gupta, Marwa Nur Muhammad, Shikha Prashad, Natasha Eilbert, and Ashley Gavin

EDITOR'S INTRO

An exciting new initiative at Georgia Tech and Bryn Mawr College is using personal robots both to motivate students and to serve as the primary programming platform for the Computer Science 1 curriculum. Here, the authors introduce the initiative and outline plans for the future. I welcome your comments and suggestions for future columns. I can be reached at midkiff@vt.edu. —*Scott Midkiff*

QUICK FACTS

Course: Computer Science 1 (CS1)
Level: Undergraduate
Institutions: Georgia Tech, Bryn Mawr College
Contact: Tucker Balch (tucker@cc.gatech.edu)
URL: www.roboteducation.org

As a field, computer science faces a problem. From 2000 to 2004, the percentage of first-year undergraduates planning to major in CS declined by more than 60 percent (see the "Declining Interest in Computer Science" sidebar).[1]

To attract more students, the introductory CS curriculum must be motivating and relevant. CS courses that are set in a motivating context (for example, using multimedia, gaming, or robotics) can excite students and get them hooked.

Other researchers have worked on introductory programming classes with robots as well as introduction to robotics classes (http://myro.roboteducation.org/robobiblio). We didn't want to create a robotics course but rather an introductory CS course based on robots. Introduced properly, robots make visible and tangible those aspects of CS that are often hidden behind computer screens and in computer memory. To further this goal, we formed the Institute for Personal Robots in Education (IPRE), a joint effort between Georgia Tech and Bryn Mawr College and sponsored by Microsoft Research (www.roboteducation.org). This article discusses the first-year results of a three-year project.

INSTITUTE FOR PERSONAL ROBOTS IN EDUCATION

The IPRE curriculum is based on several key ideas:

- use of a personal robot;
- tools with "a low floor and a high ceiling"—that is, they're easy for a novice to learn but have enough power so that an expert will continue to use them; and
- stretching the student's perceptions of computing.

An essential element of our approach is that every student should have his or her own robot. With regard to our educational vision, the most important features of these robots are that they're inexpensive, robust, and convenient (portable), and they take full advantage of the students' computers for developing, debugging, and running programs that control the robot.[2] 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.[3] A personal robot is small enough to be carried to lab and class, and individual ownership lets the students work when and where they choose.

ROBOT HARDWARE

We use the Parallax Scribbler (www.scribblerrobot.com) with a custom IPRE

EDUCATION & TRAINING

DECLINING INTEREST IN COMPUTER SCIENCE

Data collected by the US National Science Foundation indicates that while the number of Advanced Placement tests taken by high school students overall has increased by 33 percent, the number of AP tests in computer science (CS) has dropped by 20 percent. In fact, the number of AP tests taken has gone up in every field except CS, which is no longer viewed as a "hot" career. A second but equally important problem is retention—on average, at least half of the college students majoring in CS withdraw from the major, and most of these students withdraw during the first year.[1] In contrast, according to the Bureau of Labor Statistics, the number of US jobs for IT professionals will grow by more than 1.2 million in the next decade.

A variety of factors contribute to students' lack of interest and participation. Many students see computing as an asocial activity that's best suited to men drawn to computers from an early age.[2] Many female undergraduates in CS programs reported feeling socially isolated from and less capable than their male peers, particularly if they didn't have extensive computing experience prior to college.[2] In addition, many women and underrepresented minority students view CS as tedious, boring, and irrelevant,[2] with little room for creativity.[3] Beginning students have difficulty seeing the real-world relevance of topics such as byte representations and algorithmic efficiency.[2] Faced with a difficult curriculum, an unwelcoming culture, and course lectures and assignments that seem irrelevant to real-world problems, many women and underrepresented minorities choose not to pursue CS.

REFERENCES

1. E. Seymour and N.M. Hewitt, *Talking about Leaving: Why Undergraduates Leave the Sciences*, Westview Press, 2001.
2. J. Margolis and A. Fisher, *Unlocking the Clubhouse: Women in Computing*, MIT Press, 2002.
3. S.L. Pfleeger et al., "Increasing the Enrollment of Women in Computer Science," *SIGCSE Bull.*, vol. 33, no. 1, 2001, pp. 386–387.

add-on board (a dongle) as our robot platform (see figure 1). The robot has five infrared sensors, three photo sensors, a low- to medium-resolution color camera, programmable LEDs, a dual-tone speaker, and a Bluetooth wireless communications link (built-in or through a USB Bluetooth dongle). Additionally, the robot can communicate using infrared and can detect colored regions onboard.

Because we wanted all the students to have their own personal robot that they could carry between dorms and classrooms, our primary consideration when selecting a hardware platform was cost, with robustness as a secondary consideration. Ideally, we'd like the robot-and-textbook combination to cost less than US$150 to match the average price of a typical introductory science textbook.

Each student's computer sends commands to his or her robot and receives sensor values back via the wireless serial link. Not executing the programs on the robot allows for easier debugging and user-program interaction via the keyboard and screen. It also allows for less expensive hardware and greater programming capabilities using the advanced processing power of the student's laptop. For example, the speak ("Hello World") command uses laptop-based text-to-speech synthesis, as the robot's speaker and microprocessor do not support general sound output. Our hardware package containing the $60 robot and the $90 custom IPRE add-on board begins to approach our target price for the course. We've also integrated support for USB game pad controllers ($10) that students can use to write interactive programs (see figure 2).

MYRO SOFTWARE

Our programming infrastructure, named Myro, was designed to directly

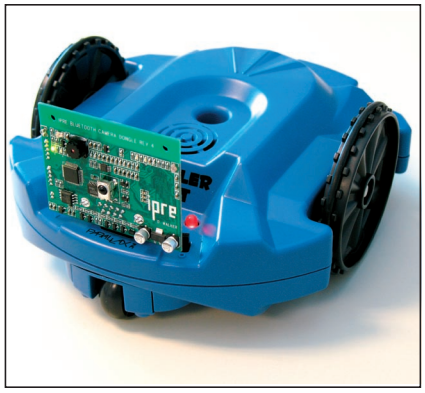

Figure 1. A Parallax Scribbler robot with an Institute for Personal Robots in Education add-on board.

support the curriculum's goals by creating an intuitive, easy-to-learn, yet powerful interface to connect each student with his or her robot. An earlier tool, Pyro, enabled students already familiar with computer science to easily learn how to control a robot by serving as a high-level programming paradigm for a wide variety of robots and sensors.[4]

In contrast, the primary design goal for Myro is to allow novices to easily control a personal robot while learning basic programming concepts. Myro is a cross-platform tool that works on Macintosh, Linux, and Windows operating systems and is written in Python. Python is a high-level interpreted scripting language that itself exhibits many of our pedagogical goals. During the pilot program, we developed a set of functions, objects, and nomenclature that helps students quickly pick up the language and syntax and jump to the heart of issues in computation. Myro is open source and developed with feedback from the academic community. It supports the Surveyor, Roomba, and Boe-Bot robots in addition to the Scribbler. An implementation of Myro using the Microsoft Robotics Studio (www.microsoft.com/robotics) is in active development and supports additional hardware.

To the student, Myro is just another Python module that can be imported, similar to the math, string, or time modules. The Myro API lets novice stu-

dents control their robot and provides easy-to-use functions for

- robot movement,
- sensor readings,
- multimedia and image processing,
- automatic web publishing,
- communication via instant messaging (Jabber client),
- music and tone generation, and
- text-to-speech translation.

As an example, figure 3 shows a complete program we wrote in Python using the Myro API.

This program loads the Myro libraries; connects to the robot; takes, records, and displays a picture; samples the brightness; and turns around a little. It repeats this for 60 seconds, after which it saves an animated image of all the pictures it took (essentially a movie of what was around the robot), computes and prints out the average observed brightness, and then says out loud that it has completed its task.

Myro provides additional easy-to-use functionality to process images using an interface based on Mark Guzdial's media computation framework.[5] Using the IDLE development environment (http://wiki.python.org/moin/IDLE), students can develop their programs interactively at first, by entering code line by line and inspecting and manipulating the results. This lets novice users quickly experiment and learn without the traditional compile-execute-debug cycle.

With a "low floor" (ease of use for novices) as our goal, we're evolving even the most basic API commands on the basis of ideas and feedback observed in our test courses and in our own research on robotics and AI. As an example, in the previous example we used a time-controlled loop with the `timeRemaining(T)` function that repeats the body of the loop for *T* seconds. In most robot behaviors that students design, it's best to run the behavior for a limited amount of time and then to stop and evaluate the behavior. Traditionally, we can do this with Python using a loop

Figure 2. Myro supports USB game pads that students can use to write interactive controllers.

```
from myro import *
init()
samples = []
pics = []
while timeRemaining(60):
  pic = takePicture()
  show(pic)
  pics.append(pic)
  samples.append(getBright("center"))
  turnLeft(0.5, 0.2)
savePicture(pics, "LookAroundMovie.gif")
avgBrightness = sum(samples)/len(samples)
print "Average brightness is", avgBrightness
speak("I have completed my mission!")
```

Figure 3. A complete program written in Python using the Myro API.

structure that checks the current time versus the starting time. A program structure that uses the `timeRemaining()` function is simpler and more appealing for beginning students.

As a second example, in our research on doing semantic analysis of natural language interactions with robots, we discovered that most imperative commands implicitly have a limited time specification. Consider the command "Move forward." Given to a human, this command doesn't imply that he or she should move forward forever. Yet, most robot commands, initial versions of Myro included, implement this request using the command `forward(defaultSpeed)`. That is, it just keeps moving until told to stop. This is a subtle semantic issue, but from a psycholinguistic standpoint, all such commands should have a time-limited behavior. We thus modified our API to have the movement commands take a duration parameter `forward(speed, duration)`. The robot moves forward at a given speed for the duration specified (in seconds) and then stops. There are several instances in the design of the Myro API where we incorporate these kinds of insights.

When first teaching the movement commands, we demonstrate the atomic, blocking versions of the commands. However, when making their robots sing and dance, the students usually ask if it's possible to have the robot beep while moving. This gives us the opportunity to explain the difference between the `forward(1,1)` and `forward(1)` commands and how the latter can be used with a `beep(1,440)` and `stop()` to cause the robot to beep at 440 Hz (A above middle C) for one second while moving forward.

CURRICULUM

Our vision is that students registering for a Computer Science 1 (CS1) course will go to the bookstore and purchase their own personal robots for approximately the cost of a textbook. Currently, the $150 cost is more expensive than most textbooks, but we expect this cost to decrease. We have let the curriculum's needs drive the selection of our robot.

Deciding to add a particular feature to the robot is driven by the need to motivate or achieve a teaching goal. The best example of this is the camera, which we added for three reasons. First, giving access to a camera makes the course more media oriented, which motivates and attracts students who care more about media than computation. Second, it provides an example 2D data structure that lets us teach array accessing and looping. And finally, it provides interesting content that the robot can post to its website and that we can use to teach basic networking and web publishing concepts.

EDUCATION & TRAINING

Figure 4. A student's robot "in costume" for a performance.

A PERSONAL ROBOT-BASED TEXTBOOK

Chapter 1. "The World of Robots" introduces the Scribbler robot and Myro software.
Chapter 2. "Robots: Personal or Otherwise" introduces Python, its development environment, and how Myro abstracts robot movements into simple Python commands.
Chapter 3. "Building Brains" introduces programming concepts, including the notion of a program and the use of names to represent values, parameters, and functions.
Chapter 4. "Sensing the World" introduces robot sensors and the data types of the values they return.
Chapter 5. "Making Decisions" introduces If statements, traditional computational examples, and the math library and other arithmetic expressions.
Chapter 6. "Behaviors" introduces the idea of programming robot behaviors using the Braitenberg paradigm—that is, the same behaviors that can be programmed using decision-making structures can be accomplished through simple mathematical transformations.
Chapter 7. "Control Paradigms" introduces the two kinds of robot control paradigms that the students have been using and discusses the advantages and shortcomings of these paradigms.
Chapter 8. "Making Music" explores sound and music.
Chapter 9. "Communication" presents an instant-messaging-like chat interface developed in Myro and web publishing.
Chapter 10. "Computing & Computation" introduces the notions of an algorithm, problem solving in the programming process, and the limits of computation.
Chapter 11. "Applications of Robots" presents the current state of the art of robot applications and looks into the future.

We want the students to use tools (a programming language and an environment) that are easy to learn and allow them to get started quickly. Python's built-in development environment meets this goal, as students can issue commands that are interpreted immediately, letting the students explore one command at a time. At the same time, we don't want to limit students to an environment or language designed specifically for CS1; the environment must be scalable and realistic. Such environments let students easily carry over concepts they learn in CS1 into more advanced programming environments, and they'll realize they can potentially transition the skills they're learning into a career. By showing students examples of real companies that are using Python and are looking to hire programmers, we show them that Python has a high ceiling.

We also strive to keep each programming assignment tied to the robot and a physical problem that it must solve (escape a maze, seek a light, give a performance). This highlights the fact that computer science is not simply programming but a tool used in general problem solving. We try to make computing a medium for creativity and social activity. We encourage collaboration on everything in the class except exams, and we depict and evaluate all robot exercises as performances or individual challenges rather than as competitions among the students. These open-ended assignments deliberately value creativity and story telling in addition to presenting technical challenges. We grade students not only on their programs' technical correctness but also on overall presentation style, including set pieces and decorations (see figure 4).

Our curriculum development has produced Myro reference materials, extensive materials for instructors, and an 11-chapter web-based textbook called *Computer Science 1—An Introduction with Robots* (see the sidebar "A Personal Robot-Based Textbook"). The most important aspect of curriculum design is to embed the personal robot into a CS1 course in a way that seems natural and inviting for students. This required a fresh approach to the overall CS1 syllabus and rethinking of the traditional sequence of topics presented in CS1. While the class deviates from traditional approaches such as the IEEE Computer Society/ACM standard curriculum (www.sigcse.org/cc2001/cs-introductory-courses.html), its overall treatment of topics provides comprehensive coverage of traditional CS1 concepts. In fact, in many ways, it goes beyond the traditional notion of a CS1 syllabus. Yet, the key

driving factor in this curriculum's design is the exploratory and engaging nature of robots.

Our initial prototype hardware is robust and inexpensive enough to test our curriculum and software on multiple classes. Our curriculum, software, and hardware platform are improving as we iterate and teach more classes. Although we're still in the early stages of our three-year program, we're beginning our dissemination and assessment process.

Over the next two years, we intend to improve our hardware platform and arrange for its manufacture and sale, allowing interested parties to purchase a ready-to-use robot. We'll also publish a robot and software standard, enabling third parties to build their own Myro-compatible robots and software. We're continuing development of the Myro implementation by leveraging the Microsoft Robotics Studio's capabilities, including additional programming languages through the .NET Common Language Runtime in addition to Iron Python. We'll continue development of our curriculum, teachers' manuals, sample homework, and textbook.

Much of our work is already visible on the Web, including the Myro software and our textbook (www.roboteducation.org). We're writing the latter via a wiki that has public read access. We encourage you to consider working with us—we're actively looking for multiple schools throughout the US to evaluate our curriculum.

ACKNOWLEDGMENTS

This work was funded in part by Microsoft Research.

REFERENCES

1. J. Vegso, "Interest in CS as a Major Drops among Incoming Freshmen," *Computing Research News*, vol. 17, no. 3, 2005, www.cra.org/CRN/articles/may05/vegso.html.
2. D. Kumar et al., "Engaging Computing Students with AI and Robotics," *Using AI to Motivate Greater Participation in Computer Science: Papers from the AAAI Spring Symp.*, tech. report SS-08-08, AAAI Press, 2008.
3. B. Fagin and L. Merkle, "Measuring the Effectiveness of Robots in Teaching Computer Science," *SIGCSE Bull.*, vol. 35, no. 1, 2003, pp. 307–311.
4. D. Blank et al., "Pyro: A Python-Based Versatile Programming Environment for Teaching Robotics," *ACM J. Educational Resources in Computing* (JERIC 05), vol. 3, no. 4, ACM Press, 2005.
5. M. Guzdial, *Introduction to Computing and Programming in Python: A Multimedia Approach*, Prentice Hall, 2004.

Tucker Balch is IPRE director and an associate professor in interactive and intelligent computing at Georgia Tech University. Contact him at tucker@cc.gatech.edu.

Jay Summet is a postdoctoral fellow in curriculum development and evaluation at Georgia Tech University. Contact him at summetj@cc.gatech.edu.

Doug Blank is codirector and lead software designer for IPRE and an associate professor at Bryn Mawr College. Contact him at dblank@cs.brynmawr.edu.

Deepak Kumar is a a co-principal investigator for curricula for IPRE and a professor and chair of the Computer Science Department at Bryn Mawr College. Contact him at dkumar@cs.brynmawr.edu .

Mark Guzdial is a co-principal investigator for curricula at Georgia Tech. Contact him at guzdial@cc.gatech.edu.

Keith O'Hara is a doctoral candidate and an instructor of robot hardware and software design at Georgia Tech. Contact him at kjohara@cc.gatech.edu.

Daniel Walker is a research scientist and lead hardware design engineer at Georgia Tech. Contact him at danielbw@cc.gatech.edu.

Monica Sweat is an instructor in curriculum evaluation at Georgia Tech. Contact her at sweat@cc.gatech.edu.

Gaurav Gupta is an IPRE fellow and a master's degree candidate at Georgia Tech. Contact him at gaurav@cc.gatech.edu.

Stewart Tansley is program manager at Microsoft Research. Contact him at stansley@microsoft.com.

Jared Jackson is a research software development engineer in the Extended Research and Programs group at Microsoft Research.

Mansi Gupta is a student at Bryn Mawr College. Contact her at mgupta@brynmawr.edu.

Marwa Nur Muhammad is a student at Bryn Mawr College. Contact her at mmuhammad@brynmawr.edu.

Shikha Prashad is a student at Bryn Mawr College. Contact her at sprashad@brynmawr.edu.

Natasha Eilbert is a 2007-2008 IPRE fellow and a student at Bryn Mawr College. Contact her at neilbert@myro.roboteducation.org.

Ashley Gavin is an IPRE fellow and a student at Bryn Mawr College. Contact her at agavin@brynmawr.edu.