Personalizing CS1 with Robots

Jay Summet[†], Deepak Kumar[‡], Keith O'Hara[†], Daniel Walker[†], Lijun Ni[†], Doug Blank[‡], Tucker Balch[†]

[†] College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30308, USA
{summetj,kjohara,tucker}@cc.gatech.edu

[‡] Department of Computer Science
Bryn Mawr College
101 N. Merion Ave.
Bryn Mawr, PA, 19010, USA
{dkumar,dblank}@cs.brynmawr.edu

ABSTRACT

We have developed a CS1 curriculum that uses a robotics context to teach introductory programming [1]. Core to our approach is that each student has their own personal robot. Our robot and software have been specifically developed to support the needs of a CS1 curriculum. We frame traditional problems (robot control) in terms that are personal, relevant, and fun. Initial trial classes have shown that our approach is successful and adaptable.

Categories and Subject Descriptors

K.3.2 [**Computers & Education**]: Computer & Information Science Education—*computer science education*

General Terms

Human Factors, Languages, Experimentation

Keywords

Robots, CS1, Pedagogy

1. INTRODUCTION

We have developed a novel CS1 curriculum using personal robots as a learning context. The personal robot provides students with a tangible artifact that embodies their programs' behavior and motivates them to learn. In the last year we have taught CS1 using robots to over two hundred and fifty students in seven classes ranging in size from 12 to 104 at three different institutions. Our curriculum approach, including textbook [10], assignments, robot hardware, and software were iteratively developed together, and this paper presents our progress and early results.

Using a form of context (such as Media Computation [7]) allows students to quickly see a tangible and relevant result from their efforts. However, media computation and other visual programming systems such as Alice [5] or robot simulators such as Josef, Karel and Logo Turtle graphics [15, 14, 13] still live "in the computer". Using robots to teach problem solving, programming and fundamental computer science concepts is not novel [13, 6]. Recently, robots have also been used in upper division classes, for example, user interface programming [4] and artificial intelligence [9, 2].

Figure 1: Student owned robots decorated for the robot performances assignment.

Our approach differs from many others due to three core tenets. First, we believe that each student should have their own personal robot. Second, our curriculum is driven by the robot, as opposed to traditional programming concepts. Third, the robot acts as dumb peripheral, and student programs are executed on the computer within an interactive development environment.

One robot per student is a core tenet of our approach, and this requirement has driven many curriculum and hardware design decisions. Because every student will have their own robot, the robots must be inexpensive, portable, and robust. Fagin and Merkle previously had negative results teaching CS1 with robots and believe the most significant factor was limited hands-on time during assigned lab times with lab based robots. In their words, "Students in robotics sections must run and debug their programs on robots during assigned lab times, and are therefore deprived of both reflective time and the rapid compile-run-debug cycle outside of class that is an important part of the learning process." [6]

The personal robot approach gives students the freedom to choose where they work, and sometimes play or showoff with the robots. Students take their robots home to

SIGCSE'09, March 3–7, 2009, Chattanooga, Tennessee, USA.

Figure 2: Parallax Scribbler with the Lancet Fluke upgrade board that provides Bluetooth, a camera, additional sensors, and LEDs.

show what they have learned at college, gather in groups to work on assignments, and decorate their robot for performances. We have been convinced by anecdotal reports that our robots attract other students into the course because they are seen in public.

Although we cover the same topics as the ACM model CS1 (Imperative) curriculum, we use a "robot centric" approach. For example, our "Robot Brains" chapter covers conditional statements, and students learn about two dimensional array manipulations when manipulating arrays of pixels from the robot's camera. In addition, the features on the robot have been driven by the needs of a CS1 class. This in contrast to typical consumer robots or research robots which have very different requirements and markets. For instance, the ability for the robot to be expressive with sounds, LEDs, and motion trumps highly accurate positioning.

The final core tenet of our approach is that students write, execute, and debug programs on desktop or laptop computers. Our robots, although actually containing multiple CPU's running custom software, are presented to students as a dumb peripherals. The robots respond to commands issued by function calls in the student's code and return sensed data as return values. This eases debugging and the students have a clear mental model of the locus of execution. Additionally, the desktop computer can provide extra features (e.g. text-to-speech synthesis, complicated image processing, graphics) that the robot might not be able to provide on-board.

2. THE ROBOT KIT

The personal robot we are using is a commercially available Scribbler robot whose features are augmented by additional plug-in hardware, called the Lancet Fluke that we have designed and developed as a part of this project (See Figure 2). This provides the following features on the robot: two motors capable of providing a full range of variable speed floor motions, a suite of sensors (light (3), proximity/obstacle (5), stall (1), and line (2)), programmable LEDS (3), a two-tone speaker, a Class 1 Bluetooth wireless (100 meter range), a color video camera, and a Pen port. Each student gets a CS1 Robot Kit which also includes a gamepad controller, USB Bluetooth dongle for the computer, a flashlight, color pens, a Myro software CD, the text, and a flashy lunchbox carrying case. The Python-based Myro software provided in the kit enables access through simple and intuitive library commands (see Section 3). To ease exploration and debugging, the commands can be issued to the robot interactively at an interpreter prompt or through a complete program running on the host computer. The robot (with Fluke) costs $150 and the remainder of the kit costs an additional $40.

The robot is fully assembled and ready to run. We deliberately chose a fully assembled approach as opposed to build-your-own kits such as the Lego Mindstorms™ because previous work has shown that even advanced computer science students feel that they spend too much time on the "design of robot chassis" when using Mindstorms™ [8]. Because we are not teaching a multidisciplinary class such as that by Weinberg et al. [16], presenting students with a fully assembled robot removes the unnecessary complications of mechanical assembly, allowing them to focus on computer science concepts.

An earlier version of our prototype robot used the Scribbler robot and a commercially available Bluetooth serial port (wireless serial cable) adapter. In pilot classes we found that having remote control of the Scribbler robot provided students with motivation to learn basic programming constructs such as calculation, iteration, functions/encapsulation, program flow/conditionals, and simple data types. However, more advanced data types, such as two dimensional arrays, were hard to motivate using only the Scribbler Robot's built in sensors. To provide more opportunities for learning, the Lancet Fluke circuit board which adds Bluetooth, additional sensors and the color camera was developed. The addition of camera images to the other data types provided by the robot sensors greatly motivates students, and allows for more complicated assignments that exercise multi-dimensional array manipulations. Additionally, students respond very positively to assignments asking them to shoot a robot movie, create special effects, find and follow a red ball, or generate a web-page that shows the output of the current robot sensors, including the camera image.

3. MYRO SOFTWARE

We use Python as our programming language in CS1 because it is a syntactically clean, dynamic language that allows students to experiment in a live environment with a minimum of syntactic difficulties. For a supporting argument, see [11]. Students interact directly with Myro, our Python module that includes all robot control functions and additional helper functions. Along with robot control, Myro enables students to easily show images from the camera, create simple graphic displays, and GUI dialogs to ask the user for input. Myro implements lessons learned from Pyro, an earlier Python based robotic control environment [3]. Students can use Myro on Linux, MacOS X, and Windows. For an example of a program using Myro, see Figure 3

Although Myro supports teaching in an Objects First manner (by creating a robot object and calling methods such as robot.forward(1,1)) our curriculum uses the syntactic simplicity of Python to start with simple expressions and function calls. Because Python is interactive, beginning stu-

```
from myro import *
init()

while timeRemaining(60):
    ir = getObstacle('center')
    if ir > 0:
        p = takePicture()
        beep(1, 440)
        show(p)
```

Figure 3: Robot Security Guard: The robot beeps and takes a picture when its IR sensor is tripped.

dents can experiment with the robot quickly simply by typing commands into the IDLE interpreter window.

4. CURRICULUM

We use the robot to motivate learning and stress collaborative group assignments to build social bonds early. Computer Science concepts are introduced as needed to make the robot perform increasingly interesting and sophisticated tasks. We want to give students the correct perception that computer science is a creative and collaborative field, and use creative group assignments as a way to build social bonds between students. We hope that these social bonds will give students a support network to be successful in CS1 and meet people who they may consider taking CS2 with. The creative, open-ended assignments also portray Computer Science as a field where you can use the programming skills learned as a creative tool. We specifically avoid robot competitions as we want our CS classes to have a collaborative and non-confrontational atmosphere.

4.1 A Robot Approach to CS1

In our first lab, we help the students connect to their robot and walk them through simple movement and output functions (forward, backward, turnLeft, turnRight, beep, speak). The first assignment asks the students to make their robot play music and dance. In class, we explain the sequential ordering of statements, program flow, and encapsulation in functions to support the Music & Dance assignment.

While students are working on the Dance exercise, we begin to cover variable assignment, simple calculations (+,-,/,*), integer and float data types, and user input and output. The dynamic nature of Python makes it easy for students to experiment with these functions in the command interpreter during lecture and in an associated lab.

A later assignment asks the students to implement a simple behavior (closed loop control) to make their robot flee or seek light or avoid obstacles, etc. This assignment requires the introduction of conditional (if, if/else, if/elif/else) statements, relational operators (>, <, ==, !=) and the Boolean data type.

Although our curriculum covers the traditional CS1 concepts, we always motivate their introduction by requiring them to make the robots perform as desired. Our textbook does not have chapters on data types, functions or iteration. Instead, it introduces students to their robot and the concepts they need to control it with programs. We work around the Scribbler's lack of precise motor control by managing student expectations and giving assignments that rely upon closed-loop control or do not require extreme precision. Students who want to make the robot draw a precise figure quickly find out that in the real world, motors do not always work as cleanly as the author of a linear control script would like, and decision making and evaluation of sensor feedback is required [12].

4.2 Group Assignments

Several times in the semester we assign the students into three person teams to work on an extended creative assignment. Because each student has a robot, we make the assignments require multiple robots, ensuring that the students work together. We give overall specifications and a grading rubric to the students, but allow them wide latitude to the content of their creations.

One example is the movie assignment. Students are given a two part assignment where the overall goal is to produce a movie using their robots as actors and camera-bots. The first part of the assignment teaches data representation concepts and multi-dimensional array manipulations, but the students believe they are creating special effects for their movies by playing with images captured by the robot's camera. The assignment is open-ended in that we specify various "standard" special effects such as wipes, cross-fades, red-tint, negate, and reverse, each with associated point values and challenge the group to accumulate 100 points. We also give the groups the option of creating their own special effect. In the second part of the assignment, the students shoot, assemble, and display their movie using the special effects previously coded. Note that because Myro does not include any "video" primitives, students must write code to capture multiple images, store them in lists, and play them back, along with any scene editing or re-arranging that they wish to perform.

Another example is our end of class Robot Performances (see Figure 1). Unlike the movie assignment, the robot performances are done live, with an audience invited from the general college community. Some groups used the robot sensors or audience input in addition to giving a scripted performance.

4.3 Class Atmosphere

Television shows such as Battle-Bots, which show robots using weapons of destruction on each other, promote a societal view of robots as masculine, competitive, and destructive. In contrast, we make great efforts to show the expressive and interactive side of robotics, working to make all students feel comfortable. None of our assignments have students competing directly against each other or completing a timed "race" where direct comparison between students is possible. We do have an "Escape the Maze" assignment; however, it is posed as a challenge for each student to program their robot to successfully escape a maze, with no bonus given for speed. Although performances are graded, we do not rank them, and give multiple "awards" to the best groups for different aspects of the performance ("best costumes", "best story", "most entertaining", "technical difficulty"). During the movie making assignment, robot "Oscars" are given out to outstanding movies in recognition of different aspects such as special effects, story, or robot acting.

5. DEPLOYMENT, ROBUSTNESS, COST

Our default deployment model is that students will purchase their own robot for the class, similar to a textbook. The current 150 USD price makes the robot purchase com-

parable to that of a multi-semester science textbook such as used in Calculus, Chemistry, or Physics. To ease the cost to the student, the robot kit includes the free textbook as a PDF on the CD, or students can download it from our website. At the Georgia Institute of Technology, the robot was stocked at the bookstores and individually owned by students. This also resulted in a secondary market for used robots, similar to used textbooks. Our goal is to continue to reduce the price of the robot. Several institutions have also purchased a group of robots and checked them out to students for the semester (similar to a chemistry lab equipment check-out model).

Considering that students carry them around in backpacks, our current robot hardware is surprisingly robust. We experienced about a 2-3% failure rate, comparable with most consumer electronics. Having a few loaner robots to rent out while a student waits for a warranty repair greatly reduces stress levels. Perhaps the most successful and easiest strategy makes use of the fact that many assignments are done in groups. We deliberately force the students into (different) groups so that they have a network of social contacts within the class. For most group assignments, one non-working robot can easily be worked around, and for individual assignments, students can borrow a friend's robot to test their code before they turn it in to be graded.

At Georgia Tech, where all students are required to have their own laptops, computer failures caused more problems than robot failures. Because we count on all students to provide their own laptops, our labs have many power strips, tables, and robot obstacle courses, but no lab computers. Several students had problems with their personal laptops ranging from a cracked motherboard to "clicking on a link in an email" that required an OS re-install. As with robot failure, the best workaround involved calling upon the social network they had established in group assignments to complete the assignments using a friend's computer. At Bryn Mawr College, the impact of non-uniform computers was mitigated by using lab computers pre-loaded with the Myro software.

One final environmental and monetary cost is batteries. We have found that each student typically uses 18-24 AA batteries over the course of a semester. We encourage students to purchase NiMH rechargeable batteries, and provide quick-chargers in our robot laboratories. The next version of our robot will include integrated rechargeable batteries.

6. EARLY RESULTS

Starting in the spring semester of 2007, we taught three CS1 classes involving two schools and 81 students using our version 1 prototype (without a camera). Informed by our our experiences in these classes, we developed our current, version 2 prototype system. Beginning in Fall semester 2007, four CS1 classes involving 178 students and three schools have been taught using the personal robot approach and the version 2 prototype. Class sizes ranged from 12 to 104 students. In addition to new hardware and software, our curriculum material, including textbook, lecture notes & slides, and assignments have undergone continuous improvement.

At Georgia Tech, 90.97% (131 of 144 students) were successful (grade of A, B, or C) in our our Fall 2007 CS1-Robots classes. These students consisted of both CS majors and non-majors. For comparison, the success rate in the Fall 2007 non-robots CS1 class, which consisted of non-majors, was 85.71% (78 of 91 students). Our other CS1 classes, CS1-MediaComp (open to all majors other then engineering and CS students) had a success rate of 73.06% (179 of 245 students), and our CS1-Matlab classes (offered exclusively to engineering students) success rate was 69.16% (740 of 1070 students). Analysis on the underlying grade distributions of the Robots and Non-Robots class results in a χ^2 value of 12.0 and p = 0.035.

Fall '07 Class	Success Rate	Students
Robots	90.97%	131 of 144
NonRobots	85.71%	78 of 91
MediaComp	73.06%	179 of 245
MATLAB	69.16%	740 of 1070

A final exam taken by students in the CS1-Robots and CS1-NonRobots classes had five shared questions that did not require experience with the robot but in some cases used "robotic" situations. On average, the CS1-Robots students did 10% better than the CS1-NonRobots students, and the difference between them on four of the five questions was statistically significant (p <= 0.015). However, as we did not assign students randomly to each class, these early results are not a controlled study and simply indicate that the Robots approach does not appear to be doing harm.

The average annual enrollment in Data Structures (CS2) at Bryn Mawr College from 1995 to 2006 was 7.45 students. The 2007 and 2008 classes, after the introduction of CS1 with Robots, averaged 17.5 students, more than doubling the average enrollment. We are currently implementing a longitudinal study (See Section 7) to determine the actual effect our curriculum has on student retention and preparation for CS2.

In the summer of 2008 we taught the CS1 with Robots curriculum to 50 instructors at two Faculty Enrichment workshops. We received surveys from fourteen of twenty six participants from the first workshop and they all indicated that they felt that the CS1 with robots approach would help attract students to their classes, that the approach would be a "real" computer science class, and that the class fit well with the general goals of introductory CS courses.

7. FUTURE WORK

Myro 2 is implemented in Python, which allows it to run on Windows, Macintosh, and Linux computers and made development easy. We have been very happy with using Python to teach CS1, but many institutions, and especially high schools, are locked into teaching with other languages such as Java and C++. Very little in the Myro API requires Python, and we are working on implementing the Myro API in a combination of IronPython and C#, two languages that run on the .Net/Mono Common Language Runtime (CLR).

By implementing Myro 3 using the CLR any language running on the CLR can link against and call the Myro 3 API without recompilation. This allows us to implement Myro3 API functionality in any CLR language, and allows instructors to use the Myro API with their language of choice. Currently, many standard and exotic languages are supported on .Net or the open-source alternative, Mono. Because Mono runs on Mac OSX and Linux, we will retain the platform neutrality of the Myro API. On the Windows platform, we will be able to leverage the Microsoft Robotics Development Studio Robot Simulator. This will allow students running Windows to program and run a simulated robot with accurate physics and an advanced visualization.

We have developed a longitudinal tracking study that examines student retention and performance in CS2 after the complete CS1 with Robots, as well as recruitment of non-CS majors into the CS program. At Georgia Tech our first cohort of students will be completing CS2 in Fall 2008, and we are rolling out this study to many other schools that are adopting the personal robot approach to CS1.

8. CONCLUSION

Our approach to CS1 education is contextually driven by a low cost student owned personal robot that acts as a peripheral, executing programs running on the student's computer. The robot hardware design process was driven by curricular needs balanced against the need for affordability to promote individual ownership. Programs are executed using a dynamic language on the student's computer to ease debugging and reduce cognitive overhead. The Myro software API allows complex robot behavior to be specified with very concise programs. With Myro, not only do we intend to provide a low barrier to entry, but also a high ceiling for future growth (e.g. support for image processing and machine learning).

Over the course of four semesters we have iteratively developed and test taught a full CS1 curriculum at widely varying institutions. Results of initial pilot classes influenced a second generation of hardware and software driven by curricular needs. Developed curricular material includes a textbook, example assignments, and instructor material. Our initial results are promising, and we are currently undertaking a longitudinal study of the effects of our approach on student performance in CS2, retention and recruitment.

All of our material can be downloaded free of charge from our website, and we encourage you to use it as-is or modify it to meet your needs. We already know of adopters who are adopting our Fluke hardware to drive iRobot Create and humanoid robots, re-implementing Myro in C++ and translating our textbook to other languages. Others are developing upper level Robotics & Architecture courses based upon our open hardware and software platform. We invite others to adopt our curricular approach to teaching CS1 with personal robots and join our longitudinal study.

9. ACKNOWLEDGMENTS

The Institute for Personal Robots in Education gratefully acknowledges seed funding received from Microsoft Research and the continuing assistance from Stewart Tansley and Jared Jackson. We are also greatly appreciative of the time spent by instructors and faculty at other institutions to learn and implement our curriculum. Finally, the students in our classes offered numerous helpful comments and suggestions on ways to improve the hardware, software, and curriculum. We especially appreciate the permission given by many of our students who consented to participate in our research program. Research on human subjects has been approved by the Georgia Tech IRB under protocol #H07339.

10. REFERENCES

[1] T. Balch, J. Summet, D. Blank, D. Kumar, M. Guzdial, K. O'Hara, D. Walker, M. Sweat, C. Gupta, S. Tansley, J. Jackson, M. Gupta, M. Muhammad, S. Prashad, N. Eilbert, and A. Gavin. Designing personal robots for education: Hardware, software, and curriculum. *Pervasive Computing, IEEE*, 7(2):5–9, April-June 2008.

[2] D. Blank, D. Kumar, J. Marshall, and L. Meeden. Advanced robotics projects for undergraduate students. In *AAAI Spring Symposium, Robots and Robot Venues: Resources for AI Education*, 2007.

[3] D. Blank, L. Meeden, and D. Kumar. Python robotics: an environment for exploring robotics beyond legos. In *SIGCSE '03*, pages 317–321, New York, NY, USA, 2003. ACM.

[4] J. Challinger. Efficient use of robots in the undergraduate curriculum. *SIGCSE Bull.*, 37:436–440, 2005.

[5] S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. In *SIGCSE '03:*, pages 191–195, New York, NY, USA, 2003. ACM.

[6] B. Fagin and L. Merkle. Measuring the effectiveness of robots in teaching computer science. In *SIGCSE '03*, pages 307–311, New York, NY, USA, 2003. ACM.

[7] M. Guzdial. *Introduction to Computing and Programming in Python, A Multimedia Approach.* Prentice Hall, January 2006.

[8] F. Klassner. A case study of lego mindstorms' suitability for artificial intelligence and robotics courses at the college level. In *SIGCSE '02*, pages 8–12, New York, NY, USA, 2002. ACM.

[9] A. N. Kumar. Three years of using robots in an artificial intelligence course: lessons learned. *J. Educ. Resour. Comput.*, 4(3):2, 2004.

[10] D. Kumar, editor. *Learning Computing with Robots.* IPRE Publication, 2008.

[11] R. P. Loui. In praise of scripting: Real programming pragmatism. *Computer*, 41(7):22–26, July 2008.

[12] F. Martin. Real robots don't drive straight. In *AAAI Spring Symposium, Robots and Robot Venues: Resources for AI Education*, 2007.

[13] S. Papert. *Mindstorms: Children, computers, and powerful ideas.* Basic Books, January 1981.

[14] R. E. Pattis. *Karel the Robot (2nd Edition).* John Wiley and Sons, 1995.

[15] I. Tomek. Josef, programming for everybody. In *SIGCSE '82*, pages 188–192, New York, NY, USA, 1982. ACM.

[16] J. B. Weinberg, W. W. White, C. Karacal, G. Engel, and A.-P. Hu. Multidisciplinary teamwork in a robotics course. In *SIGCSE '05*, pages 446–450, New York, NY, USA, 2005. ACM.