USING ROBOTICS TO TEACH THE SCIENTIFIC METHOD

Gabriel J. Ferrer
Department of Mathematics and Computer Science
Hendrix College
Conway, AR 72032
(501) 450-3879
http://ozark.hendrix.edu/~ferrer/
ferrer@hendrix.edu

ABSTRACT

We present a series of lab exercises that teach non-science majors effective use of the scientific method. We have employed a robotics theme for these exercises, as robotics problems provide a compelling framework for the use of the scientific method in problem-solving. We identify four types of scientific experiments that are useful in this regard, and demonstrate how these types of experiments support the types of problem-solving essential to completing robotics tasks. Our approach is influenced by previous results in robotics education, computational use of the scientific method, and active learning techniques. Our exercises are available as a lab manual [12].

INTRODUCTION

Hendrix College has a campus-wide requirement (NS-L) that every student must take a science course with a lab. A student completing an NS-L course is expected to be able to apply scientific and mathematical principles in the context of the scientific method to systematically and critically assess the validity of observations related to the natural world.

We have created a no-prerequisites robotics course that satisfies this requirement; we have taught it 2-4 times per year since 2003. It combines introductory material from computing and physics. In this paper, we describe how our lab activities enable students to achieve our learning goals. These activities have been bundled into a lab manual [12].

We believe that for students (especially non-science majors) to find the scientific method interesting requires it to be situated within the context of a larger goal or purpose, a context we supply with robotics. The utility of robotic automation, in combination with the philosophical issues robots raise, makes the study of robotics compelling to students from a wide variety of backgrounds. The scientific method, then, is contextualized as a tool that enables students to solve problems that arise when building and programming robots.

The course uses the Lego Mindstorms NXT platform. Students program the robots using the pbLua programming language. We have augmented the pbLua language with a custom library to simplify certain programming tasks.

This paper is organized as follows. We begin with an overview of our learning goals. In this context, we describe the different types of problem-solving scenarios and scientific experiments that students undertake in their lab exercises, along with an overview of how those types are represented in the lab activities. We next give several examples of lab exercises that exemplify the different types of problem-solving scenarios and scientific experiments described previously. We then describe examples of free-form projects that students have developed after

completing the lab exercises. After a discussion of student reaction and a review of related work, we present our conclusions.

COURSE GOALS AND STRUCTURE

Our primary learning goal for the course is for students to learn to use the scientific method as part of the problem-solving process for building and programming robots. From this general goal, we formulated the following specific learning goals for our course:

1. Apply the physics of translational motion, rotational motion, forces, and gear ratios to create and test mathematical models of aspects of robot behavior.
2. Translate an English-language description of robot behavior into a corresponding computer program that employs basic programming constructs (including variables, loops, conditionals, blocks, and subroutines) to select robot motor settings based on inputs from light, sonar, rotation, and touch sensors.
3. Formulate and test hypotheses about expected robot behavior, given a robot and a program running on that robot.
4. Learn how to identify and fix problems on a robot that is not behaving as desired.

The course consists of ten lab activities. Each lab requires about four hours of in-class work; students complete a lab report outside of class time. One lab is completed each week. The final four weeks of the course consist of a free-form design project.

The lab activities consist of several different types of problem-solving scenarios:

1. Program Understanding: Students are given some code, and must figure out its meaning through a combination of reasoning and experimentation. This supports Learning Goal 2.
2. Equipment Understanding: Given some equipment (e.g. sensors or motors), students must determine how to use the equipment to enable the robot to achieve its goals. This scenario supports Learning Goals 1 and 2.
3. Program Modification: Students modify a given program in order to attempt to alter the robot's behavior in a desired manner. They test the program to see if the modifications succeed; they continue modifying the program until its behavior is acceptable. This scenario supports Learning Goal 2.
4. Program Creation: Students are given an English-language description of a desired robot behavior, and must translate it into a functioning computer program. This supports Learning Goal 2.
5. Robot Modification: Students are given a partially complete robot, and must modify it to give it a capability. This scenario supports Learning Goal 1.
6. Robot Construction: Students must construct a robot from scratch with certain desired capabilities. This indirectly supports all of our learning goals.
7. Robot Optimization: Students start with a functioning robot and control program, and must discover either physical or program modifications to improve its performance. This scenario supports Learning Goal 1.

To enable students to solve these problems, we teach them to conduct different kinds of scientific experiments:

1. Program Analysis: Students start with a section of code, either provided for them or of their own creation. Before running the code, they write down a hypothesis of its behavior. If a goal for

the code is available, students are also required to determine whether this goal will be achieved by the code. They then run the code to test whether the hypothesis is correct. This addresses problem types 1, 3, and 4, and supports Learning Goals 2, 3, and 4.

2. Program Comparison: Students compare the performance of programs representing different approaches to solving a problem. They use both quantitative and qualitative performance analysis to construct an argument for the relative merits of each approach. This addresses problem type 7 and supports Learning Goals 2 and 3.

3. Physical Analysis: Students create a mathematical model of robot behavior. They run the program, followed by performing measurements to test the mathematical model. This addresses problem types 3, 4, 5, and 7, and supports Learning Goals 1 and 3.

4. Equipment Analysis: Students are given a previously unused sensor and a program that displays the sensor's current reading. They then place the robot in various physical situations to determine the meaning of the readings, addressing problem types 2, 3, and 4, and supporting Learning Goals 1, 2, and 3.

LAB ACTIVITIES

Having discussed the types of experiments that are employed, in this section we will give some specific instances to illustrate their characteristics. In our first example, students are given the following program (after a brief explanation of the repeat-until loop):

```
forward(A)
repeat until isTouched(1)
stop(A)
```

They must then predict the program's behavior, type in the program, and check their prediction. They then must suggest modifications that would enable the robot to drive forward, stopping when the bump sensor is hit. They then test their hypothesis by seeing if their modified program successfully does so.

In the next exercise, most students fail to correctly predict the outcome. In the last exercise, the motors were going forward, and the robot stopped after 1000 rotation counts.

```
backward(A)
backward(B)
repeat
  write(rotationCounts(A))
until rotationCounts(A) > 1000
stop(A)
stop(B)
```

The students typically pause in puzzlement as the robot fails to stop, not immediately realizing that the rotation counts are descending rather than ascending. This is one of many disequilibration exercises [2] we have incorporated into the course.

In our final example exercise, students are given the following block of code. They are then instructed to point the light sensor at a variety of objects, recording the readings as they go. Rather than simply telling students that the values range from 0 to 100, and represent the percentage of light saturation, we require them to conduct experiments to investigate the practical meaning of the sensor values in a concrete environment.

```
repeat
  write("Level: " .. activeLightReading(3))
until isPressed(ESCAPE)
```

OPEN-ENDED PROJECTS

For the final stage of the course, students design and implement their own open-ended projects. It has been our experience that our lab exercises have prepared students very well for this task. Some of the most interesting projects students have devised over the years include:

- **A door-opener**: The robot used light sensors to receive a "request" for a door to be opened. It had a lever mechanism that was geared strongly enough to pull the door handle while its wheel base dragged the door open.
- **A robotic airship**: The students carefully determined the amount of lift necessary to transport the robot aloft, and were able to acquire a suitable helium balloon for this purpose. They hand-carved their own propellers for propulsion.
- **A keyboardist**: The robot would drive between locations in front of an electronic keyboard with marks in place to enable the light sensor to identify key notes. It played a melody from a popular movie soundtrack.
- **A page-turner**: This robot would turn pages in a cardboard-style children's book upon a sensory cue from the reader.

STUDENT RESPONSE

Course evaluations from the most recent offering confirm that students are receptive to these lab activities. The 15 enrolled students included one Mathematics major, four Economics majors, two International Relations majors, one Religious Studies major, one English major, two Art majors, one History major, and three students who had not yet declared a major. On the course evaluation, every student agreed with the statement that "The assigned text was an effective instructional tool." Here are some representative free-response comments from students:

- "I know your goal for this class is to make science fun and accessible to people like myself and I just wanted to let you know that you have succeeded."
- "Learned new things every lab."
- "The lab book is pretty clear but doesn't always help on really hard labs that introduce new commands."

The last comment is not unexpected; it is part of the course design that the students have to experiment a bit to see how certain commands work. We see it as the role of the instructor to help the student during the lab who begins to flounder during this process.

RELATED WORK

The course that most directly inspired our own is the Robotic Design Studio from Wellesley College [9]. The most important ideas we have adopted from their course are the final design project, a "no prerequisites" approach, and an immersive lab experience with no formal lectures. Our course differs from theirs in that the Wellesley course goal is "Engineering

for Everyone", exposing liberal arts students to engineering concepts. In our case, the main motivation is to provide hands-on experience with the scientific method. The lab activities in the two courses consequently have a very different flavor.

The lab manual we have written for our course shares strong affinities with that created and described by Briggs [3]. His course employs an active learning approach based on a lab manual that interleaves reading material with exercises. He describes positive learning outcomes that we can confirm on an anecdotal basis. A similar problem we faced is that of consistent attendance; as with his course, we have a very strict attendance policy.

Many computing courses employing robotics are introductory programming courses (e.g., [5] [8] [10] [11]). The primary learning goal of these courses is to develop general programming skills and stimulate student interest in majoring in computer science. As our focus is on developing the use of the scientific method, general programming skills are deemphasized. In our exercises, looping has very little application beyond repeating the execution of a reactive control program. Consequently, we neither intend to nor expect our students to develop a strong intuition about the more general abstractions that loops provide.

Several papers have studied the impact of robotics on student motivation (e.g. [5] [11] [6]) and technical competence [4]. Our experience anecdotally confirms the positive results on student motivation from the literature. The negative result for technical competency from Fagin and Merkle [4] does not have much impact on our course design, for developing programming skills at a level comparable to CS1 is not one of our learning goals.

Several papers have investigated the explicit incorporation of the scientific method into computing education (e.g. [2] [7] [1]), the most pertinent of which describes Braught's disequilibration exercises [2]. We employ this technique across many program analysis exercises, especially in the early part of the course. The resulting atmosphere of uncertainty has yielded for us the benefits he describes, on an anecdotal basis.

CONCLUSION

Robotics problems provide an ideal framework for teaching students the scientific method in a hands-on active learning environment. In this experience report, we have shown how a series of lab exercises involving escalating problem complexity suffices to immerse students in the scientific method. We identified four major types of experiments that students perform repeatedly in different forms to support their problem-solving efforts.

The course has proven to be well-received, both in terms of course evaluations and in terms of enrollment; over 10% of graduating students at our institution (enrollment approximately 1350) have taken the course over its ten-year history. Due both to its popularity and its academic success, we highly recommend this approach to faculty at institutions with a comparable curricular situation.

REFERENCES

[1] G. Braught, C. S. Miller, and D. Reed. Core empirical concepts and skills for computer science. In SIGCSE '04, pages 245–249, 2004.

[2] G. Braught and D. Reed. Disequilibration for teaching the scientific method in computer science. In SIGCSE '02, pages 106–110, 2002.

[3] T. Briggs. Techniques for active learning in CS courses. Journal of Computing Sciences in Colleges, 21(2):156–165, Dec. 2005.
[4] B. S. Fagin and L. Merkle. Quantitative analysis of the effects of robots on introductory computer science education. Journal on Educational Resources in Computing, 2, Dec. 2002.
[5] M. M. McGill. Learning to program with personal robots: Influences on student motivation. ACM Transactions on Computing Education, 12(1):4:1–4:32, Mar. 2012.
[6] W. I. McWhorter and B. C. O'Connor. Do Lego Mindstorms motivate students in CS1? In SIGCSE '09, pages 438–442, 2009.
[7] D. Reed. The use of ill-defined problems for developing problem-solving and empirical skills in CS1. Journal of Computing Sciences in Colleges, 18(1):121–133, Oct. 2002.
[8] J. Summet, D. Kumar, K. O'Hara, D. Walker, L. Ni, D. Blank, and T. Balch. Personalizing CS1 with robots. In SIGCSE '09, pages 433–437, 2009.
[9] F. Turbak and R. Berg. Robotic design studio: Exploring the big ideas of engineering in a liberal arts environment. Journal of Sci. Ed. and Tech., 11(3):237–253, Sept. 2003.
[10] S. van Delden and W. Zhong. Effective integration of autonomous robots into an introductory computer science course: a case study. Journal of Computing Sciences in Colleges, 23(4):10–19, Apr. 2008.
[11] D. Xu, D. Blank, and D. Kumar. Games, robots, and robot games: complementary contexts for introductory computing education. In GDCSE '08, pages 66–70, 2008.
[12] G. Ferrer. *Introduction to Robotics using Lego Mindstorms NXT and pbLua*. Lulu.com. 2012.