A Robotics Introduction to Computer Science

Debra T. Burhans

Canisius College, Computer Science Department
2001 Main Street WTC 207, Buffalo, NY 14208
burhansd@canisius.edu

Abstract

This paper describes a new undergraduate course that serves two purposes. First, it satisfies a general education requirement in mathematical sciences, and second, it serves as a first course for computer science majors. The course has no prerequisites: the student population is drawn primarily from college freshmen. This paper focuses on two aspects of the course. The curriculum, which blends topics from basic computing, artificial intelligence, and robotics, is discussed. The use of an in-house-developed robot simulator in conjunction with actual robots to help realize some of the course goals is also described.

Introduction

This paper describes a new undergraduate course that serves two purposes. First, it satisfies a general education requirement in mathematical sciences, and second, it serves as a first course for computer science (CS) majors. The course has no prerequisites: the student population is drawn primarily from college freshmen. This paper focuses on two aspects of the course. The curriculum, which blends topics from basic computing, artificial intelligence, and robotics, is discussed. The use of an in-house-developed robot simulator in conjunction with actual robots to help realize some of the course goals is also described.

The robots employed in the course are Lego Mindstorms that use the RCX platform. The simulator provides a basic model of the brick that can be used to test student programs both in and outside of the lab setting.

Background

Robotics courses are proliferating due to a number of factors. Availability of low-cost robot platforms, faculty enthusiasm, and the need to level the playing field and to help attract new computer science majors are just some of these. Some robotics courses are for computer science majors and serve to introduce students to AI concepts in the context of robotics. Others are pure robotics courses. A growing number of schools are using robots in their introductory courses. The majority of these efforts can be divided into two categories. (1) Increased use of robots in CS I and II to help students grasp fundamental concepts as well as to boost enthusiasm and retention. (2) Robotics courses for non-computer science majors that serve to convey ideas from computing and AI to a general audience through robotics.

CSCI 108 at Williams College (http://www.cs.williams.edu/~andrea/cs108/) is an example of the latter type of course. It provides an introduction to the field of AI through a variety of readings and activities including a robotics laboratory. Robots are built using the Handyboard platform and are programmed with Interactive C. There are no course prerequisites. The course is not part of a CS major, rather is offered to students outside the major to fulfill a general education requirement.

In the Fall 2006 semester George Washington University is offering a robot-based section of its 8-week freshman course CS 001: Computer Science Orientation (http://www.seas.gwu.edu/%7Ebhagiweb/cs1/). This course is designed to introduce CS majors to the field and includes a lab-based project. The project involves teams of students working on maze navigation with Lego Mindstorms (RCX brick). This is a short introductory experience for majors.

Lego Mindstorms have been used with the Ada programming language to teach introductory computing to both CS majors and non-majors at the US Air Force Academy (Fagin). The stated goals of this course, along with an assessment of the ease of doing so with robotics, are:

> …to introduce students to basic computing ideas, including sequential control flow, selection, iteration, input/output, arrays, graphics, procedures, and file processing. Some of these concepts easily lend themselves to robotics, others are a better suited for a more conventional paradigm. Sequential control flow is the easiest.

Many colleges offer an upper-level AI course that includes robotics or a robotics course for CS majors.

All of our computer science majors begin with a breadth-first computing course (CS 0) prior to their first course in Java programming (CS 1). This introductory semester is

intended to help equip students with some of the skills that are needed to succeed in CS 1. Our students overall are weak in quantitative skills and tend to be math-phobic. A standard programming language is not introduced in this course because students are unprepared for the precision and rigor it entails. It should also be noted that this course carries 3 semester credit hours as opposed to CS 1 and most other CS courses, which carry 4 semester credits. Students have 4 contact hours a week in our CS 0 course including a 50-minute lab meeting, but expectations are considerably lower than they would be for a 4 credit offering, particularly for work done outside of the class meetings. This new robotics course represents a specialization of our "vanilla" CS 0 course that must operate within the constraints of our current CS 0. Thus, it is also a 3 credit course with a 50 minute lab, it must address the needs of students who are often poorly equipped for quantitative reasoning (and often who are taking a CS course to escape from mathematics), and must also prepare students who are going on in the major for their first programming course (CS 1). The goal is to satisfy all of these needs and in doing so give students an interesting and exciting first exposure to college-level computer science that may encourage them to continue in the discipline.

Curriculum

In order to develop not only this new robotics CS 0 but other possible specialized version of the course we identified key topics currently covered in CS 0. These include: problem solving, algorithms, history, ethical and social issues, databases, computer system components, networking, WWW, software/OS, binary representation, basic logic, limitations of computing, AI, and simulation. These topics correspond approximately to those presented in most CS 0 texts, some topics underlie the rest (problem solving and algorithms) while others receive a cursory overview (AI).

From this list we identified core topics that all of our specialized CS 0 courses would cover: problem solving, algorithms, history (of whatever topic is the course focus), binary representation, basic logic, and social and ethical issues (again, focused on the topic area). In order to reinforce the breadth and unifying CS concepts in the robotics course a CS 0 book was required. There was no suitable robotics book as we do not use a traditional programming language for the robotics; rather, we are using our own algorithmic language.

The robotics course has covered many of the core topics listed above as well as some other topics, endeavoring to cast most everything in terms of robotics. This section will highlight some of these topics as the course is still in progress.

Material for the history of robotics unit (a combination of history of CS and history of AI with more material from the recent past on robotics) was assembled by the instructor based on a number of Internet and book resources. It was difficult to assimilate the vast amount of material and some traditional history of CS material was excluded. Some historical events such as the construction of mechanical animals and people could be seen to foreshadow the development of androids.

Hands on from the start

We elected to have each student build his or her own robot, without any expectation as to whether this would be good or bad. Once students started programming the robots they worked in teams, developing programs in groups of 2 or 3 yet downloading the programs onto each individual robot. As more complex tasks were posed, including preparing for a competition, teams selected one robot to use for task completion yet continued (for fun) to alter the other robots belonging to the team. While building robots was frustrating there was clearly value in getting one completed and getting it to do things. Students will be surveyed at the end of the semester to assess the value of this approach to robot building.

Problem Solving and Algorithms

Problem solving and algorithms are the foundation of the course and are developed primarily in the laboratory through hands on exercises. The language used is Robolang, which is associated with Robotran, a program that translates Robolang programs into Lejos (http://www-cs.canisius.edu/~rmmeyer/ROBOTRAN/home.html). All of the software for the course as well as the simulator has been developed in our department. Robolang is a simple algorithmic language that is easy for students to understand. It provides a rich set of primitives for problem solving and is designed specifically for use with the Lego RCX.

The following are a couple representative Robotran programs. The first is a simple loop to move the robot forward, albeit slowly (three seconds forward, two seconds back).

```
program prog4
     while true
        go forward 3 seconds
        stop
        go backward 2 seconds
     end
```

The second program tests the value of a touch sensor connected to S1. If the sensor is triggered the robot backs up and turns right a random amount. The random function in Robolang by default returns a value in the range 0-99. The units for turns may be specified, here degrees are used.

```
program prog14
     S1 is a touch sensor
     go forward
     while true
        if S1 == 1 then
          stop
          go backward 1 second
          var angle = random
          turn sharp right angle degrees
          go forward
        end
     end
```

Methodology: Students are first presented with a task for the robot to complete. They work in teams to come up with a strategy for completing the task. The strategy is then formalized using Robolang. Robolang programs are translated into Lejos, compiled, and downloaded onto the robot and run.

The simulator provides an alternative to trying a program on a real robot: programs can be run on the simulated robot in a manner analogous to using the real robot. The simulator is an essential component for a number of reasons. Our resources are limited. There are not enough robots for every student or team to use, we can't monitor them to the point that we could let students take them home, and there are not enough open hours in the robot lab to allow for extensive student experimentation. Students generally do not have their own robot kits.

It should also be noted that, while it is fun to work with all of the real hardware, there are times when it is simpler and more fun to work with a keyboard and screen.

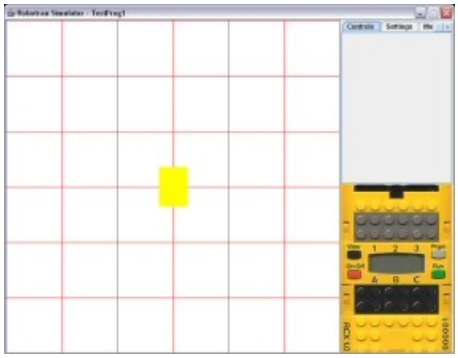

Figure 1. Screen shot of simulator, note that all buttons shown on the brick (lower right hand corner) are functional and are used to control robot activity in the simulator.

The first tasks we assign involve having the robot draw letters. The first robots the students build (called "penbots") can raise and lower a pen using a third motor (the other two are used for movement). They draw on large pads of paper. Other tasks (without the pen attachment) include line following, staying within a ring, and maze navigation. We have found that letter drawing is an ideal initial problem to work on, and the results are immediate and visual.

Letter drawing using the simulator is also very visual and gratifying for students to see. The simulated robot performs "pen up" and "pen down" commands. With the pen down the simulated robot essentially traces the path it follows. While we have used this to work on drawing letters, it could be used to simply keep a trace of where the robot has been during the run of a program.

Binary Representation and Logic

The idea of different representations of information and translation among representations is familiar to students through the use of Robotran.

The GUI for Robotran displays both the Robolang algorithm and the corresponding Lejos source code. The value in this approach is that students understand that what actually is "understood" by the computer, in this case the RCX, is not necessarily written in the language they use to think and problem solve. Students must compile and download their programs, reinforcing the idea of a hierarchy of formats and translation among them. This provides a good basis for introducing binary representation to students.

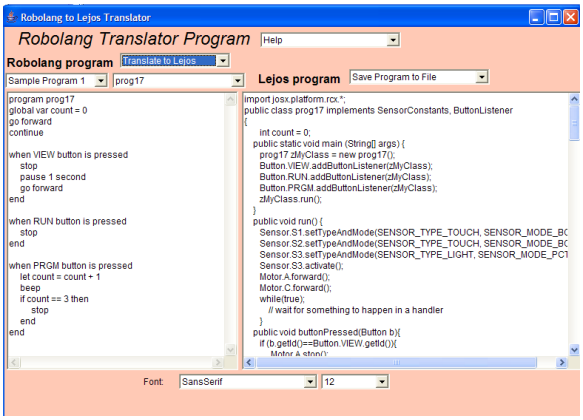

Figure 2. Screen shot of Robotran GUI showing Robolang on the left and Lejos code on the right.

Note however that teaching students to translate between bases is, at present, not done using robotics.

Once students understand binary representation they are introduced to logic gates as the fundamental way of manipulating the binary information in a computer. When they see how logic functions can be used to implement bit-wise addition they understand the power of logic for computation. Again, this is not taught in the context of robotics. We are currently using Logisim for circuit building (http://ozark.hendrix.edu/~burch/logisim/).

AI and Robotics

Concepts of AI are introduced in a number of ways, including literature and current news. Students take turns presenting "Robots in the News" where they have to find something in the news that focuses on robotics. Thus far these stories have ranged from nursing home robots in Japan to the Scooba. We have read Asimov's *I Robot* and Lem's *Cyberiad*. There is an excellent episode of the cartoon series "Futurama" (Obsoletely Fabulous) that we showed in the class. It touches on issues including human aggression vs. "killer robots", the lack of souls in robots, and Luddism. Through these fun and extremely accessible activities students come to see the importance of AI and robotics in their own lives. The robots they are building and programming, while extremely simple compared to those they encounter in fiction and the news, help them to better understand the potential difficulties in creating robots as well as highlighting some of the social and ethical issues.

Logistics

The course was scheduled to meet Tuesday/Thursday, making lecture periods 1.25 hrs in length. The laboratory was scheduled for Thursday immediately following the lecture for an additional hour. This allowed us the flexibility to spend varying amounts of time up to 2.25 hours in the lab on Thursday which has proved invaluable. Limiting the lab time to 50 minutes would not be sufficient.

Simulator: The ability to use the simulator affects the logistics of the course: we can do many things that would not be possible if we had to do them all on actual robots in a laboratory setting. The simulator enables students to work on and test different strategies in preparation for an upcoming competition, which involves maze navigation, a sumo contest, and a drag race, without having to try everything first on a real robot.

There are limitations to the simulator. For example, modifications to the robot that make it heavier, can not currently be represented in the simulator, but navigation and basic sensor input can. The simulator allows users to insert objects such as walls and blocks that the robot will detect with touch sensors.

Results

The course is still in progress, and no doubt there will be modifications to its next offering this coming spring. The students are evenly divided between CS majors and non-majors. Interestingly, students who took programming courses in high school have proven to be the slowest at completing tasks in the lab. It is difficult for them to work with a simple algorithmic language even though they clearly have not grasped many programming concepts. They tend to be poor at following directions, which is an essential component for success in the course.

The creation of teams has helped students work together and to get to know one another better. It is the most cohesive group of students I have had in many years of teaching. They did not start this way, but have evolved during the first two-thirds of the semester to their current state. The shared experiences they have had (often frustrating), particularly during the first couple weeks of building a robot from scratch, has led them to help one another with everything from finding robot parts to understanding robot programs.

Interestingly, some of the intended CS majors are not as certain that they want to continue in the major. It does not appear that we have picked up any new majors, either. The course may give students a better feel for what computer science involves, and many students don't find it compelling. In particular, getting hardware and software to work together can be challenging and frustrating, and having the patience to persevere is an important trail for CS majors. Many students discover this in CS 1 or 2, which often have high dropout rates.

Future Work

There are a number of aspects of the course that need tuning or further development. For example, if Robotran could be expanded so that it could show translated forms of the Lejos programs, including assembly language and byte code, it could provide a seamless and compelling way for students to understand the importance of binary representation. An end-of-semester survey is planned to collect data on how the course was received by students. In addition, those students going on to CS 1 will be compared to their peers to see how their experience in this course affects their understanding of new concepts in that course.

References

Barry Fagin, Using Ada-based robotics to teach computer science, ACM SIGCSE Bulletin, v.32 n.3, p.148-151, Sept. 2000.