Introducing Computer Programming with Lego Robotics

Michael Greenspan

Karen Rudie

Stan Simmons

Dept. Electrical & Computer Engineering
Queen's University, Kingston, Ontario, Canada
michael.greenspan@queensu.ca

Abstract

The first year undergraduate course Introduction to Computer Programming for Engineers (APSC 142) was considered to be challenging for both students and instructors alike. The engineering curriculum at Queen's University is structured so that the students take a common first year, and select their particular engineering discipline near the end of that year. Not only did the course therefore have large enrolment, with 700 students, but it also comprised many students who did not feel that computer programming was a required skill set for their ultimate intended disciplines (e.g., Chemical, Industrial, Mining, or Mechanical Engineering).

In order to re-engage the students, the course was significantly redesigned to center around the use of LegoNXT robots. A series of labs was developed where the students would program directly on the robotic platforms, which offered immediate and tangible feedback for their efforts. A software simulator of the robot was also custom-developed and distributed to the students, so that they could further exercise their programming skills. This also allowed students to work independently and to test their programs and ideas in a simulated robot environment, thus alleviating the need for 700 students to have access to

the more limited hardware during nonworking hours. The role of lectures was reduced, in favour of instruction through an interactive design studio. This paper will describe the rationale for the changes that were put in place, some of the issues and challenges that resulted, and the positive effect that they have had on improving student engagement.

1 Introduction

APSC 142 is an introductory computer programming course offered to first year engineering students at Queen's University. The instruction of this course posed a number of challenges, one being the common first year at Queen's, whereby all incoming engineering students take exactly the same courses, irrespective of their eventual intended discipline. This not only led to a large class enrolment of 700 students, but also meant that there was a wide variation in the students' interest levels. Students destined to pursue degrees in Chemical or Mining Engineering, for example, did not necessarily view computer programming as essential an element of their education as those heading for Electrical or Computer Engineering. The students also had a large variation in background preparation. About 10% had been exposed to programming in high school, and some had already developed sig-

nificant skills, such as multiple languages, OO and GUI development, etc. Other students had no exposure to computer programming whatsoever, and were starting at square one.

Historically, the course had been instructed through the School of Computing within the Faculty of Arts and Science. For budgetary reasons, in 2007 it was decided to patriate the instruction into the Dept. of Electrical and Computer Engineering within the Faculty of Engineering and Applied Science. The switch introduced a natural opportunity to adjust and/or redesign the course, the main issues being the following:

- **Student Engagement:** Many students seemed to view this course as a hurdle to get through, rather than as an opportunity to learn a skill that would be important to them in their future education and eventual career. This was particularly ironic in that the instructors of the course tended to view programming as not only important to academic development, but also as an enjoyable intellectual pastime. Was there a way to increase student engagement, so that they could learn to appreciate the discipline of programming?
- **Language:** Language of instruction is a perennial issue in an introductory computer programming course. Previously, the course had been taught in Java, a decision that was made in the late 1990s. At that time, Java was viewed as being cutting-edge, and many institutions were turning to Java to attract students. By 2007, however, it was not clear that Java was the most desirable introductory language: The OO capabilities of Java are supported by a rich syntax, whereas most if not all of the concepts of OO are ignored in an introductory course. Was there a language that was more suitable

Figure 1: Lego NXT Robot Configured for APSC 142

than Java for introductory computer programming instruction?

- **Resources:** APSC 142 was a large and resource intensive course, requiring five instructors and two lab coordinators. The instruction was delivered over two lecture hours and two lab hours per student per week. Was it possible to streamline the instruction, to reduce the number of required resources, especially instructors?

2 Instructing with Lego Robotics

The main design decision was to make use of Lego NXT robotic kits as an instructional device. Despite its origins from a children's toy manufacturer, the Lego NXT platform is a sophisticated device with significant capabilities. Illustrated in Fig. 1, each NXT unit comprises: a 32-bit embedded ARM proces-

Figure 2: RobotC IDE

sor with Bluetooth wireless capability; a 64-by-100 pixel LCD display; a speaker for sound creation; ports to simultaneously connect up to four sensors; and independent left and right wheel motor drives with built-in wheel rotation encoders (and/or internal speed-correcting feedback). The firmware has built-in arrays for sensor values and timing control, as well as callable functions for sound and screen-display, motor control, and the standard set of mathematical functions. The sensor set includes two contact sensors, an ultrasonic range sensor, a sound sensor, a downward-looking reflective light sensor, a 3-axis accelerometer, and a multi-directional infrared sensor.

One option for programming the Lego NXT robots is to use RobotC. The C programming language is a standard procedural programming language, which is grammatically straightforward and is viewed by many as the lingua franca for engineering application programming. RobotC is a subset of C developed by Carnegie Mellon University [1], and includes all of the major syntactic elements, as well as a library of functions specific to the control of the NXT robots. RobotC is bundled with a rich Integrated Development Environment (IDE) that includes wireless downloads via Bluetooth and a language-sensitive symbolic debugger, shown in Fig. 2.

In addition to the decision to make use of

Figure 3: The QNXBotSim Simulator

Lego robotics as a focus, the course was also reorganized to make use of a Teaching Studio, which is a relatively new facility within the Integrated Learning Center at Queen's. The studio comprises a set of concentric computer workstations, each of which seats two students. Each workstation can serve as a standard networked PC, and is also connected to a central master PC, which can display information on a separate slave display. The studio provided an alternative to traditional computer labs and take-home programming assignments, which were not only resource intensive to grade, but which provided an opportunity for copying that was practically impossible to catch given the large number of students in the class.

Both the studio and the lab also made use of a custom simulator called QNXBotSim, illustrated in Fig. 3. This simulator provided full support of RobotC, with an IDE and execution environment, including single-stepping with a variable-display pane to facilitate source-code debugging. The simulator also included a graphical birds-eye rendering of the robot itself, with animated motion as the program was executed. The simulator was distributed freely so that the students could practise coding outside of the studio and lab, and it was also a useful tool for instructor demonstrations during lectures. Any code that was developed on the simulator was fully compatible with the

hardware robots, and could be ported to and executed on the robot during lab sessions.

3 Course Content and Logistics

The course covered the standard introductory content, i.e. selection, loops, arrays, functions etc., over a 12 week term. Each week, each student was scheduled into an hour-long lecture on Monday, followed by a 1.5 hour studio session later in the week that built upon the lecture material. There was also a two hour weekly lab period, where the students programmed in groups of two on the Lego NXT robot platforms. In the final four weeks of the lab, students implemented an open-ended design project that required them to develop often non-trivial programs that made use of various sensors, sounds, display, and motor actuators to accomplish a task.

The use of a robotic platform for learning programming has a number of advantages. Rather than learning programming for “programming’s sake”, students were motivated to learn programming to get the robot to do what they want it to. Feedback was immediate — if there was a program bug it became obvious through incorrect robot behaviour. Although students in the course spent less time on the more advanced capabilities of the C language (pointers, structure, file operations), they spent more time on the behaviour of a hardware/software “system”, which could be seen as a more productive focus for engineering students.

The robot’s capabilities also assisted students in learning the programming fundamentals. The various sensors created an immediate need for program selection operations like if-else and switch-case. The first exposure to arrays occurred early on through the built-in 1-D sensor and motor arrays, and understanding of 2-D arrays was facilitated by use of the robot’s 2-D LCD display screen. The sequential nature of program execution was nicely demon-

strated with the use of 1-D arrays of musical tones that could be played through the built-in robot sound functions to create recognizable ringtones. Functions must be employed to access built-in capabilities (like screen display and sound), and the need for modularization through functions became evident when students tackled the more complex programs required in their projects (and the utility of function modules was reinforced through an earlier obstacle-avoidance lab).

The final four-week project was open-ended. Students were expected to design their own projects, in groups of two or three. They were to decide which sensors to use and what robot behaviour they wished to create (high-level goals, plus lower-level motions, screen display, and sounds). This afforded students a chance to use creativity, and exercise their design skills. They could also tailor their project’s complexity to their aptitude, interest, and time. The flexible project scoring scheme rewarded program complexity in five categories, and the use of the more complex sensors (infrared and accelerometer I2C sensors) was rewarded with more points. In the past two years, the students have come up with a surprising number of completely original and very interesting projects, including things as complex as multi-robot projects involving coordination using inter-robot Bluetooth text communications. Finally, a significant benefit of the student-defined projects (rather than a common “assigned” project) was the removal of any incentive for copying, which was a significant issue in previous incarnations of the course.

The number of faculty instructors required to deliver the course was reduced. The class of 700 was divided into 21 sections of ~ 34 students each. Each of the three instructors was responsible for delivering:

- A one hour lecture per week of seven sections (~ 243 students).

- A 1.5 hour studio per week of one or two sections (~ 50 to 80 students).
- Periodic presence at four weekly two-hour lab sessions.

In addition to the faculty instructors, there were also 11 graduate Teaching Assistants (TAs) assigned to the course as follows:

- Three covered the remaining six weekly studios (two each), and assisted with exam marking.
- Six presided over the 12 weekly two hour lab sessions (two each).
- One was designated as the *meta-TA*, and performed TA scheduling and administrative tasks.

Finally, there were 24 undergraduate TAs, drawn from the pool of students who had taken the course and excelled within the previous two years. Each undergraduate TA helped in one of the weekly two-hour lab sessions, with which they were very familiar from having taken the course, and also marked the short weekly quizzes that were taken at the end of each studio session.

Although there are no required assignment hand-ins for credit (which avoided a very large resource need for TA marking), weekly studio session quizzes kept students on pace with course material. In addition, weekly exercises and solutions were posted on the course web page. Quizzes were based on the previous week's studio and lab material, so that students had an incentive to try the practice exercises as preparation for the following week's quiz.

4 Outcome

The primary and most tangible result from the course redesign was a perceived increase in student engagement. The students were excited to use the robots, and were consequently motivated to learn the course material.

This was especially the case during labs, where the hardware robots were used, and especially for the term project, which was an open-ended design. Even those students destined for engineering programs that do not traditionally require strong programming skills viewed the course favourably. In the course evaluations, many students commented on the positive lab experience, and expressed a desire for more access to the robots. There were requests the following year from a subset of the same group of students to borrow the robots as a basis for a second year Mechanical Engineering project.

The increased student engagement also translated into increased instructor satisfaction. Previously, the course was renowned as being a difficult teaching assignment. One incoming instructor was warned that his teaching evaluation scores were likely to suffer as a result of having been assigned this difficult course. To the contrary, all instructors scored very well on student evaluations, above departmental average, which is unusual for a large lower-year course. While grading exams for ~ 700 students will always be a chore, the student body is no longer perceived as overly apathetic, the course is no longer viewed as unusually difficult to instruct, and indeed is now a course of preference for some.

There is a diversity of opinion as to the degree to which the redesigned course was less resource intensive. Prior to the redesign, the course was delivered by five faculty instructors, each responsible for two weekly lecture hours. There were four graduate and 24 undergraduate TAs, who conducted the 12 weekly two-hour lab sections. There was also a single course coordinator who administered over the labs and the course. Whereas the number of faculty instructors assigned to the course was reduced from five to three, there were seven more graduate TAs required. Faculty instructors tend to be more scarce than graduate TAs, so it is likely that this tradeoff represents a re-

duction. It was felt that the larger lecture sections and addition of studios, not to mention the requirement to attend some labs, may have increased the individual instructor contact time and work load somewhat over the previous organization.

The final, and in many ways most important consideration, was the quality of the course content delivery in the new design, i.e. did the students learn the material better? Despite the relatively high grades in the course and the perceived increase in student engagement, it is difficult to know whether learning improved. Even though the course average improved seven percentage points in the first year's offering, grades in a subsequent second year course were actually somewhat lower. To muddy the waters further, in the next course in the series, the average had improved over the previous year. Needless to say, it is difficult to draw conclusions in the short term for performance over a small set of courses, as any grade fluctuations may be due to other factors. Indeed, it might not be possible to gauge the quality of the course content delivery until some time has passed, and students are challenged to build on material in later year courses. For the short term, the somewhat anecdotal perception of student engagement and satisfaction, as for example indicated on course exit survey scores, may be the best indicator.

5 Summary and Conclusions

The redesign of APSC142 made use of Lego NXT Robotics as a focus to motivate first year engineering students to learn the basics of computer programming. The resulting redesigned course is believed to have increased student interest and engagement in the material. The students' perception of the course seemed to change from it being an unpleasant chore, to it being an interesting and fun activity. The instructors' impression of the

course also improved: what was previously a difficult assignment, is now for some a preferred course. Instructor resources were nearly halved, from five to three, albeit at an increase in graduate TA resources from four to 11. It is inconclusive whether or not there was any change, positive or negative, to student learning, and it is not clear how this might be fairly measured. Overall, the redesigned course provided a good balance of theory and practice, which is an often sought-after goal of engineering education.

References

[1] <http://www.robotc.net>