Development of a system for teaching CS1 in C/C++ with Lego NXT robots

A. Delman, A. Ishak, L. Goetz, M. Kunin, Y. Langsam and T. Raphan

langsam@sci.brooklyn.cuny.edu

Department of Computer and Information Science, City University of New York (Brooklyn College), Brooklyn, New York, USA

ABSTRACT

This paper describes the development of a system for teaching C/C++ using a Lego™ NXT in a CSI college course on introductory programming. The programming of the NXT robot has been implemented using a C/C++ cross-compiler, which generates code that runs on an Open Source firmware platform, nxtOSEK. The system has built in commands and objects that run the motors and receive information from the sensors that are on the robots. The cross-compiler has been embedded in an Open Source Integrated Development Environment (IDE) Code::Blocks. The programming environment for the NXT has evolved from a previous development using the Lego™ RCX and has the advantage that it utilizes a Bluetooth interface, while the RCX uses a tower based infrared communication device. The NXT is more reliable and can be programmed to pair a specific interface with a particular robot, so that there is no cross-talk when different robots are utilized in a classroom setting. The IDE and robotic software executes on a virtual machine running under the freely available software, Sun™ VirtualBox. This allows for a uniform programming platform for Windows, MacOS, and Unix/Linux. The use of robots in CS1 affords science and engineering students the opportunity to learn sensory-motor based control, to work with an IDE early in their careers, and to gain experience with development and debugging tools that can be utilized throughout the students' academic and professional careers.

Keywords

College, Introductory Programming, Robotics, Science, CS 1

1. INTRODUCTION

We have recently presented results on the development of Lego™ RCX platform for teaching C/C++ to students entering in an Introductory Course in Programming (CS1). The purpose of our development was to address the needs and interests of students who plan on entering science-based careers and who would be better served by a course whose emphasis is on scientific and engineering applications of programming [1].

Our approach to teaching the science-based CS1 is two pronged, utilizing text based console applications on a personal computer to teach the students standard programming techniques, and using robotics as a means for teaching programming constructs and sensory-motor computer interactions. This approach uses the science students' innate interest and the cache of robotics as motivating mechanisms to help the students commit to demanding programming practices. Once the students have mastered a few simple programming skills, they can immediately advance to writing programs to control a robot.

Our basic development of the robot component of the Introductory CS1 course utilized the Lego™ RCX MindStorms robot kits. In this paper, we describe the development of a system for teaching C/C++ using Lego™ NXT Robots, the NeXT generation of robot from Lego, which is an extension of our development using RCX robots [28].

2. BACKGROUND AND OVERVIEW

Computer science and programming practice has experienced a considerable conceptual revolution in the past two decades and the educational component has sought to keep pace with this development. One important development has been the use of robotics as both a motivating and a "hands on" approach toward computer science education [4, 5] and to teach students algorithmic thinking [4, 6]. These approaches have their origins in the turtle graphics of Logo [7], whose aim was to teach algorithmic thinking by moving a turtle around and not concentrating on the specifics of the language. Other mini-languages that were developed were Karel the Robot [7-9], which was designed as a "gentle" introduction to Pascal for students taking their introductory course and Josef the Robot [10]. The development of science-based computer science instruction gave rise to such mini-languages as Wayfarer, Turingal and Tortoise [11-13].

Recent developments in educational robotics have utilized Lego-based robots embedded in a wide range of intelligent agent-based environments [14-16] and which utilized other languages [17]. These approaches have shown that working with the MindStorms Lego™ robot enhances the usual CS lab experience that entails sitting in front of a computer screen and typing. The laboratory experience now becomes an active learning session, enhances interaction with other students [18], and gives them the opportunity to test their code and solutions to a programming problem on a moving robot. More importantly, Programming the MindStorms Lego™ robot gives students immediate visual feedback, allowing visual debugging, and reinforces the idea that the ultimate goal of a program is to accomplish a task. For engineering and science students, it accomplishes the important task of introducing them to sensory-motor based computer control early in their careers.

Despite these advances in the use of robotics in computer science education, students are still required to learn a traditional programming language such as C/C++ for use in advanced courses, and become successful programmers. Mini-languages, regardless of their ability to simply convey concepts, do not prepare students to be successful programmers. Therefore, there is a need to develop an environment where students can learn to develop and debug code that they write and also incorporate a robot environment. This is especially important for science/engineering students whose future will depend on not only being able to program in a language such as C/C++, but also have an appreciation for how to control processes in such an environment.

Unfortunately, no simple robotic platform is available to students for home use or within a classical CIS laboratory environment. This is especially true for newer operating systems such as Vista and the latest Linux-based distributions. Furthermore, many existing educational robotic systems require that students learn new programming languages that have been developed for that specific robot [2]. This is an inhibiting factor in learning programming techniques using a basic procedural language such as C++. Another important challenge in incorporating robotics into a first course on programming is that most students will not have access to the robots outside the classroom. There is therefore a need to have a uniform development environment that each student can work with both at school and at home.

Recently, we have developed a uniform system for teaching robotics programming by combining open source environments that include Sun™ VirtualBox and consolidating the tools that have been developed for robot control within a single integrated development environment (Code::Blocks), which was useful in helping students learn to program and debug their code even at the beginning CS1 level [1, 19]. This environment has both advantages and disadvantages. The advantages are that it is simple to use for a first course in programming and open source environments have been developed for its programming [28]. It also was important in introducing students to the constructs in programming a robot, how to compile programs for it, and how to communicate with the motors and sensors that interact with the environment. The main disadvantages are that it is obsolete, having been replaced by the Lego™ NXT, and difficulties in computer-robot communication within a classroom environment, where multiple robots and computers are being utilized. The infra-red towers used for communication are also a disadvantage in that they do not allow for communication over far distances and are subject to noise and miscommunication.

3. SYSTEM COMPONENTS

To address the many disadvantages in using the Lego™ RCX, we developed an environment for programming the next generation robot, the Lego™ NXT. In our development of the system, we have used the same open source software as for the RCX system. However, the incorporation of an appropriate cross-compiler was more challenging than the incorporation of the RCX cross-compiler and is described here. We have also developed scripts, which have been stored on a DVD and are handed out to students, that self installs each component of the system. The various system components and their integration are described in this section.

3.1 The Lego ™ NXT Brick

The Lego™ NXT is a programmable robotics kit released by Lego in late July 2006. It is the NeXT (NXT) generation Lego MindStorms kit (Figure 1), and replaces the first generation kit called the Robotics Invention System (RIS or RCX) (Figure 2). The NXT kit contains an NXT brick, commonly called the "brick," which receives a program via Bluetooth or a standard USB connection. It has various connectors for motor control and sensors. In addition, NXT allows for output on an LCD screen and for sound tones. Programs developed on the PC are downloaded to the brick via the Bluetooth connection.

The brick can be configured so that it is a simple wheeled robot (Figure 3), but has the capability to be configured as a biped, which will not be considered in this paper. The motors and sensors can be independently controlled, so that students can be taught about decision as well as trajectory following. More importantly, they can be taught fundamental programming constructs in the context of a

Figure 1

Figure 2

Figure 3

fun and exciting environment. It also has an LED display so that messages can be sent to the robot and be displayed.

3.2 nxtOSEK, Code::Blocks, and ARM Cross-Compiler

The Lego™ NXT system contains built in firmware for programming the brick. Lego™ developed firmware for the brick to interact with its own Robolab software, a propriety non-procedural programming language and environment using LabView™ (National Instruments, Inc). Because of the desire to teach robotics using various languages, various types of firmware have been developed by the open source community and are available for the NXT brick. NxtOSEK, which is an open source platform for the Lego MindStorms NXT, is uploaded by using the Enhanced NXT standard firmware. This allows for other programming software (i.e. NXT-G, NXC/NBC) to be used without replacing the firmware. Creating a program for the brick in C++ is then no different from creating a console program. Both are created using the C++ language, but the brick runs the John Hansen Enhanced NXT firmware, which gives it the ability to run C++ compiled code.

The tools involved required that extensive configurations be made, since students work on various platforms (PC, Mac, and Linux). Each of these platforms requires students to take a different approach in configuring the tools. To simplify the configuration of the environment, we used a virtual machine (See Section 3.5), which provided an easily replicated environment for students to do their lab assignments and experimentation [28].

The cross compiler for nxtOSEK, which uses the ARM processor, was built from sources that were contained in GNUARM, an open source cross-compiler. The compiler was then easily set up in Code::Blocks. Thus, when students create projects within Code::Blocks, they simply select the project type to be a nxtOSEK project. The cross compiler is automatically selected for them, and the language choice is set to nxtOSEK. Compiling a program for the brick within Code::Blocks is therefore similar to the console

applications they learn to program. The cross-compiled linked C++ program that is to be executed on the brick is then downloaded to the brick.

3.3 Robot Library Configuration

In order to make it easy for students to program the robots, we incorporated the header files that nxtOSEK needed to control the robot into a single header file called robot.h. In addition, nxtOSEK does not provide a main function. It works on a series of tasks. We defined in `robot.h` the nxtOSEK task to become `main`.

By having the main function available, CS1 students are able to seamlessly move to the robot programming from the console applications they had been exposed to.

An example of code using nxtOSEK to display a message on the LCD screen of the brick is as follows:

```
#include "robot.h"
main() {
    Lcd lcd;

    lcd.clear();
    lcd.putf("s", "Hello Bluetooth");
    lcd.disp();

    while(1);
}
```

This program first includes the robot libraries and the re-definition of `task()` to `main()` in the header file `robot.h`. The program can then start with `main()` and the class `Lcd` and its object functions become available. The method `lcd.clear()`, clears the display, `lcd.putf()` sends the string to be displayed, i.e., in this instance "Hello Bluetooth." The designation "`s`", as the first parameter in the method indicates that a string is being sent. Since the string is sent to a buffer, a separate method needs to be called, `lcd.disp()`, which then displays the string on the LCD. The `while (1)` statement insures that the program continues to run indefinitely.

Another program that turns the wheels by programming the motors is given as follows:

```
#include "robot.h"
Motor motorA(PORT_A); // brake by default
Motor motorB(PORT_B); // brake by default
main()
{
      Clock clock;

      motorA.setPWM(100);
      motorB.setPWM(100);

      clock.wait(1000);

      motorA.setPWM(0);
      motorB.setPWM(0);

}
```

This program utilizes the classes `Motor` and `Clock` to define instances of the `Motor` class, `motorA` and `motorB`, and an instance of `Clock`, `clock`. The program then utilizes the method `setPWM()`, which sets the pulse-width-modulation parameter that does the low level control of the motor. The method `wait()` contained in the class `Clock` defines how long the motors are activated and then the motors are deactivated by a parameter of `setPWM(0)`. While these programs use object oriented concepts, they are simple enough so that CS1 students can understand them, even without having grounding in object oriented programming.

3.4 Sun™ VirtualBox

Sun™ VirtualBox [25] is a freely available virtual machine software package. This allows us to configure a machine with all the software necessary for the class. The same windows are obtained if the user is running the system on a computer whose host operating system is Windows XP, Mac OS X, Windows Vista or Windows 7. Since each student works on a common virtual machine (Ubuntu Linux client), troubleshooting for instructors is simplified. Instructors may still be faced with the problems that students encounter installing and accessing the virtual machine. However these problems are minimal compared to the problems related to having a student setup all the individual robotic software files.

3.5 System Configuration

Since the student population exposed to this system and environment is a CS1 group, we have developed a preconfigured system, which allows the student to install the entire system on a wide range of platforms, including Windows XP/Vista/7 and MacOS, automatically. The Code::Blocks IDE and the robotics based tools can also be installed directly on Unix/Linux.

We have created a DVD containing VirtualBox and the virtual machine file. We have called our package CPlusVEBot (C++ Virtual Environment Robot). A student uses CPlusVEBot to install VirtualBox on their computer using our system file. When the virtual machine begins running, the students have instant access to the IDE and robotic software. This is the same software that is installed on the computers in our laboratory. Whether students are running their programs on a Mac, PC or Linux, they will have the exact same interface and tools. A detailed student friendly manual has been written, which guides the student through the installation

4. CLASSROOM EXPERIENCE

Our initial use of the system was on an RCX system and was successful [28] for use with students in the first CS1 course, who are generally unsophisticated in the use of systems and have had little programming experience, if any. Despite this lack of experience all students (25), in our initial class, were able to install the system the first time without assistance. After two C++ programming assignments to introduce simple C++ constructs, students were given their first robot programming project (See Web Page [26]). The students were able to successfully utilize the IDE to develop and debug their programs and to use the emulator to verify the robot-based program execution. Our initial feedback from the students is that they were engaged in their project and were motivated to complete it. We expect that the NXT experience will be equally productive.

5. CONCLUSIONS AND FUTURE WORK

The primary advantage of evolving to the NXT robot is that it utilizes a Bluetooth interface, which allows for a more targeted communication between computer and robot. This makes the Lego™ NXT more reliable and can be programmed to pair a specific

interface with a particular robot. This reduces cross-talk and noise in communication when a group of students utilize different robots in a classroom setting. It also has better sensors and can be adapted to build bipedal robots for learning algorithms about locomotion in a more general setting. The use of the NXT robot also has the advantage that it is a newer platform and utilized more object oriented constructs, which introduces students to Objects and their utility earlier in their careers in a motivating manner, using robot programming. Students also learn about newer modes of communication between the computer and peripheral devices, such as Bluetooth. Thus, this robot platform has the potential for development of courses for a wider range of topics in multimedia applications. We anticipate that the system we have developed will have a strong positive influence on both science, technology, engineering, and mathematics (STEM) based instruction as well as on basic computer science instruction. It will consolidate the learning experience of STEM-based topics under the rubric of a programming environment. It will also teach programming techniques using tools that will motivate students to trace and debug their programs. This is known to enhance good programming practice and helps students learn better. It will also give them experience with a simple, freely available IDE (Code::Blocks) early in their careers, which will serve them well throughout their careers.

Another important benefit of our approach, which we have described previously [28] is that the system can be used with Windows (XP, Vista) and MacOS based systems. Moreover, despite the fact that no students in our initial group had any experience with Unix/Linux, they easily adapted to this environment and should help them in future courses. The system has the added benefits of exposing students, early in their careers, to a wide range of operating systems and gives students the opportunity to experience first-hand the value of virtualization, which is becoming an important topic in computer system development.

Finally, the use of open source and freely available programs and systems is also important in that students learn the importance of the open source community in the development of computer resources. This should reinforce their learning experience. Ongoing work includes the development of a simulator for the Lego™ NXT robot, which would allow students to develop programs at home and was a useful learning tool using the RCX environment.

6. ACKNOWLEDGMENTS

Supported by NSF-CCLI award and PSC-CUNY-09

7. REFERENCES

[1] Gurwitz, C. and T. Raphan, “CS1 for an Early College Program”; FECS, 13-17, 2008.

[2] Powers, K., et al., “Tools for Teaching Introductory Programming: What Works”; ACM SIGCSE Bulletin, 38(1), 560-561, 2006.

[3] Moskal, B., D. Lurie, and S. Cooper, “Evaluating the Effectiveness of A New Instructional Approach”; Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education. (ACM Press), 75-79, 2004.

[4] Beer, R.D., H.J. Chiel, and R.F. Drushel, “Using Autonomous Robotics to Teach Science and Engineering”; Communications of the ACM, 42(16), 85-92, 1999.

[5] Meeden, L., “Using Robots as an Introduction to Computer Science”; Proceedings of FLAIRS-9: The Ninth Florida Artificial Intelligence Research Symposium, 473-477, 1996.

[6] Powers, K., S. Ecott, and L.M. Hirshfeld, “Through the Looking Glass: Teaching CS0 with Alice”; Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education; (ACM Press), 213-217, 2007.

[7] Papert, S., “MindStorms, Children, Computers, and Powerful Ideas”. Basic Books, 1980.

[8] Pattis, R.E., “Karel the Robot: A Gentle Introduction To The Art of Programming”. John Wiley and Sons, 1981.

[9] Pattis, R.E., J. Roberts and M. Stehlik, “Karel - The Robot, A Gentle Introduction to the Art of Programming”, (Second Ed.). Wiley, 1995.

[10] Tomek, I., “An Introduction to Computer Programming”. Prentice Hall, Inc., Upper Saddle River, NJ: page 320, 1983.

[11] Brusilovsky, P., “Languages for Teaching the Principles of Programming”, (in Russian); Informatika i Obrasovanije (Informatics and Education). 1990.

[12] Brusilovsky, P., “The Intelligent Tutor, Environment and Manual for Introductory programming”; Educational Technology and Training International, 29(1), 26-34, 1992.

[13] Brusilovsky, P., E. Calabrese, J. Hvorecky, A. Kouchnirenko, and P. Miller, “Mini-Languages: A Way To Learn Programming Principles”; Education and Information Technologies, 2(1), 65-83, 1997.

[14] Erwin, B., M. Cyr, and C.B. Rogers, “Lego Engineer and Robolab: Teaching Engineering with Labview from Kindergarten to Graduate School”; International Journal of Engineering Education, 16(3), 181-192, 2000.

[15] Fagin, B.S., L.D. Merkle, and T. Eggers, “Teaching Computer Science With Robotics Using Ada/MindStorms 2.0”; Proceedings of the 2001 Annual ACM SIGAda International Conference on Ada, 73-78, 2001.

[16] Azhar, M.Q., R. Goldman, and E. Sklar, “An Agent Oriented-Oriented Behavior-Based Interface Framework For Educational Robotics”; Agent-based Systems for Human Learning (ABSHL) Workshop at Autonomous Agents and Multiagent Systems (AAMAS-2006), 2006.

[17] Blank, D.S., M. L. Meeden, and D. Kumar, “Python Robotics: An Environment for Exploring Robotics Beyond LEGOs”; Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, (ACM Press), 317-321, 2003.

[18] Imberman, S.P., “Teaching Neural Networks Using Lego Handy Board Robots”; Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, (ACM Press), 312-316, 2003.

[19] Imberman, S.P. and R. Klibaner, “A Robotics Lab for CS1”; Journal of Computing Sciences in Colleges, 21(2), 131-137, 2005.

[20] Hoenick, J., Available from: http://hoenicke.ath.cx/rcx/brickemu.html.

[21] Sklar, E., et al., “Educational Robotics in Brooklyn”; AAAI-06 Mobile Robot Workshop, 2006.

[22] Flowers, T.R. and K.A. Gosset, “Teaching Problem Solving, Computing, and Information Technology with Robots”; Journal of Computing Sciences in Colleges, 17(6), 45-55, 2002.

[23] Hundersmarck, C., C. Mancinelli, and M. Martelli, “Viva la BrickOS”; Journal of Computing in Small Colleges, 19(5), 305-307, 2004.

[24] Hickman, G.D., “An Overview of Virtual Machine (VM) Technology and Its Implementation in I.T. Student Labs at Utah

Valley State College"; Journal of Computing in Small Colleges, 23(6), 203-212, 2008.

[25] Sun Microsystems, Available from: http://www.virtualbox.org/.

[26] Delman, A., L. Goetz, Y. Langsam, and T. Raphan, Available from: http://eilat.sci.brooklyn.cuny.edu/cis1_5/Programming%20the%20LEGO.pdf.

[27] Goetz, L., Y. Langsam, and T. Raphan, Available from: http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/codeblocks-instructions.pdf

[28] A. Delman, L. Goetz, Y. Langsam, and T. Raphan, "Development of a System For Teaching C/C++ Using Robots in the first CS1 Course Using Open Source Software", Proceedings of the WORLDCOMP'09 - The 2009 World Congress in Computer Science, Computer Engineering, and Applied Computing