Providing Robotic Experiences Through Object-Based Programming (PREOP)

James Davis
The University of Alabama
Tuscaloosa, AL
jdavis@cs.ua.edu

Briana Lowe Wellman
The University of Alabama
Tuscaloosa, AL
blowe@cs.ua.edu

Monica Anderson
The University of Alabama
Tuscaloosa, AL
anderson@cs.ua.edu

Michael Raines
The University of Alabama
Tuscaloosa, AL
mraines@cs.ua.edu

ABSTRACT
PREOP (Providing Robotic Experiences Through Object-Based Programming) is an integrated approach to using robots within the introductory Computer Science sequence. Rather than use the “program-by-wire” approach of existing tools, PREOP leverages an object-based, syntax-free environment. PREOP provides a physical, relevant, challenging problem domain that appeals to students. This paper presents the PREOP implementation, its technical challenges and the promising preliminary results.

Categories and Subject Descriptors
K.3.2 [Computing Milieux]: Computers and Education—Computer and Information Science Education
; D.2.8 [Software Engineering]: Programming Environments—Graphical environments, Integrated environments
; I.2.9 [Computing Methodologies]: Artificial Intelligence—Robotics

General Terms
Design, Languages, Human Factors

Keywords
Computer Science Education, Graphical Development Environments, Syntax-free development environments

1. INTRODUCTION
This paper seeks to introduce and describe a modification to the interactive programming environment Alice. By supplementing the virtual world of Alice with robotics, the modification expands upon the utility of the software through the addition of a tactile functionality. Considering the highly lauded success of Alice in addressing the problematic retention rates in the field of Computer Science, it seems reasonable to predict that such an expansion may increase its efficacy.

The purpose of Alice is to instill confidence and motivate introductory Computer Science students, which, in turn, increases interest and therefore, retention. Syntax can be a significant hurdle for entering students with no prior programming experience. Alice’s interface requires no knowledge of syntax. This results in increased confidence in programming ability, and through a maintained program structure, Alice is able to simultaneously emphasize core concepts. Alice motivates students by enticing them with the proposition of creating interactive 3D worlds. In short, Alice keeps Computer Science majors interested by giving them the ability to easily program with a compelling output.

Our addition to Alice will be referred to as “Providing Robotic Experiences Through Object-Based Programming” or “PREOP” (available at cs.ua.edu/PREOP). PREOP adds a physical component to the output of Alice, leaving the rest of the software intact. Students can program a robot to perform actions and respond to virtual or physical stimuli with the proven Alice interface. This modification serves the same function as Alice’s creation of interactive worlds. Students unmotivated by a purely virtual output could potentially be compelled by the physical element of PREOP.

As a preliminary study, PREOP was implemented in the introductory Computer Science curriculum at The University of Alabama (UA) in the fall semester of 2008. In this experiment, students in a class that utilized PREOP were compared to students in classes that followed the pre-existing curriculum, the results of which will be discussed in this paper.

The following explains the organization of this paper. The related work is presented in Section 2. In Section 3, the approach including software features, hardware, and software development is discussed. Section 4 follows with a discussion of classroom deployment and preliminary results. The conclusion and future directions are in Section 5.

2. RELATED WORK
Prior research supports the use of robots and robotic applications to help increase student motivation, confidence and competence. There are a number of reports that de-
scribe the usefulness of robotics in teaching science, math and computing at all education levels [12, 10, 3]. However, the application of robotics in Computer Science first-year sequences has lacked key elements required to successfully teach programming.

Syntax-free graphical environments have proven useful for teaching programming constructs [4]. However, current robotics-based approaches do not mirror any of the well-known programming paradigms well enough to facilitate knowledge transfer to programming. Most graphical robotics-based programming environments focus on “wire-based” graphical interfaces that connect data sources with data consumers. Although studies have shown that robotics in introductory sequences requires a simulation component to facilitate initial code generation and testing [5, 6], only the Microsoft Robotics Studio provides a simulator.

The Institute for Personal Education Robotics (IPER) [1] is a consortium of university and industry partners that focuses on an integrated approach that combines software, hardware, and instructional materials. The center of the approach is the scribbler robot, an inexpensive robot that is purchased and owned by the students. Robot control programs are created using an API from traditional languages such as Python and C.

Berea College takes an alternative funnel-based approach that uses robotics as an elective course designed to pique interest but leaves the first-year sequence intact [11]. The Berea College Math and Computer Science department has experimented with different technologies to attract undecided students to Computer Science. A series of special topic “CS0” classes were offered that included web development, Alice and robotics using RoboLab. Each technology appealed to different groups based on CS0 enrollment. Although students taking the Alice CS0 course had the highest contribution of students to the CS1-programming with C++ course (57.5%), robotics had the second highest retention rate (50%). The robotics course had a higher level of minority students persisting in CS than did the Alice course, which may suggest that robotics is more appealing to minorities. This is consistent with previously documented phenomena that suggests that minority students prefer to work on problems with societal benefit [7, 9]. However, women preferred the Alice CS0 course to the robotics CS0 course. The CS0 web course appealed strongly to both women and minorities.

3. APPROACH

This approach outlines the steps needed to create a typical PREOP project to familiarize the reader with the new software’s features. Then, we discuss the technical details regarding these features and the process of development. Finally, we present the method by which the preliminary study was conducted at The University of Alabama.

3.1 Hardware

A personal computer (Windows/Unix/MAC) with Bluetooth support and an iRobot Create (Figure 1) with a Bluetooth Adapter Module (BAM) are required to fully operate PREOP. PCs without Bluetooth hardware can be fitted with a Bluetooth device. iRobot, makers of the autonomous vacuum Roomba, produces the Create, a vacuum-less Roomba, for hobbyists and educators. The BAM connects to a serial port on the Create and forwards received Bluetooth transmissions to the host for wireless control of the robot.

Figure 1: iRobot Create robot with a BAM installed.

3.2 Demonstration of Software Features

The purpose of this section is to familiarize the reader with the use and features of PREOP to ensure that subsequent, technical explanations will be more meaningful.

After starting PREOP, users are presented with an interface nearly identical to that of Alice. In order to control a nearby Create, outfitted with a BAM, users must add the CREATE OBJECT. As the CREATE OBJECT follows the same rules as any other object in Alice, the new object can be added and manipulated with a basic knowledge of the software. If a user selects the added virtual robot, the OBJECT DETAILS panel will display the functionality of the Create, decomposed into the now-familiar categories of properties, methods and functions. Each of the categories contains details that are specific to the CREATE OBJECT.

To establish a Bluetooth connection between the software and the BAM, the user modifies the virtual robot’s CONNECTED TO property. This presents a dialog where the user can type a unique four-digit code to connect to a particular BAM. Once connected, the user might drag MOVE, TURN and PLAY NOTE methods or functions that return sensor data, into the code editor.

Figure 3 shows a program that moves the robot forward until it hits a wall, whereupon it will play a note while reversing at an arc and then, proceed forward again. The figure includes a DO TOGETHER block. When commands are located within one of these designators they are non-blocking, that is, they are permitted to run simultaneously with other commands. The example includes a TURN and MOVE command executing simultaneously, which results in the tracing of a rounded path.

Figure 3 also includes a PLAY NOTE method that accepts a NOTE variable, a category of variable not included in the original version of Alice. Users define notes by selecting values for pitch and duration in a prompt (Figure 4). Also, the Alice type LIST can be used to create an array of notes accepted by the PLAY SONG method, which causes the robot to sound a succession of notes.

Users can simulate the output of their programs by selecting the PLAY button, which presents a new window with a
3.3 Technical Challenges

The development of PREOP posed many challenges, the most important of which are covered in the following sections.

3.3.1 Designing the Create object

Careful consideration was taken to ensure PREOP maintained the proven structure of Alice. The most important decisions involving congruency with Alice were made with regard to the seamless integration of the Create’s functions into Alice’s world.

Objects in Alice are composed of properties, methods and functions (Figure 2). Properties hold data that can be freely altered in much the same way that public attributes may be. Methods are the equivalent of their namesake in conventional class vocabulary, that is, they tell the object to do something. Lastly, functions correspond to private, read-only attributes of a class.

The only actions the iRobot Create can perform, movement and note playing, are designated as methods. The color and power status of the LEDs are denoted as properties, because the lights’ characteristics can be written to and read from. The bump detectors and wheel-drop sensors are categorized as functions, because the data from these sensors can only be read.

3.3.2 Connection using Bluetooth

Alice’s code consists almost entirely of Java. Sun Microsystems’s Java has no official support for Bluetooth communications; however, a specification (JSR-82) [8] has been written outlining a recommended interface for a Java Bluetooth library. Several third-party companies have created libraries to fit this specification. BlueCove [2] was chosen for our purposes, because it is a free, open-source Bluetooth library that functions on both Windows and Linux. PREOP uses BlueCove to send and receive data from the BAM.

In an early version of PREOP, the connection procedure searched for and listed all of the BAMs detected by the PC. The user would then select a device and connect to it. In a room filled with BAMs, populating the list proved time-consuming, so a shorter method was devised. Now, users input the last four digits of a BAM’s unique identification code (Figure 5). The system searches for devices until the correct one is found and automatically connects to the device. The new method has accelerated and simplified the connection process, although some versatility has been lost.

3.3.3 Moving the Create

Translating the movement of a virtual object in Alice to
the physical movement of a robot poses unique problems. Tantamount to these is the disparity between the way that Alice animates objects and the way that the Create follows instructions.

The virtual Create and the physical Create function on different planes. For the sake of animation, Alice makes minute, incremental changes to the virtual robot’s orientation. These rapid changes, however, would be impossible for the real robot to perform, because once set, the velocities of the robot’s wheels are maintained until otherwise commanded.

The first step taken to convert Alice methods into commands for the real Create was the development of a function that would yield the desired trajectory in a physical setting. This function takes the forward and turning velocities in the 3D environment and converts them into the individual wheel velocities of the real Create. If the real Create is given input that is not physically possible, the function caps the wheels’ rotational velocities to approximate the motion.

Next, the Alice code that controls the execution of methods was modified to add and subtract velocity from forward and turning velocities already in use. Velocity is added when the Create’s method begins, and is subtracted when the method is complete.

Finally, after any change in velocity, the wheel velocities of the Create are updated using the function discussed earlier. A buffer combines commands received in rapid succession to avoid jumpy movements that could damage the hardware.

This solution allows for the successful synthesis of any combination of method calls. For example, consider the methods MOV E FORWARD and TURN LEFT performing simultaneously. The velocities would be updated when Alice ran the methods, triggering a command to be sent to the Create. The robot would travel in an arc until the methods terminated and the velocities were subtracted. The change would then send a command to stop the robot. Figures 6 and 7 show an example program and illustration of how commands are sent to the Create.

The following describes the approach used to test the effect of integrating PREOP into an introductory Computer Science course. UA requires a problem-solving lab (CS116) to be taken with or before the first Computer Science course (CS114). Of the five labs offered in the fall semester of 2008, one was chosen as the test group. The class consisted of forty students that worked in pairs, each with a PC and an iRobot Create. Classes began with a fifteen-minute lecture on a programming concept, after which, students completed a PREOP assignment illustrating the concept. Topics discussed included: objects, methods, variables, if-statements, loops and other fundamental programming concepts (Table I). Students were required to submit a lab report demonstrating their comprehension of the fundamental concepts. At the end of the semester, the entire population of lab students was given surveys about their performance in their first Computer Science course.

4. PREOP IN THE CLASSROOM

4.1 Results

The survey was administered to students during the 12th week of class, meaning that students still in the class could no longer drop out of the course. The survey was administered in the CS116 lab and included students who had already dropped the CS114 course (generally considered to be more challenging than the lab). Of the seven CS majors in the Alice with Robotics CS116 lab section, there was only one withdrawal from the CS114 course. The remaining sections of the traditional CS116 lab reported eight withdrawals out of the twenty CS majors, which correlates with the one-third drop rate usually experienced in CS114. Of the eight students that dropped CS114, five reported that they would re-attempt CS114 the following semester. The student who dropped CS114 from the experimental section of CS116 planned to retake the course the following semester. Because this data is self-reported, additional data must be obtained from the Registrar to accurately characterize the effects of the new CS116 programming lab on both student drop rates and persistence. Also, larger numbers need to be studied for statistical significance. However, this data, coupled with much higher recommendation rates for the experimental course over the traditional course, suggests that the new lab may be an asset to achieving greater skill acquisition and persistence in the major.

These results motivated Berea College to incorporate PREOP into their CS curriculum. Berea students implemented the same assignment (programming a polygon) in both PREOP and RoboLab. The thirty-seven students were then asked to contrast the two environments. Students reported appreciating the availability of the simulation environment.
Approximately 40% of students used a programming concept to describe PREOP such as, “coding,” “sequence,” “algorithm,” “function” and “object-oriented.” Most students referred to RoboLab as “wiring,” which is not a programming concept, and further commented that, “RoboLab is for an advanced group of individuals.” Comments made about PREOP included that: “[PREOP] required fewer commands,” “we know what [the icons] mean,” and “[it was] easier to input algorithms.”

5. CONCLUSION AND FUTURE WORK

In this paper, we detailed the process used to integrate robotics into an existing educational programming environment and discussed the resulting product’s first round of testing in an introductory Computer Science curriculum. The approach was successful, but PREOP lacked some important features, and further study will be required to thoroughly establish our software’s efficacy.

A limitation of PREOP involves the simulator’s lack of support for the Create’s virtual sensor responsiveness. Users cannot test collisions in Alice’s virtual environment; thus, users can only use the bumper sensors in a physical environment. Future additions to the software should include a more robust simulation environment, including collision detection, to solve this problem.

The results of our study indicated that the use of the robotics software was an improvement over the existing curriculum, but the study lacked the scope needed for statistical reliability. Therefore, a larger study, including a comparison to the original Alice, should be conducted in order to evaluate the effectiveness of our method.

6. ACKNOWLEDGMENTS

This work is supported by NSF grant DUE-0736789. We would like to acknowledge Dr. Jan Pearce for her assistance.

7. REFERENCES


