Robot, Meet C++: The Process Of Writing an Introductory C++ Lab Set That Utilizes a Robot

Genevieve Lynn Walker
Mathematics Department, Austin College,
Sherman, TX 75090 Email: glwalker@austincollege.edu

***Abstract*— This paper describes the process of creating a set of labs that utilize a robot to convey computing concepts in an introductory, C++ based Computer Science course. There are existing programs which use robots in this manner, but the University of Tennessee has special needs which require a curriculum tailored to their introductory course. The process my partner and I carried out involved five steps: research and understanding, organization, brainstorming, solving, and evaluation. We developed eight complete labs, two partially complete, and one lab idea, and determined that these are sufficient for the course because they meet three predefined criteria.**

I. Introduction

Currently, Computer Science is a bust business in the major declaration market of American colleges. According to the Computing Research Association, after seven years of declines, the number of new CS majors in fall 2007 was half of what it was in fall 2000 (15,958 versus 7,915) [7]. Luckily, many educators and concerned computing citizens have been working to solve this problem. One group of such people is the Institute for Personal Robots in Education (IPRE). This institute is based out of the Georgia Institute of Technology and Bryn Mawr College, and aims to "show that by empowering every student with their own personal robot, purchased with the class textbook, [IPRE] can improve retention in and attraction of students to computer science" [1]. Their current curriculum involves teaching computing concepts using a small, inexpensive robot, the Parallax Scribbler, and a Python interface.

A. Motivation

IPRE intends to make their curriculum easily available, and easily implementable; the Institute maintains a Wiki Page, where the textbook is avaliable for free [3]. They chose a cheap robot so that the program would not be cost prohibitive, and most information about the program is easily accessible. Essentially, they hope that, if a professor is interested in the idea of teaching computing with robots, it would be very easy to drop the IPRE program right into an existing classroom relatively quickly. Unfortunately, many college computing departments have restrictions and requirements concerning the language their introductory computing students must learn. In the case of the University of Tennessee at Knoxville (UTK), this means that all introductory computing classes must be structured so that the students learn C++. If the IPRE program is to be successful in reaching as many students as possible, curriculum for the robots must be developed in other computing languages.

The following project concerns the development of lab curriculum for the Scribbler robots in a C++ environment. A C++ robot API was developed by Richard Edwards, a graduate student at UTK, alongside (or, rather, in front of) the set of labs. UTK will use the robots, the new API, and the new set of labs in one of its introductory CS 102 lab sections in the fall of 2008. My partner, Allison Thompson, and I were assigned the task of creating labs that would prepare students for future Computer Science courses in a manner equivalent to or better than the existing labs.

II. Related Works

The main resources in the development of the labs were the notes from UTK's CS 102 class, and the IPRE Myro textbook. At the beginning of the summer, I envisioned the new labs to be a direct melding of the two resources. Other important resources have been the websites of the Georgia Tech and Bryn Mawr Myro-based introductory and exploratory courses and the lab write-ups for the CS 102 course last year and in previous years . As I researched each of these sources, I came to realize that the UTK CS 102 course covers many more computing topics (several of them C++ specific) than the Myro textbook, and that the Myro textbook includes topics that are not necessary or that are impossible to include in a C++ course. Also, the CS 102 course is not functionally equivalent to the introductory courses at Bryn Mawr and Georgia Tech. The UTK course must prepare its students for the C++ based course load that follows the introductory course while some of the courses at the other schools are designed for non-Computer Science majors.

Due to this difference, a new task became reconciling the differences between the less demanding Myro curricula and the relatively fast-paced C++ curricula. This meant adapting ideas from the Myro textbook to the CS 102 lab write-ups. I relied on the IPRE annual report and the wiki IPRE Philosophy page to make sure that the adaptations I made were still in alignment with the ideas and motivations of the Institute. Related works are avaliable in [8] , [5] , [4], [2].

III. Approach

There were five stages in the creation of the labs. In order, they are Research and Understanding, Organization, Brainstorming, Solving, and, finally, Evaluation.

A. Research and Understanding

During the first stage, I read the CS 102 notes and Myro Textbook, outlined both sources, and carried out the labs in the Myro textbook. After making my way through all of the instructional material, I made sure that I understood all of the concepts presented and why they were included. This meant analyzing the order of concepts in the Myro book and having discussions with the professor and TA of the CS 102 course.

I also read through the CS 102 lab instructions from last year. While we used these write-ups to get an idea of their structure and the amount of material covered in each session, we did not use them to model the order or depth of concepts in the labs. As a result of scheduling, funding for TAs in the labs next semester, and the fact that the labs are out-dated, it turns out that the aforementioned write-ups will be re-written or adjusted to better suit the future lab environment, regardless of whether or not the robots are used in the lab. Thus, we decided not to mirror the existing labs in any formal way; though, we did use a few approaches from beginning labs which were less likely to be drastically changed.

In this stage, we began using the developing API. It was important to become comfortable with this interface and understand its potential. I wrote simple programs to experiment with the developed functions and test their utility.

B. Organization

The next step was to organize all of the concepts that are taught in each course and rank them. We had to decide which concepts must be taught, which would be interesting for the students to encounter, and which could not be addressed with the new C++ API.

Once we had a good idea of the importance of the concepts, we began brainstorming ideas for labs. Our goal—to create labs that would prepare a solid computing and C++ knowledge base for upcoming courses—will not be fully evaluated until the end of the 2008 Fall semester. Therefore, we developed the following criteria for the labs to ensure that the goal would be met eventually. We feel that an appropriate set of labs will (in order of priority):

1) Include the concepts covered in the CS 102 course lecture in similar order and depth.
2) Utilize the robot in a non-artificial way that is engaging, exciting, and not frustrating.
3) Include topics from the Myro textbook.

C. Brainstorming

To secure the first criterion, we took the list of necessary concepts from the lecture notes, grouped them based on the order in which they were taught and how many concepts could be covered in a week. Then, we looked at each group and created a lab that would incorporate as many concepts from that group as possible. Most of the inspiration for the topics of labs came from the C++ API that was developed concurrently with our project. For example, we would notice that vectors were used in the retrieval of the robot's line following sensor data, and would then develop a lab around a line following capability and have it correspond to the lecture in which vectors were taught.

We came across several obstacles in developing new ideas for the labs. One such obstacle was how to introduce the idea of building and using a class in C++. Obviously, the students use the robot Scribbler object in all the programs that use the robot. We needed, however, an object that they could create—one that would also interact with the robot in a non-artificial way. The last thing we wanted was to create a lab where the students would build an object that either truly had nothing to do with the robot or had no real purpose. We knew that the students would see through this type of lab and exhibit the "I don't see the point of this" syndrome.

Our solution to the object problem was to create a Navigator class which would contain methods for navigation and exploration that would utilize the robot's sensors. We were already planning to write labs that incorporated line following, blob-following, manual-driving, etc., so having the students combine all of these methods into one object seemed (after much brainstorming) the only logical solution to our problem.

D. Solving

Once all of the labs had been outlined, we went about solving the labs. We would use the new C++ API to write a program that would solve the lab problem we had outlined. During this process, we would come across problems that we had not anticipated. Sometimes we would find that we could not solve the problem as we had planned, thus, the lab would cover different concepts than expected, which may then affect another lab. In this way, it was necessary to shuffle the order and content of the labs several times. After we solved a lab, we then used that experience to write the lab instructions. In this way we could include hints and processes that would lead the students in the direction to solve the problem in a way that would teach the proper concepts.

E. Evaluation

In the Evaluation stage, my partner and I as well as the lecture professor, other Computer Science professors, and the TA for the upcoming course, took a look at all of the labs in series and made sure that they all followed a logical pattern and covered the concepts necessary for the course.

During evaluation, we realized that we need essentially two more lab write-ups. Luckily, this only meant expanding two one-week labs into two-week labs. This included adding a blob-following application to the light-following lab, and increasing the complexity of the previous panorama lab by including smart-stitching instead of simple end-to-end picture placement. The ideas of the panorama lab are based off of a lab written by Keith O'Hara and carried out in the Scribbler's Python environment [6]. Most of the brainstorming for the last additions had already been done, but had been omitted because we felt that the labs were getting too long. Therefore, we simply moved on to the Solving stage and added the lab instructions to the final collection.

TABLE I

SUMMARY OF THE DEVELOPED LABS

Lab	Description	Concepts
1. Introduction	Students are introduced to Linux, the Scribbler robot, and the Lab's structure.	Linux commands, a Robot Program, compilers, execution cycle
2. Follow a Path	Students write programs to convert temperatures from one scale to another, inches to seconds and degrees to seconds via robot velocity. They also read in a path from the standard input stream and print out data.	Arithmetic, least-squares regression, i/o stream, while loops, conditional statements, include statements, robot movement functions
3. Braitenberg Vehicles	Students build a few of Braitenberg's vehicles.	Retrieving sensor data, loops, normalization, uphill analysis, downhill invention
4. Navigator Class	Students begin creating their Navigator class, add a few methods and write a driver program to solve a simple problem (writing a word).	Header files, scope, a full robot program, classes
5. Light Following (Navigator Class)	Students write a program to have the robot follow a light source using the robot's camera.	Arrays, pixels, camera functions, enumeration, switch statements, decision methods
6. Blob Following (Navigator Class)	Students expand their light following methods to construct a method to follow a specific object, or blob.	Robot training, pointers, blob picture functions
7. Line Following (Navigator Class)	Students write a program to have their robot follow a line on the floor using the robots IR sensors	Vectors, line sensor retrieval, casting, state programming, decision loops, delete operator
8 and 9. Panorama (Parts I and II)	Students write a program which commands the robot to take a series of pictures, then stitches them together (part one) at the computed points of best overlap (part two).	Two dimensional arrays, vectors, pairs, main function arguments, error messages, complex nested for loops, memory management, pass by address, scope
10. Auto Pilot and Manual Drive (Navigator Class)	Students write interactive programs which allow a user to manually drive the robot or specify a file with movement directions.	Functions for parsing, cstrings, File I/O
11. Obstacle course or Creative assignment	Students work in teams to solve a maze, obstacle course, or self determined problem.	Behavior-based decision making, open ended

Of course, we will not be able to fully evaluate the effectiveness of this process until the grades are submitted for the fall CS 102 courses. This would technically be the final stage.

IV. RESULTS

At the end of the project, we produced eight complete labs, two partially completed ones, and one lab idea for the final lab. These labs are intended to span the entire course material. Table I summarizes our final labs. It includes the lab's title, a short description, and the concepts that the students are intended to encounter in the lab.

Lab eleven is incomplete because the final description of the maze or open ended project and requirements of the lab will depend greatly on how successful the students are at solving the rest of the labs. There is the possibility that some of the labs will not be completed, or that the students will be very successful and will create even more behaviors for their robots. The demands of the final project should reflect what the students have accomplished in the course and utilize a meaningful portion of what they have built during the semester.

V. SUMMARY AND FUTURE WORK

This summer's project is only a small part of a nationwide effort to revitalize Computer Science education. I have confidence that the new set of labs my partner and I have developed will be a positive addition to this movement. The set of labs cover a large and important section of C++ and computing techniques, and they do so in an interactive and engaging manner. The C++ API is also straightforward and should be applicable to courses taught with the Scribbler robot outside of the University of Tennessee. Next semester's course will be an experiment, and the results of the class—how well the students do and how many of them choose to continue studying Computer Science—will be analyzed and changes will inevitably be made to the curriculum to reflect these results.

VI. ACKNOWLEDGMENTS

This project would not have been possible without the collaboration of my partner, Allison Thompson; the guidance of my mentor, Dr. Lynne Parker; the knowledge and skills of Richard Edwards; and the encouragement and resources of Ms. Wallace Mayo and Dr. Bruce MacLennan.

REFERENCES

[1] Doug Blank. Ipre philosophy. *IPRE Wiki*, May 2008. Found at: http://wiki.roboteducation.org/IPRE_Philosophy.

[2] Intro to computing with robots, 2007. Found at: http://www.static.cc.gatech.edu/classes/AY2008/cs1301-robot_fall/index.html.

[3] IPRE, editor. *Learning Computing With Robots*. IPRE, second edition, 2008. Found at: http://wiki.roboteducation.org/Learning_Computing_With_Robots.

[4] Deepak Kumar. Cs 110: Introduction to computing, 2008. Found at: http://cs.brynmawr.edu/Courses/cs110/spring2008/.

[5] University of Tennesse at Knoxville. Cs 102 labs, 2007-08. Found at: http://www.cs.utk.edu/cs102/index.php?id=3.

[6] Keith O'Hara. Robot panoramas. *IPRE Wiki*, July 2008. Found at: http://wiki.roboteducation.org/RobotPanoramas.

[7] Jay Vegso. Enrollments and degree production at us cs departments drop further in 2006/2007. *CRA Bulletin*, March 2008. Found at: http://www.cra.org/wp/index.php?p=139.
[8] J. Wallace-Mayo. Cs 102 class notes, 2007-08. Found at: http://www.cs.utk.edu/cs102/.