AC 2009-1777: COMPARING THE USE OF A GRAPHICAL PROGRAMMING LANGUAGE TO A TRADITIONAL TEXT-BASED LANGUAGE TO LEARN PROGRAMMING CONCEPTS IN A FIRST-YEAR COURSE

Kathleen Harper, Denison University

Kathleen A. Harper is a Visiting Assistant Professor in the Department of Physics & Astronomy at Denison University and has actively taught in the Fundamentals of Engineering for Honors (FEH) Program. She has also been an Instructional Consultant with Faculty & TA Development at The Ohio State University. Dr. Harper earned her BS in Electrical Engineering at Case Western Reserve University and her Ph.D. in Physics at The Ohio State University, specializing in physics education. Following her Ph.D., she worked as a postdoctoral fellow in the Physics Education Research Group at Ohio State with Alan Van Heuvelen.

Richard Freuler, Ohio State University

Richard J. Freuler is the Faculty Coordinator for the Fundamentals of Engineering for Honors (FEH) Program in the OSU Engineering Education Innovation Center, and he teaches the three-quarter FEH engineering course sequence. He is also a Professor of Practice in the Aerospace Engineering Department and Associate Director of the Aeronautical and Astronautical Research Laboratory at Ohio State. Dr. Freuler earned his Bachelor of Aeronautical and Astronautical Engineering (1974), his BS in Computer and Information Science (1974), his MS in Aeronautical Engineering (1974), and his Ph.D. in Aeronautical and Astronautical Engineering (1991) all from The Ohio State University.

Stuart Brand, Ohio State University

Stuart H. Brand is an Engineering Lab Supervisor for the First Year Engineering Program at the Ohio State University College of Engineering. He earned his BS in Physics from The Ohio State University in 1997, after previously serving as a nuclear reactor operator and instructor in the U.S. Navy, at NPTU Charleston, MTS-635 and aboard the USS Atlanta, SSN-712.

Craig Morin, Ohio State University

Craig E. Morin is a Design Engineer with MindWare Technologies in Columbus, Ohio where he develops medical research equipment. Previously, he was a Graduate Teaching Associate with the OSU Fundamentals of Engineering for Honors (FEH) Program where he taught labs and developed course materials. Mr. Morin earned his BS in Electrical and Computer Engineering (2004) and his MS in Biomedical Engineering (2008), both from The Ohio State University.

Patrick Wensing, Ohio State University

Patrick M. Wensing is senior honors student in the Department of Electrical and Computer Engineering (ECE) and has served as a Teaching Assistant for the OSU Fundamentals of Engineering for Honors (FEH) Program. He is also an undergraduate research assistant, working in the area of robotic locomotion. Mr. Wensing will graduate with his B.S.E.C.E. from The Ohio State University in June 2009.

John Demel, Ohio State University

John T. Demel is Professor of Engineering Graphics in the Engineering Education Innovation Center. Dr. Demel is the Faculty Coordinator and teaches for the First-Year Engineering Program (FEP). He helped create and develop the FEP program. Dr. Demel earned his B.S.M.E. at the University of Nebraska-Lincoln (1965) and his Ph.D. (1973) degree in Metallurgy from Iowa State University. He is a registered professional engineer in Texas. He was the institutional Principal Investigator for the Gateway Engineering Education Coalition from 1992 – 2003.

Comparing the Use of a Graphical Programming Language To a Traditional Text-based Language To Learn Programming Concepts in a First-year Course

Abstract

The research study sought to compare whether students can learn programming concepts using a graphical programming language instead of a text-based language. A small group of students was taught their first programming course using LabVIEW while the control group was using C/C++. One result showed that the C/C++ students (control group) had slightly better performance (10%) on equivalent final examination problems than the LabVIEW students. A second result showed that both groups of students performed equally in MATLAB programming exercises – their ‘second’ programming language. A third result compared the programming beliefs of the pilot and control groups with the instructors’ beliefs. The beliefs survey was administered at the beginning and end of the quarter. The pilot group of students exhibited a shift to more expert-like beliefs. This paper provides details about the processes and problems used in this investigation. The work described here began in early 2007 and was completed in 2008. This project was funded by the National Instruments Foundation.

Introduction

Ohio State’s First-Year Engineering Program was developed as part of a National Science Foundation (NSF) funded research and development project[1]. The honors sequence covers engineering graphics and CAD in the Autumn Quarter (Engineering H191), C/C++ and MATLAB in the Winter Quarter (Engineering H192), and a design/build project in the Spring Quarter (Engineering H193). The design/build project has teams of four students each building a small, autonomous robot to complete a specified set of tasks. H193 makes use of the graphics and programming skills developed earlier in H191 and H192. The student teams use the MIT Handy Board[2] as the controller for their robots, and the students write their own program for the controller in Interactive C. Prior to this study, the primary programming language used in the H192 course was C/C++ with a short introduction to MATLAB. C/C++ was covered in eight weeks, and the introduction to MATLAB took about two weeks. Until very recently, the required course providing an introduction to computer programming for engineering students was only offered either in C/C++ or FORTRAN.

In Engineering H191 and H192, there is a hands-on laboratory exercise almost every week. Many of these labs, especially the ones for H192, require the students to collect and analyze data. They have traditionally used C/C++ and MATLAB to do the data analysis. Data acquisition hardware and LabVIEW software are used for a number of the hands-on laboratory experiments.

This first-year engineering environment was selected as the test bed to investigate engineering student learning and beliefs about computer programming in an introductory course. The study sought to compare whether students can learn programming concepts effectively using a graphical programming language instead of a text-based language.

Current Text-Based Programming Learning Environment

In H192, the C/C++ language has been taught using a textbook and the GNU C++ compiler (g++) under Linux. The students typically have used the GNU Emacs editor for composing programs and learned Linux commands for compiling, linking and running their programs. The content includes conditional statements, loops, arrays, functions, pointers, characters and strings, formatted input and output, file handling, and objects and classes. Parts of the C/C++ laboratory exercises require the students to read data from a file and analyze the data. Two of these exercises also are analyzed using MATLAB so that the students can see the comparison. The current C/C++ and MATLAB texts are by Deitel and Deitel[3] and Gilat[4], respectively.

Laboratory Component

Three of the laboratory exercises were 1) firing a model rocket engine and taking thrust data in real time, 2) riding an instrumented bicycle and taking strain gage data, and 3) using a falling ball viscometer setup and taking position and time data. As noted above, the students used their own C/C++ or MATLAB programs to analyze data. Students wrote programs to read in and count the data points and to numerically integrate the area under the thrust versus time curve for the model rocket. In doing so, students were able to determine the total impulse for the rocket engine under test. They found maxima and minima in the bicycle strain data to determine peak stresses on the bicycle frame. By analyzing successive video frames captured with a webcam type of camera in the viscometer experiment, students determined the terminal velocity of the falling ball and subsequently were able to calculate the viscosity of the fluid.

The Graphical Programming Language Learning Environment

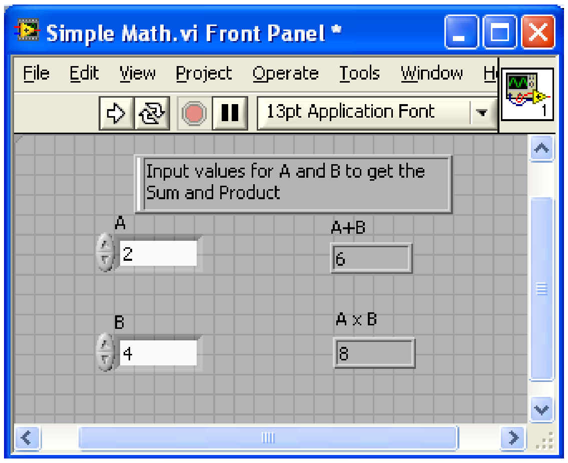
The graphical programming language can provide all of the features of the text-based language. There are graphical icon equivalents for most of the C/C++ statements used in Engineering H192. The programs are prepared by using the Front Panel for input (controls) and display (indicators) and a Block Diagram window (processing). The Front Panel window is used to run the program, to provide for user input, and to see the program output. Running the program from the Front Panel is equivalent to the compiling, linking, and running of a program in the text-based programming approach.

C/C++ and MATLAB are sequential languages and the program flow is driven by the language statements, whereas LabVIEW programs are data driven. Figure 1 shows a program written in C that prompts the user for input and then displays the results on the screen.

```
#include <stdio.h>
int main ( )
{
  int A, B, AplusB, AtimesB ;
  printf (“ input 2 numbers, A B”) ;
  scanf (“%d%d”, &A, &B) ;
  AplusB = A + B ;
  AtimesB = A * B ;
  printf (“  A + B = %d\n”, AplusB) ;
  printf (“  A x B = %d\n”, AtimesB) ;
  return 0 ;
}
```

Figure 1. A simple C language program that prompts the user for two numbers and then adds and multiplies the two numbers and displays the result on the screen.

Figures 2 and 3 below show the LabVIEW Front Panel and the Block Diagram conceptual equivalent, respectively, for the C language program shown in Figure 1. When the program is executed, the data flows from the inputs to the Add and Multiply blocks, and then to the outputs. Unlike the C example, the execution order of the add and multiply operations is nondeterministic. A functional equivalent of the C program would require the use of LabVIEW structures to force the sequential, deterministic execution order.


Simple Math.vi Front Panel *
File Edit View Project Operate Tools Window H
13pt Application Font

Input values for A and B to get the Sum and Product

A: 2
A+B: 6
B: 4
A x B: 8

Figure 2. LabVIEW Front Panel.

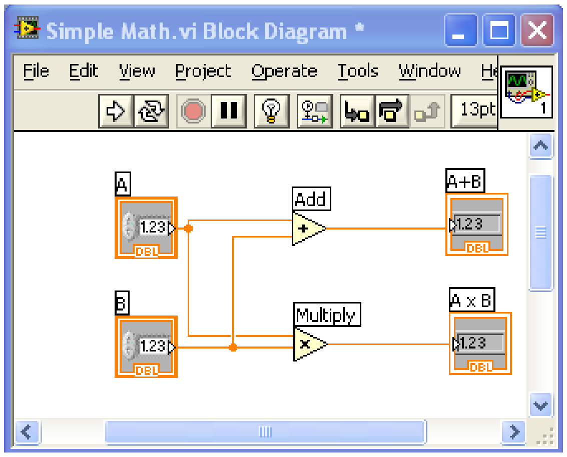

Simple Math.vi Block Diagram *
File Edit View Project Operate Tools Window H
13pt
A
B
Add
Multiply
A+B
A x B
1.23
DBL

Figure 3. LabVIEW Block Diagram.

Research Work

Problems were developed for the graphical environment that were equivalent to the existing H192 C/C++ problems. These included the use of mathematical operators, conditional operators for 'if' and 'while' constructs, input from keyboard and mouse, and loop structures. The choice of functional versus conceptual equivalence was made on a case-by-case basis to preserve the learning concepts of each problem. Graphical icon programming components that parallel the C/C++ components were identified, and a sample set of equivalent components is shown in a table provided in Appendix I. Not every equivalent component utilized in the study is included in the sample set. Many of the C/C++ class notes were directly adapted for teaching the graphical programming language, while others were developed to encompass the topics traditionally covered in H192. The pilot course used *LabVIEW 8* by Bishop[5], but the order of presentation of concepts and topics was modified to accommodate the study. An introduction to MATLAB was taught to both groups in the latter part of the course. The Mathscript feature available in LabVIEW 8, which affords math-oriented, textual programming ("MATLAB-like") capabilities in LabVIEW, was not used so that the MATLAB instruction could be exactly the same in both the pilot and control groups of the study.

Timeline

Winter 2007 – determined the graphical equivalent icons for the C/C++ statements, developed parallel graphical solutions for some C/C++ problems
Spring 2007 – modified the C/C++ lecture notes for the graphical programming language, continued development of graphical solutions for the C/C++ problems
Summer 2007 – tested and refined the notes and problems with a group of teaching assistants and instructors
Autumn 2007 – advertised the pilot section of Engineering H192 (called ENG H494L), selected a group of H192 students to take the pilot course
Winter 2008 – taught the Engineering H494L version and assessed the learning of programming concepts using common exam questions and beliefs surveys

The Pilot and Control Groups

All of the students in the engineering honors sequence in the autumn quarter were asked if they wanted to be a part of the pilot program in the winter quarter. We contacted departmental advisors and two departments, Chemical & Bio-molecular Engineering (ChBE) and Mechanical Engineering (ME), agreed to work with us by allowing their students to substitute the pilot course for the regular course. Fourteen students volunteered for the pilot, providing a somewhat small sample size. All of these students were in ChBE or ME.

Students in the pilot group were matched with students not in the pilot who consented to participate in the comparison group. Students were first matched on previous computing experience, then by intended major, and then by gender (where possible). In aggregate, the samples were well-matched. Four of the fourteen in the pilot and matching control groups had similar prior programming experience. Twelve of the control group were ME majors and two were ChBE majors. There were five women in the pilot group but only one in the control group.

Assessment

The assessment plan was two-pronged. The first was to compare student performance on common questions on the final exams. The second was to administer a programming beliefs survey both pre-course and post-course.

I. Exam Performance -- The final exams contained two styles of problems that were of interest for this study. Two problems on the final exam asked the students to write programs to accomplish specified tasks. These problems were worded identically and were graded on very similar scales. Statistically, the C/C++ group outperformed the LabVIEW group by about 10% on each problem ($P < 0.0034$ for problem 1, $P < 0.0202$ for problem 2). This comparison was also done with a Mann-Whitney test. While the overall scores on these problems only differed by slightly more than a point on average, the 10% difference was not expected. One explanation might be found in the fact that this was the first attempt at teaching these programming concepts in LabVIEW, and thus, there are opportunities to improve the LabVIEW instructional presentations and materials to achieve the same quality level and thoroughness found in the materials of C/C++ course. Further work is necessary to establish exam questions and grading rubrics for common questions in LabVIEW and C/C++ that are verified to be substantially equivalent. Fundamental differences between LabVIEW and C/C++ subordinate the significance of this result to that of the following direct comparisons.

There were two identical MATLAB problems on the H192 and H494L final exams. These problems were graded by the same person for all students in both courses. A statistical comparison between the two samples was done using the Mann-Whitney test, a nonparametric test used to determine the independence of two samples or ordinal numbers. The results showed that there were no significant differences in performance ($P < 0.1164$ for the first problem, and $P < 0.4129$ for the second problem). This indicates that both groups of students were able to take the programming knowledge they had acquired during the earlier part of the term and apply it equally well to the new language. The LabVIEW experience did not hinder students in learning a new programming language, even though the new one was text-based.

II. Beliefs -- To develop a programming beliefs survey, 25 items that were thought to have some relevance to programming were adapted from the 40-item Maryland Physics Expectations Survey[6] (MPEX). These items are statements about programming and the learning of programming that were evaluated on a 5-point Likert scale. As this was the first administration of this instrument, it was expected that some of the questions would be more reliable than others. To get a measure of this, the survey was administered to five programming instructors, all with experience teaching C/C++ and two of whom taught LabVIEW. All of the instructors were in complete agreement on nine of the 25 statements. These nine statements were then used as the basis to compare the student responses.

A comparison was done between the two student groups using a Wilcoxan Signed-Rank test, which is an appropriate nonparametric test for testing for independence between pairs of data points. The pre-course responses on the survey by the two groups were indistinguishable ($P < 0.1936$) from each other. The post-course comparison showed that the two groups shifted their opinions such that the two groups were now significantly different ($P < 0.0061$). Since there was no preconceived notion about which group would be more like the instructors, each sample's post-course responses were compared to the instructor responses. The findings were that the C/C++ students bordered on being distinguishable from the instructors ($P < 0.0808$), whereas the LabVIEW students were completely indistinguishable from the instructors ($P < 0.3639$). Thus, the students in the LabVIEW course seemed to develop more expert-like views regarding the nature of computer programming and how one learns it.

These results are intriguing and should be investigated in greater detail. For this reason, sample exam questions and belief survey questions are not provided here. It was thought that the slightly lower exam performance by the LabVIEW pilot group compared to the C/C++ group might possibly be addressed by improved LabVIEW instructional materials. While unexpected, the more expert-like views of the LabVIEW group would provide a strong basis from which to advocate use of a graphical programming language in an introductory course. Follow-on work should involve further refinement of the beliefs survey, as well as probing the responses on individual questions. It is believed that some of the differences between the post-course responses of the two groups will have logical relationships to the differences in the two platforms, but whether or not this is really the case will only be able to be determined through interviews of students in the two courses.

Summary

The project sought to compare whether students can learn programming concepts using a graphical programming language instead of a text-based language. A pilot group volunteered to learn programming using LabVIEW. From the rest of the students in H192 a matching group was selected to form the control group. A common set of problems were written for both the text-based course and the graphical icon-based course. The current C/C++ instructor notes for Engineering H192 (C/C+) were re-written for use with the Engineering H494L (pilot) section. A beliefs survey was administered to the students at the beginning and end of H192 and H494L. Selected problems on the final exam were used to compare the performance in programming using LabVIEW and C/C++, and two exam problems were used to compare performance in programming using MATLAB. The results showed that the understanding of computer programming concepts acquired in learning and programming in a first language (either C/C++ or LabVIEW) were carried over to the learning and programming in a second language (MATLAB). However, when comparing the programming performance in C/C++ versus LabVIEW on two identical problems, it appears that the C/C++ students did just slightly better. The beliefs surveys indicate, however, that the students in the LabVIEW pilot course may have developed more expert-like views regarding the nature of programming and how one learns it as determined by the fact that the students in pilot course more closely matched the instructors. All of these findings should be probed in greater detail.

Acknowledgements

This project was funded by the National Instruments Foundation.

References

1. Demel, J.T., R.J. Freuler, and A.W. Fentiman, "Building a Successful Fundamentals of Engineering for Honors Program", Proceedings of the 2004 American Society for Engineering Education Annual Conference, June 2004.
2. Martin, F.G., "The Handy Board", Internet: http://handyboard.com, accessed March 2007.
3. Deitel, P.J., and Deitel, H.M., *C How to Program - 5th Edition*, Prentice-Hall, New York, 2007.
4. Gilat, Amos, *MATLAB – An Introduction with Applications*, 2nd Ed. John Wiley & Sons, New York, 2007.
5. Bishop, Robert H., *LabVIEW 8 – Student Edition*, Pearson Prentice Hall, Upper Saddle River, NJ, 2007.
6. Redish, E. F., R. N. Steinberg, and J. M. Saul, "Student Expectations in Introductory Physics," *American Journal of Physics* 66, 1998, 212-224.

Appendix I - Table of C/C++ Text-based and Graphical Icons-based Environments

C/C++	Graphical Icons
array	array
for	for (N times)
while(test)	while (test)
if (test)	case (includes if and case) or select
case	case (includes if and case) or select
printf	display or indicator
scanf	input or control
fopen	open, create or replace file
fprintf	format into file
fscanf	scan from file
fclose	close file
program structure is visible in text only	program structure is visibly revealed in graphical images
sequential statements	graphical icons wired in data-flow sequence block diagram
editor (Emacs)	LabVIEW programming environment
compile, link, run	display front panel and run