J Sci Educ Technol (2016) 25:427–438
DOI 10.1007/s10956-016-9603-y

Development of a Computer-Assisted Instrumentation Curriculum for Physics Students: Using LabVIEW and Arduino Platform

Wen-Hsuan Kuan[1] · Chi-Hung Tseng[2] · Sufen Chen[3] · Ching-Chang Wong[2]

Published online: 30 January 2016

Abstract We propose an integrated curriculum to establish essential abilities of computer programming for the freshmen of a physics department. The implementation of the graphical-based interfaces from Scratch to LabVIEW then to LabVIEW for Arduino in the curriculum 'Computer-Assisted Instrumentation in the Design of Physics Laboratories' brings rigorous algorithm and syntax protocols together with imagination, communication, scientific applications and experimental innovation. The effectiveness of the curriculum was evaluated via statistical analysis of questionnaires, interview responses, the increase in student numbers majoring in physics, and performance in a competition. The results provide quantitative support that the curriculum remove huge barriers to programming which occur in text-based environments, helped students gain knowledge of programming and instrumentation, and increased the students' confidence and motivation to learn physics and computer languages.

Keywords Curriculum design · Graphical-based platform · Scratch · LabVIEW · Arduino · Instrumentation

✉ Wen-Hsuan Kuan
wenhsuan.kuan@gmail.com

[1] Department of Applied Physics and Chemistry, University of Taipei, 1, Ai-Kuo W. Rd., Taipei, Taiwan

[2] Department of Electrical Engineering, TamKang University, 151, Yingzhuan Rd., New Taipei City, Taiwan

[3] Graduate Institute of Digital Learning and Education, National Taiwan University of Science and Technology, 43, Sec. 4, Keelung Rd., Taipei, Taiwan

Introduction

Computer programming has a great and widespread influence on all aspects of science and engineering. For physicists, the development of numerical analysis methods (SIAM 2016) promotes explorations from macroscopic astronomy and aerology to microscopic particle physics, from classical Newtonian mechanics, hydrodynamics to electro-optical quantum dynamics in semiconductors and novel materials. For example, the rapid development of GPU and MPI technologies (NVIDIA Inc 2015; Sur et al. 2006), which raise computing power than ever before, help physicists to explore complicated spin dynamics and phase transitions (Beaurepaire et al. 1996; Sandvik and Moessner 2006; Schau et al. 2015; Garcia-March and Carr 2015) by means of the quantum Monte Carlo method (Anderson 1986) or the density matrix method (Kohn 1996). Thriving in the age of technology, our scientists are putting very high hopes on the invention of quantum computers. Computer programming is not far from us. Actually, in our everyday life, it is as close as the application of cryptology to the communication protocol standard for safeguarding the transfer of classified information(Schneier 1993). More friendly and exciting for us, the great contribution of computer programming is artificial intelligence which has given birth to robotics, machine learning, data mining, and 3D visualizations (Kovahi and Provost 1998; Gareth et al. 2013; Russel and Norvig 2009). All these applications dramatically diversify and rich our living styles.

In recent decades, the advances in network technologies and cloud services have brought revolutionary changes in computing, programming and management of information systems. The dream of the Internet of Things has been realized (Tse 2005; Platt 2009; Atzori et al. 2010) and still continues with the development of the integration of

wireless transmission, microcontroller utilities, sensors, and monitoring devices. These achievements on the one hand signify the connection of computer language and modern technology, and on the other establish a new platform or a new look for people who have many interesting ideas and want to write their programs just for fun (Schmidt 2012; McComb 2015; Erwin et al. 2000). This unprecedented scenery has changed the way of thinking and learning; the power of computer programming has shed light on science and engineering practices that are highly emphasized in the next generation science standards (National Research Council 2012). As president Obama told the world, everyone should learn how to code, and he himself became the first president to write programs (https://www.whitehouse.gov/blog/2014/12/10/president-obama-first-president-write-line-code). Is that not persuasive enough for the young to master these tools to ensure that they remain on the cutting edge? We are lucky enough to witness the fourth evolution of industry. Students in physics departments are never too young to participate in the feast of technologies or too naive to be outstanding designers. On the contrary, with good training, they shall play an important role in interdisciplinary professions (Adams et al. 2011; Zhan et al. 2014).

Owing to the authors' theoretical research background, the importance of computer programming for physics students has been deeply imprinted in our minds. Therefore, we designed a 3-year computing course syllabus. We chose the C language for the beginners, the sophomore students, because C is well known as having been developed with rigorous and structural built-in syntax. Besides, it is also well known as being specifically flexible, extensive, efficient, and portable. These features mean that C is frequently adopted in the development of solid state circuits, image processing or modern optics design, areas which our students are likely to be engaged in their future careers. For practical use, it is important for students to obtain authorized utilities including compilers, libraries, and friendly user interfaces at the lowest cost. From these view points, all Linux operating systems support the gcc compiler, and the Window system's Bloodshed Dev-C++ provides an integrated development environment (IDE). In addition to convenient hardware structures, there are also rich software supplements (Press et al. 1992; Perimeter Scholars International Course 2015; Algorithms and Libraries for Physics Simulations 2015; Laboratory for Tensor Decomposition and Analysis 2015) to help solve frontier scientific problems.

However, some frustrating experiences have bothered both students and teachers in the past years: (i) Students have no idea about computer structure; (ii) students born in the 90s have never heard of DOS or UNIX so they have to spend more time getting used to text-based environments and commands; (iii) the logic and algorithms are confusing and to figure out the causality is not easy; (iv) the practices of pointers are difficult and the memory misallocation always prompts serious interruptions; (v) the rules to identify the correct path to open files and access data are complicated; (vi) the error messages include confusing terminology; and (vii) extra scripts are required for real-time plots. We were facing the depressing fact that many students tended to give up learning anything about programming even when they became senior students; they assumed that they could never write a correct program for simple calculations. We were not willing to see this happen, and so we started to examine if perhaps the tool was inappropriate and if the contents were too difficult for our students who mostly ranked at the 50th percentile in the College Entrance Exam. To know whether these puzzles could be resolved with a simultaneous systematic and methodological approach became the emergent issue.

Since education is for construction, our first goal is to change stiff impressions about programming and to provide students with confidence and fun. This was the original intent of the development of the CADPL curriculum. Furthermore, we wanted to know if the inadequate situations could be reduced if we started the training from the freshman rather than the sophomore year. Inspired by Scratch (2015), an interactive and easy-to-use graphical IDE developed by MIT Media Lab which provides free space for drawing and considerable catalogues of command bricks, making it an ideal kit for children and parents to enjoy programming together, we started to organize a graphical-based programming course for the freshmen. Knowing that the training aimed to strongly reinforce computer programming with physics, and to help freshmen complete some delicate experiments, we finally designed a three-phase graphical-interface slot, from Scratch to LabVIEW then to LabVIEW for Arduino. The investigation was aimed to explore if the CADPL course could successfully help students to read and write simple programs, to do data access and analysis, and even to achieve remote instrument control with LabVIEW.

The significance of this paper is to illustrate the original intent and the detailed progress in the development of a curriculum CADPL to improve the programming abilities of students in a physics department using a graphical platform. The following paragraphs focus on demonstrating our curriculum design, including introductory training topics set to help students become familiar with the new materials as quickly as possible. We also provide some examples to give clear illustrations and verification of the students' progress. In the third section, we introduce the methodology of efficacy evaluation by means of pretest and post-test questionnaires, and retrospective interviews. Via long-term quantitative and qualitative investigations,

we were able to observe if the graphical-based environments and cross-platform kits implemented in CADPL did in fact bring confidence in programming to students and raise their learning motivation for physics and computer applications. Finally, we give a brief conclusion and perspective on our work.

Curriculum Design

Advantages of Graphical Programming

Although naively one can use MATLAB or C to do tedious calculations, students might take more time to recognize the algorithms and the syntax of these text-based development environments. To help students without any programming background to merge a computer language with physics and electronic instruments, the use of a graphical-based platform might be practical and feasible. To verify our conjecture, we proposed three topics in our project: Scratch, LabVIEW and LabVIEW for Arduino. The main advantages of graphical programming for beginners are listed as follows:

- *Easy to understand:* When users drag visual components on the screen, they have already decided the sequence of how the program is executed.
- *Visual and straight forward:* LabVIEW provides plenty of visualized components on its virtual panel, such as meters, slides and light indicators. Students can design and adjust the panel according to different functional requirements. This feature can help students acquire the necessary instrumentation skills in the future. Another superiority of visual programming components is that with the help of a graphical programming language, students can better understand the concept of how data flows from one module to another.
- *Minimum effort for debugging:* Graphical programming languages take advantage of colors and shapes, which give clear hints for users to design different scenarios or an animation in a few hours by doing the Scratch puzzle. Users can also do data transfer by wires but not by text variables in LabVIEW. In this way, they will not be trapped in the rigorous syntax of a text-based environment, but be able to build programs relatively quickly.
- *Suitable for data logging and analysis:* With the help of LabVIEW's graphical environment, students can have a better understanding of the process of instrumentation from establishing communication with equipment, continuously logging data, and closing the connection. Besides, LabVIEW provides rich libraries and modules for users who can directly use them to process data with mathematical functions but without getting confused about the advanced mathematical skills such as derivative/integration, filtering or nonlinear fitting, an example demonstrated in Fig. 1.

Although friendly to beginners, graphical platforms might not be the best for professional designers since they tend to become a spider's web of complex code and are structurally improper for precise scientific computing. Despite these minor limitations, students do still benefit from the underlying instruction. The 17 weeks of teaching materials followed the learning objectives of basic execution, syntax, algorithms and human–computer interfaces. First and foremost, students would gain much more confidence and joy from coding.

Course Projects

In Table 1 we show a complete curriculum design in which each phase was implemented for 3 or 4 weeks. The 2 hour/week curriculum includes not only compact and basic training but also advanced introductions and tasks to achieve comprehensive learning via integrated materials. The class flow of weekly runs includes: (a) Introduce new components, demonstrate and analyze sample programs; (b) Proceed with the curriculum to focus on the integration and application of the new elements; (c) Execute an in-class challenge according to the day's materials; (d) Discuss and share the in-class exercise; and (e) Illustrate the after-class assignment. Furthermore, the contents in each phase are organized to be closely associated with specific learning objectives to give intensive and extensive instructions from phase to phase.

Phase 1: Scratch

In this phase, the learning objectives are to help students to write their first programs and inspire their imagination and creativity to combine elements into compounds. At the end of this phase, the demonstration time aims to fertilize the students' communication ability.

Having its commands categorized in functions such that it is possible even for young children to build their first programs with puzzles in different shapes and colors in half an hour is the real advantage of playing Scratch. Since we chose Scratch as the first and special platform of programming learning, we spent some time giving a brief introduction to the nature of programming and emphasizing basic concepts and commands in programming.

Students have to let the cat not just stand there but have to initiate an action. Therefore, we first impressed upon them that 'An actor is a kind of variable.' Then it would naturally come to the question 'How can we continue the

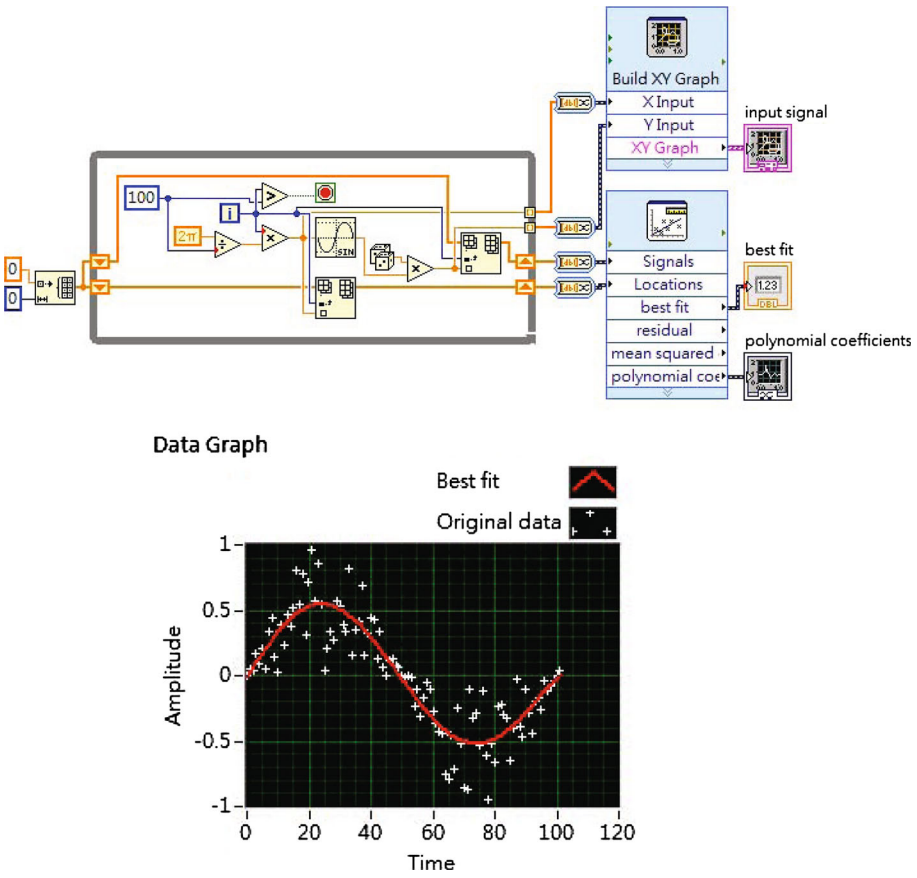

Fig. 1 Nonlinear fitting of data within a sinusoidally approximated distribution

actions?' and the next question 'What if...' These questions tend to inspire a desire to know how to bring different conditions into action, and a strong motivation to learn loops and to incorporate if-else conditional structures and events into loops. To this end, students can understand the essentials of the execution of a program and are expected to be able to design a miniature interactive game. Herein, full imagination can be freely injected and intensified by the advanced skill, interaction, which can be viewed as the preliminary concept of instrumentation by getting the user's mouse or keyboard to control the actor's properties. Figure 2 demonstrates an animation game produced by a student for a sectional project assigned to deliver a realization of integration.

From making an animation, the Scratch platform proves itself as having at least two advantages: (i) It would take much longer and requires more advanced skills to create the same scenery on a text-based platform, for example, Java; and (ii) It takes no effort and is advisable to transplant the scenery to Phase 3 as instrumentation material due to the compatibility of the Arduino–Scratch interface. These experiences would guide the students to learn how to acquire signals in the real world with a microcontroller utility, Arduino, so as to achieve further control of a third-party device.

Phase 2: LabVIEW and Arduino

While Scratch successfully played the role of bringing students to a wonderland of computer programming, for the authors the next question to ask must be 'What computer language can bring scientific applications to physics major students?' This was the motivation to introduce LabVIEW, a powerful graphical-based platform, from Phase 2, to advance to a comprehensive knowledge of programming. In this phase, understanding syntax and algorithms is the most significant learning objective. To achieve the goals we organized a 4-week schedule to introduce fundamental concepts, from data types, data flow logic, conditional structures to arrays and subVI. These constitute the complete essential knowledge of programming. Since practice leads to progress, quantitative exercises are apparently practical and of immediate concern.

Therefore, exercises applied for the generation of mathematical series, geometrical graphs, or physical formulas are expected to multiply qualitative progress for freshmen in the sciences. In Fig. 3 we demonstrate some of the students' work. The Fibonacci series generator emphasizes the shift register and the logic representing accumulation. A polygon generator shapes discrete points

Table 1 The curriculum design

Week	Topic	Contents	Learning objectives
Phase 1			
1.	Scratch first program	Introduction/Movement/Sound and color/Scene	Execution
2.	Scratch	Structure: for loop, if-else and event	+ Imagination
3.	Scratch	Interaction/Get user input to control characters' properties	+ Communication
		Project: animation game	
4.	Project Discussion	Demo and Share	
Phase 2			
5.	LabVIEW Introduction	Data types /Frequently used modules/Dataflow programming	+Syntax & Algorithm
6.	LabVIEW Introduction	Structures/Waveform chart /XY graph	+Scientific & Application
7.	LabVIEW Introduction	Structures/Arrays	
8.	LabVIEW Introduction	Arrays/Bundles/SubVI	
		Project: prime number generator interpolating curves	
9.	Project Discussion	Demo and Share	
10.	Arduino	Electronic basics /IDE introduction/Digital IO:	+ Electronic
11.	Arduino	Analog input: potentiometer, photo-resistor/ Analog output: PWM control	Practice
		Project: joystick controlling LED luminance	
12.	Project Discussion	Demo and Share	
Phase 3			
13.	Interface Design	Display data on LabVIEW's front panel/Understand controllers and indicators	+Human –Computer Interfaces
14.	LabVIEW for Arduino	Retrieving data from Arduino/Combining external device	
15.	LabVIEW for Arduino	Control seven-segment display	+Instrumentation
		Project: Revise LabVIEW kernal & modify a C-like Arduino code	+Innovation
16.	LabVIEW for Arduino	Continuous data logging & getting statistical quantities	Experiment
17.	Final presentation	Demo and Report	

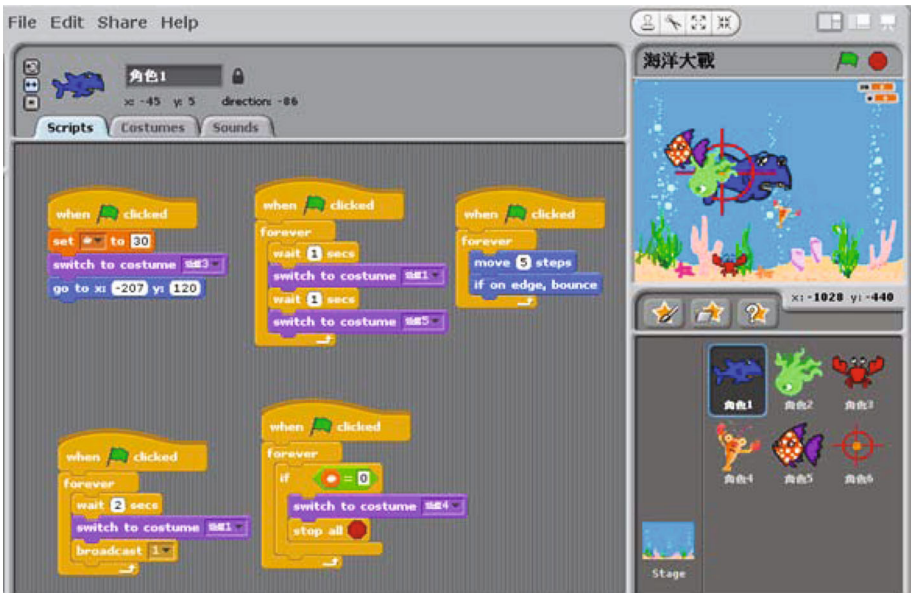

Fig. 2 Interactive animation designed by Scratch

into geometry, and the concepts of random walk and statistics are delivered by a Monte Carlo catapult.

At the end of the LabVIEW introduction, project design is implemented as integrative practices for knowledge application. Two practices, prime number generator and data interpolations, are designed to help integrate while structure, case structure, logic elements and arrays. The former tends to aggregate nested judgment and filtering

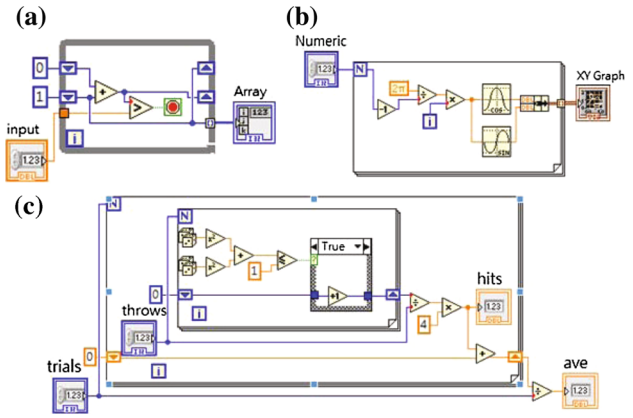

Fig. 3 **a** The Fibonacci series generator emphasizes the shift register and the logic representing accumulation. **b** A polygon generator shapes discrete points into geometry. **c** A Monte Carlo catapult delivers the concepts of random walk and statistics

strategies. While difficulties in piping interpolation and sorting skills into data analysis often involve advanced numerical methods in text-based environments, they can be avoided with LabVIEW from the later project. In Figs. 4 and 5 we demonstrate students' works. Overall, the friendly features make it possible for freshmen to carry out multi-level thinking and design on the LabVIEW platform.

From the viewpoints of scientific experiments, electronic practice plays an indispensable role in instrumentation. Therefore we arranged the schedule in phase 2 to introduce the Arduino platform for the basic design and control of digital and analog circuits. Furthermore, the use of Arduino IDE would bring another experience of programming that can be taken as preparation to shift to a C-based environment. Therefore in this section, we try to bridge communication between computer and circuit elements via very basic electronic manipulation. We expected that the experience of a joystick controlling an LED would open the door of advanced instrumentation with LabVIEW supplements.

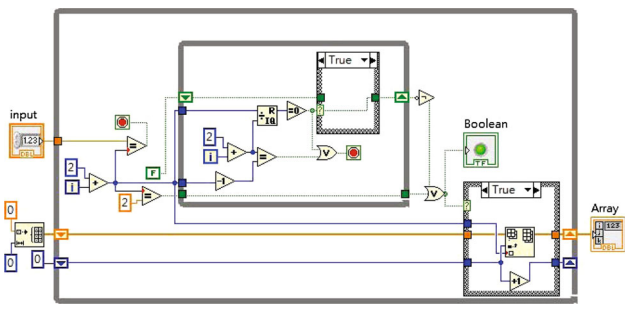

Fig. 4 A prime number generator would return all prime numbers with their values less and equal to an arbitrary input integer *N*

Phase 3: LabVIEW for Arduino

Whenever complication and precision in science and technology become unavoidable challenges, automatic control may be considered as a feasible and practical solution. Within reason, the learning objectives in phase 3 are focused on human–computer interfaces and instrumentation. The topic LabVIEW for Arduino instrumentation tends to create a closer connection between computer programs and scientific experiments. We expected that the efforts in the whole semester would allow the students to demonstrate some innovation in the experiments.

As low-priced and developable hardware, Arduino has been extensively implemented in basic physical computing and interactive devices. In addition to its affordable price, Arduino kits support many computer languages such as Scratch and LabVIEW used in this curriculum. By using built-in functions, students can easily write a simple program to realize the remote control via Arduino. In such a preparative training, for example, they can connect with the real world via controlling the LED luminance by photosensitive resistors; they can estimate the distance between the robot and an obstacle from collision reaction or ultrasonic sensors. In other words, they will be able to control devices and various kinds of actuators (McClain 2014). Programming is not for virtual computations but can be for interactions.

Via the first structural diagram to configure Arduino by LabVIEW (Fig. 6) we can introduce the most important start-up: The Call of Interface. In principle, to activate the connection with advanced electronic instruments, students firstly need to let the calling string '*IDN?' work. Therefore in the first class of phase 3 the universal rules for fundamental instrumentation are presented: First, initialize and configure the external utility. Then write/read data continuously to/from low level pin nodes for testing. Finally, close the connection. After right checking the consistency of the Baud rate, board type, and communication protocol, students might replace the 'READ' module with any 'SENSOR-READ' element and start the remote control of the Arduino microcontroller utility.

In Fig. 7, we demonstrate a class assignment on acquiring the *X*- and *Y*-axis status of the joystick module with two potentiometers connected to Arduino's analog input pins. We expected that a good human–computer interaction experience could be provided by mimicking a game-playing practice.

As mentioned at the very beginning, we expected that students would be less frustrated learning C in the future if they had some training from this curriculum. Therefore in the 15th class we arranged comparisons of the interface and language syntax of LabVIEW and C-like Arduino IDE by using a seven-segment LED (Fig. 8). There are three

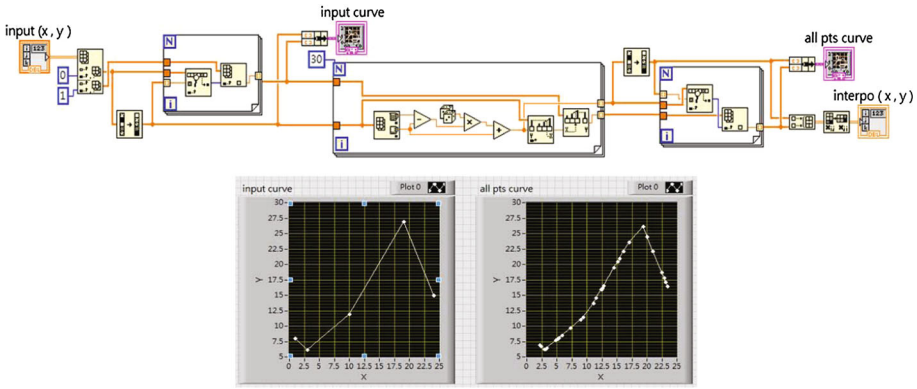

Fig. 5 Multi-level thinking and design carried out by a 2D sorting and interpolating program

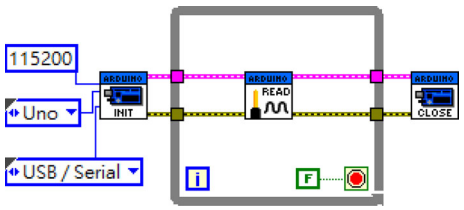

Fig. 6 Basic communication structure between LabVIEW and Arduino

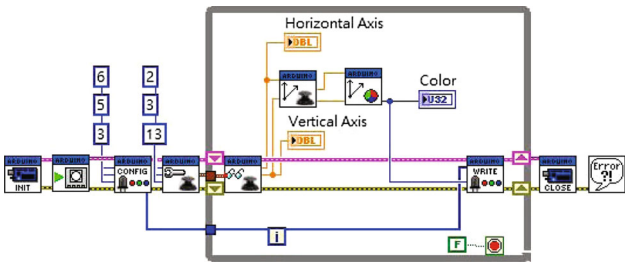

Fig. 7 Acquiring joystick's axis status to control LED luminance

reasons why it was appropriate to introduce this example. Firstly, the practice including common anode/cathode testing of LED modules would verify the training of electronic basics in the 10th class. Secondly, when repeatedly executing 'digitalWrite', students must ask 'Is there a clearer way to code?' Then it is time to emphasize the structures again and introduce the syntax of arrays and functions of this C-like language. It is also time to convey the importance of structural and concrete expressions. Finally, imitation leads to promotion. Students must have the ability to identify the paths of the officially-supplied examples and enter the kernel and understand the master codes. In the seven-segment LED case, they can prove themselves by revising the LabVIEW example and letting the word 'HELLO' be displayed.

While the combination of computers, sensors and mechanism design results in a feasible way to design and encode the motion of a robot with simple formulas by LabVIEW, their applications in the visualization of physical education have been found to be surprisingly good by many research works (López-Rodr'guez 2015). Therefore in the last lecture, we introduced the concepts of data logging and how to get statistical information from raw data by a line-following robot. Those concepts can be easily realized with the data logging module and data viewer supplement in NI MAX (National Instruments Inc 2015). Figure 9 shows the line graph of the motor encoder as a function of time, which also includes some statistical quantities such as average and standard deviation automatically generated by LabVIEW. It is conceptually important since a real-time display of recorded values that brings information of failures and affective conditions would be helpful to improve the efficacy by properly adjusting formulas in the program. This lecture demonstrated that modular and low-cost robotics can not only be used for industrial improvement but is also good for physics education (Hwang et al. 2011; Gómez-de-Gabriel et al. 2011; McLurkin et al. 2013).

Research Questions

To evaluate whether this curriculum has positive effects on knowledge gain and learning motivation for the freshmen, we designed a questionnaire to help identify the areas requiring further improvement. The investigation focused on three research questions:

(a) What did the students gain from this curriculum?
(b) Did this curriculum improve students' confidence and willingness to learn computer languages?
(c) Did this curriculum improve students' attitudes toward physics?

The first question aims at the gain of skill or knowledge, whereas the other two are concerned with determination, which cannot be ignored, especially for the students in our department.

Fig. 8 A C-based countdown monitor constructed by user-defined array

```
byte pin_define[8]={2,3,5,6,7,8,9,4};
byte seven_seg_digits[10][7] = {
{ 0,0,0,0,0,1,0 },  // = 0
{ 0,0,1,1,1,1,1 },  // = 1
{ 0,1,0,0,1,0,0 },  // = 2
{ 0,0,0,1,1,0,0 },  // = 3
…. and so on
void sevenSegWrite(byte digit) {
  for (byte seg = 0; seg< 7; ++seg) {
digitalWrite(pin_define[seg], seven_seg_digits[digit][seg]); }}

void loop() {
  for (byte digit = 10; digit > 0; --digit) { delay(1000);
sevenSegWrite(digit - 1); }}
```

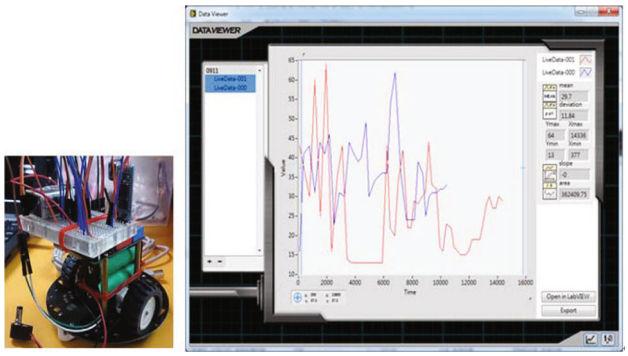

Fig. 9 Line graph of motor encoder as a function of time

Research Methods

Subjects

CADPL is an elective course for freshmen, and our subjects were 14 freshmen in the Department of Applied Physics and Chemistry. This department recruits average achieving students from high schools. The students' academic performance as indicated by the college entrance exam ranged from the 48th–53rd percentile among the college students nationwide based on data of the past 6 years (2009–2014). They typically had low scores in science and mathematics and demonstrated low motivation and confidence in science learning. The participants' mean scores of mathematics and physics were 50 and 49 out of one hundred in the college entrance exam, whereas the top one-quarter of students nationwide scored higher than 80.

The department recruits 35 students each year. They take both physics and chemistry classes in the first year and choose either physics or chemistry as their major in their sophomore year. In the past few years, they tended to choose chemistry and avoided numerical classes, such as numerical analysis or computational physics.

Procedures

This study used a pre- and post-test design to investigate the effectiveness of the curriculum. The subjects filled in an anonymous questionnaire about their understanding of basic programming and learning motivation in the beginning. They completed the course and filled out the questionnaire again as the post-test in the 17th week. A follow-up interview was administered six months later to examine whether they still remembered the content. Meanwhile, students' decisions regarding their majors (physics versus chemistry) and course taking patterns were collected.

Measurements

Estimations of learning efficacy were implemented via

1. quantitative questionnaires regarding basic understanding of programming concepts and learning motivation and confidence,
2. qualitative retrospective interviews concerning how well they can apply the knowledge taught in the class,
3. numbers of student selecting physics as majors and taking advanced numerical classes in the following year, and
4. competition performance in an annual national programming competition.

It should be noticed that all instruments concerned mostly the elementary knowledge gain from this course and then the confidence and attitudes toward computer applications and physics in the future.

Questionnaire

This questionnaire included 24 items divided into two scales, with a five-point rating scale of strongly disagree, disagree, neither agree nor disagree, agree and strongly agree. The first scale focused on knowledge of program design and practice, including fundamental understanding

of programming and algorithms, basic understanding of electrical components and circuitry, and instrumentation (Table 2). The second scale was regarding self-confidence and learning motivation (Table 3). In the data analysis, the five ratings were replaced by values 1–5. The reliability as measured by Cronbach α was 0.73. The differences between pre- and post-test were examined by t test.

Retrospective Interview

The retrospective interviews were used to understand if, after six months, they were able to carry out tasks using the concepts taught in CADPL. The interviews covered 17 tasks guided by the question, 'If there was a part-time job to complete some tasks by LabVIEW, would you be able to take it?' Immediately following their answer to each task, the interviewer assessed their competency using a four-point rating scale: absolutely cannot, barely, maybe, and definitely. The rating was done as part of the interviews and was checked immediately with the interviewee to ensure the correctness of interpretation. The ratings were given in numerical values 1–4 for further analysis. The 17 tasks measured their programming, mathematical operation, debugging, and instrumentation abilities.

Major Choice

Since programming ability as well as performance in math has prejudiced students against physics, the number of students choosing physics versus chemistry as well as taking advanced numerical classes was also taken as an indicator of the effectiveness of the CADPL curriculum. Students' choices in the past 5 years were used as a comparison. There were 3, 10, 14, 13, and 7, respectively, out of 35 students selecting physics as their major in their sophomore year. Moreover, in the past 5 years, there were 3, 7, 10, 10, and 0 sophomores who continued taking advanced numerical classes.

Table 2 Knowledge of programming and instrumentation

A. Fundamental understanding of programming and algorithm
1. I can finish a program on graphical-based platform such as Scratch or LabVIEW
2. I know how to use conditions and variables to control loops
3. I know the difference between data types, such as integer, float and character
4. I know how to pass parameters from one module to another
5. I know how to use subVI to make compact programs
6. I understand the dataflow sequence when the program is running
B. Basic understanding of electrical components and circuitry
7. I can upload the sketch to Arduino and update the firmware
8. I know the difference between analog and digital components
9. I know the difference between common cathode and common anode of RGB LED
10. I can use breadboard and jump wire to complete a circuit
11. I know how to control LED luminescence thorough a joystick
12. I can use keyboard or mouse to control the program execution
C. Instrumentation
13. I can use different visual aids (graph, table, pie chart, etc.) to represent data
14. I know how to use arrays to do data logging
15. I know how to retrieve statistical result from raw data
16. I know how to configure the interface parameters to monitor Arduino's status
17. I know how to convert motor encoder value to actual moving distance of a robot
18. I know how to modify sample programs to remotely control robot and MCU

Table 3 Confidence and learning motivation

D. Self-confidence and learning motivation toward computer applications
19. I wish to learn more about programming and instrumentation tools in C or MATLAB
20. This course provides materials with a clear introduction of programming
21. I become more proficient on programming and instrumentation
22. I know how to find solutions from books, classmates or internet resources
23. Whenever there is a problem, I want to find out where goes wrong
24. I think a graphical-based programming platform is easy to learn and I will strongly recommend this platform to the beginners

Annual Competition

The courage to participate in an annual national competition of programming and their records were partly considered as a reference of measurement.

Results

The statistical analysis of the questionnaire listed in Table 4 quantitatively shows that concerning the scale of knowledge gain, in all dimensions, the scores in the post-test were significantly higher than those in the pretest. As comparisons of the means and standard deviations of the pre- and post-tests, the t values in all dimensions ranged from 4.55 to 5.55, corresponding to the Cohen's-d values from 1.72 to 2.10. The effects were large. It should be noted that dimensions B and C corresponding to circuitry and instrumentation are difficult in the practical aspect. However, the results were not a concern since the hardware practices set good examples of multiple applications and were not an issue for computer programming for the beginners. Overall, the main purpose of promoting a fundamental understanding of programming and algorithms was successfully achieved.

For the scale concerning confidence and learning motivation, the scores of mean, standard deviation, t value, and Cohen's-d for both the pre-/post-tests were 3.36/3.82, 0.50/0.48, 2.43, and 0.94. The scores in the post-test were also higher than those in the pretest.

From a qualitative perspective, the curriculum was beneficial because, in the students' opinion, the course was helpful for:

1. attaining the basic concepts of programming algorithms and design methods;
2. attaining the basic skills of circuitry;
3. getting practical experience of instrumentation; and
4. improving the motivation for further studies in computer applications.

Analysis of the retrospective interviews is reported as follows. The average scores for programming ability, mathematical operation, debugging, and instrumentation were 2.931, 2.528, 2.778, and 2.767, respectively. The data indicated that more than 50 % of the interviewees were able to program with some hints or references. The results provided inspiring and crucial support for the effectiveness of CADPL.

Table 4 Statistical analysis of the questionnaire in knowledge scale

	Mean (Pre/Post)	SD (Pre/Post)	t value	Cohen-d
A	3.04/3.90	0.510/0.492	4.55	−1.716
B	2.49/3.27	0.427/0.418	4.86	−1.846
C	2.56/3.61	0.517/0.481	5.55	−2.103

Of the 35 students in the 2014 class, the number of students who chose to major in physics increased from the 5-year average of 9.4 to 18, of which 14 had joined CADPL, and all of them enrolled in the advanced computer class. We therefore have strong confidence to claim that the increase in student numbers was an indication of the efficacy of CADPL.

Finally, the courage to participate in the annual competition of programming and their records could also be partly considered as a reference of measurement, since none of the former students had chosen to participate. We are delighted and proud that four of the students volunteered to participate and passed the preliminary stage of the competition. Furthermore, three of them also registered for the certification test in October 2015. This was a great achievement both for the instructor and the students due to their courage to join the international competition which took place six months after one-semester training in the curriculum.

Concluding Remarks and Perspectives

In this article we demonstrate how we brought physics department freshmen into the world of computer programming by a three-phase course of 17 weeks. In this course we applied a graphical-based interface to help students get into the flow of functions and get keys for debugging with fewer barriers.

We started from Scratch then moved to LabVIEW, and then we incorporated LabVIEW with Arduino such that students could not only learn the basic programming rules but also learn how to make connections between computers and external utilities. The serial instructions led to a very good experience of instrumentation. Following our introductory and consecutive arrangements for each session, the students were required to complete basic programming practices every week and an integrated project before the end of the session. Thanks to these short but systematic introductions, the students developed the abilities to demonstrate game animation by Scratch, prime number generator and regular polygon drawer by LabVIEW. The good use of 2D array sorting, interpolation, and graphical demonstration has turned the integrated training in CADPL into a very good account, thus indirectly but successfully proving the effectiveness of CADPL. The experience of completing the seven-segment LED code by both LabVIEW and Arduino IDE bridged the application of computer language and electronics. In addition, they were also required to know how to use a breadboard, read resistor

color codes, build simple parallel and series circuits, and test with multimeters in the hardware training. In the final project we successfully let students start using C without any of the setbacks or depression that we had encountered previously.

The efficacy of the curriculum was evaluated via a questionnaire, interviews, the number of students who chose to major in physics, and the competition records. The results of the questionnaire showed that for both scales, the knowledge gain in programming and self-confidence, the scores in the post-test were significantly higher than those in the pretest. The data of the retrospective interviews indicated that more than 50 % of the interviewees were able to program with some hints or references six months after the completion of the curriculum.

As mentioned above, our freshmen did not perform sufficiently well on their entrance examination. The scores revealed that they had low motivation and confidence in studying physics and had deep impressions of the necessity of good mathematics ability for physics. Interestingly, this game-feeding generation suggested that knowledge of information technology is the privilege of those majoring in computer engineering. This was why it has been difficult for us to promote programming education. Even worse, the freshmen were all frustrated with learning the C language before CADPL was adopted. As a consequence, only one-third and sometimes even fewer of the freshmen would continue studying physics in their sophomore year, and very few of them had confidence to take advanced numerical classes or to choose theoretical groups in graduate school.

Therefore the authors felt deeply encouraged knowing that after taking CADPL, more than half of the freshmen chose physics as their major in their sophomore year, and four of them participated in an annual competition of programming and passed the preliminary stage. Science is no longer for the tests but can be touched, manipulated, and designed by themselves. We believe that the idea of starting from the graphical platform works, and the progress of these freshmen is great and very inspiring.

We have organized a multi-year plan to help students in the physics department learn computer programming skills such as LabVIEW, C and MATLAB in the application of theoretical simulations, instrumentation, and data acquisition in scientific and engineering studies. We are delighted to share some experiences and achievements of the first term and are encouraged to implement laboratory innovation in the second term. We expect to push ahead with the Maker experiences from CADPL and to provide the new generation with different views of knowledge and learning, everywhere and anytime; not say it but do it; nothing old but everything new. We expect that a close connection among practical abilities, computer programming, and communication mobility would span a whole new world of study and design for the young.

Acknowledgments This work was supported in part by the Ministry of Science and Technology of the Republic of China under MOST 104-2511-S-845 -009 -MY3.

References

Adams R, Evangelou D, English L, Figueiredo AD, Mousoulides N, Pawley A, Schifellite C, Stevens R, Svinicki M, Trenor JM, Wilson DM (2011) Multiple perspectives on engaging future engineers. J Eng Educ 100(1):48–88

Algorithms and Libraries for Physics Simulations (2015) Algorithms and Libraries for Physics Simulations, USA. http://alps.comp-phys.org/mediawiki/index.php/Main_Page

Anderson HL (1986) Metropolis, Monte Carlo and the MANIAC. Science 14:96–108

Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805

Beaurepaire E, Merle JC, Daunois A, Bigot JY (1996) Ultrafast spin dynamics in ferromagnetic nickel. Phys Rev Lett 76(22–27): 4250–4253

Erwin B, Cyr M, Rogers C (2000) LEGO engineer and ROBOLAB: teaching engineering with LabVIEW from kindergarten to graduate school. Int J Eng Educ 16(3):181–192

Garcia-March MA, Carr LD (2015) Vortex macroscopic superpositions in ultracold bosons in a double-well potential. Phys Rev A 91(3):033626

Gareth J, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning. Springer, New York

Gómez-de-Gabriel JM, Mandow A, FernandezLozano J, Garcia-Cerezo AJ (2011) Using LEGO NXT mobile robots with LabVIEW for undergraduate courses on mechatronics. IEEE Trans Educ 54(1):41–47

Hwang KS, Hsiao WH, Shing GT, Chen KJ (2011) Rapid prototyping platform for robotics applications. IEEE Trans Educ 54(2): 236–246

Kohn W (1996) Density functional and density matrix method scaling linearly with the number of atoms. Phys Rev Lett 76(17–22): 3168–3171

Kovahi R, Provost F (1998) Glossary of terms. Mach Learn 30(2–3):271–274

Laboratory for Tensor Decomposition and Analysis (2015) Laboratory for Tensor Decomposition and Analysis, JP. http://www.bsp.brain.riken.jp/TDALAB/

López-Rodr'guez FM, Cuesta F (2015) Andruino-A1: low-cost educational mobile robot based on android and arduino. J Intell Robot Syst. doi:10.1007/s10846-015-0227-x

McClain RL (2014) Construction of a photometer as an instructional tool for electronics and instrumentation. J Chem Educ 91:741–750

McComb G (2015) Robot builder's bonanza, 4th edn. Chi-Hung Tseng, Gotop, Chinese Edition. ISBN 9789863411369

McLurkin J, Rykowski J, John M, Kaseman Q, Lynch AJ (2013) Using multi-robot systems for engineering education: teaching and outreach with large numbers of an advanced, low-cost robot. IEEE Trans Educ 56(1):24–33

National Instruments Inc (2015) Where Can I Download the Latest Version of Measurement & Automation Explorer (MAX)? National Instruments Inc, USA. http://digital.ni.com/public.nsf/allkb/F046BF3874E45BFF862574E2006EA83B

National Research Council (US) (2012) A framework for K-12 science education: practices, crosscutting concepts, and core ideas. In: Committee on a Conceptual Framework for New K-12 Science Education Standards. Board on Science Education Division of Behavioral and Social Sciences and Education. National Academies Press, Washington, DC

NVIDIA Inc (2015) What is GPU accelerated computing? NVIDIA Inc, Santa Clara, CA, USA. http://www.nvidia.com/object/what-is-gpu-computing.html

Perimeter Scholars International Course (2015) Quantum spin simulations. Perimeter Scholars International Course, USA. http://physics.bu.edu/~sandvik/perimeter/index.html

Platt C (2009) Make: electronics (learning by discovery). O'Reilly media, CA

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C: the art of scientific computing, 2nd edn. Cambridge University Press, NY

Russel S, Norvig P (2009) Artificial intelligence: a modern approach. Prentice Hall, NJ

Sandvik AW, Moessner R (2006) Correlations and confinement in nonplanar two-dimensional dimer models. Phys Rev Lett 73(14):144504

Schau P, Zeiher J, Fukuhara T, Hild S, Cheneau M, Macr T, Pohl T, Bloch I, Gross C (2015) Crystallization in Ising quantum magnets. Science 347(6299):1455–1458

Schmidt M (2012) Arduino: a quick start guide (pragmatic programmers), Chinese Edition: Chi-Hung Tseng, Gotop ISBN: 9789866076510

Schneier B (1993) Applied cryptography: protocols, algorithms, and source code in C. Wiley, NJ

Scratch (2015) Create stories, games, and animations share with others around the world. Scratch, USA. https://scratch.mit.edu/

SIAM (2016) SIAM journal on numerical analysis: a publication of the Society of Industrial and Applied Mathematics, Philadelphia, PA, USA. https://www.siam.org/journals/sinum.php

Sur S, Koop MJ, Panda DK (2006) High-performance and scalable MPI over infiniband with reduced memory usage: an in-depth performance analysis. In: SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing. ACM, New York, NY, USA, p 105

Tse D (2005) Fundamentals of wireless communication. Cambridge University Press, Cambridge

Zhan W, Porter JR, Morgan JA (2014) Experiential learning of digital communication using LabVIEW. IEEE Trans Educ 57(1):34–41