A Dual Abstraction Mobile Computing Framework for Developing Programming Skills

Ahmed Abukmail[1]

Abstract – In this paper we propose a teaching framework for a mobile computing-based programming course. The target students will include first-time programmers, both computer science majors and non-majors. Normal daily use of a computer is the only assumed pre-requisite for this course. Student motivation is a key factor to the success of this course in developing programming skills. As a result we adopt a mobile computing approach to develop these skills. In particular we target the development of mobile applications on smart phones due to their popularity as well as their increasing capabilities. We start the first half of the course by introducing the students to a high-level method of program development via Google's App Inventor where program syntax is not prevalent in the learning process. Using abstract concepts, we develop the students' logical thinking without the burden of learning the full details of the syntax of a programming language. Throughout this phase of the course, basic programming and logic concepts, such as variable declaration, loops and conditionals will be explained. In the second half of the course, we introduce a more detailed view of programming to illustrate the exact same programming concepts discussed in the first half of the course. In specific we will show the students how to build Google Android applications using the Android SDK. We believe when students observe the development of these applications on their mobile devices, their attention and their motivation will increase. This is because they will be advancing the capabilities of their own mobile devices with applications that have the potential to be useful in their daily use.

Keywords: Program Syntax, Semantics, Mobile Computing, Google Android SDK, Google App Inventor.

1. INTRODUCTION

The learning curve for many students when first exposed to computer programming is significantly steep. This often leads to students' frustration with the learning process and with the ability to see a future in computer programming in specific, and computer science in general. Many attempts have been made to alleviate this frustration to quickly demonstrate to the student that he/she can learn the material. One of the first and quite successful attempts,used in the past, is Karel the Robot [1]. A newer tool that was developed and is based on Karel the Robot is JKarelRobot [2] that incorporates additional language features in developing exercises for the students. JKarelRobot is language and platform independent and supports many languages. Other tools have been developed and used, and each of these tools had its own level of success [3]. Some of these tools are based on telling a story, Jeroo is one of these tools [4]. The success of Jeroo as a pre-cursor to higher level languages was also reported in the classroom in [5]. The work in [6] utilizes UML and interactive tools for learning the object concept. However, students, new to computing, tend to also have some difficulties with narrative-based programming tools such as Karel the Robot and Jeroo [1,4] regardless of the intended use of the tools.

We believe that teaching computer programming must evolve with the current trends in computing. Mobile devices are becoming extremely popular especially among the younger generation. A significant number of students are in possession of smart phones. Recent studies have found that the number of college students using smart phones is on the rise. According to [7], the number of students using smart phone has gone from 27% to 49% in a little over a year. Additionally, a study done in [8], shows that 53% of on-campus users use smart phones with a large majority of these users using phones that rely on downloading applications. Motivating these students to be able to modify

[1] University of Southern Mississippi, School of Computing, 118 College Dr. #5106 Hattiesburg, MS 39406, ahmed.abukmail@usm.edu.

their phones by implementing their own applications will yield a valuable teaching tool for developing their programming skills. Therefore, a simpler platform must be utilized to allow for easy learning experience. We believe that Google's App Inventor [9] is a suitable tool for learning how to develop computer programming more quickly.

One of the major difficulties that face students when first introduced to computer programming is balancing between learning a new language and applying it. This is very similar to learning natural languages where a person is trying to express an idea (semantics) with the newly acquired grammar and vocabulary (syntax). Google's App Inventor requires very little syntax involvement to express one's semantics. This will lead students to focus more on understanding problem solving with little involvement from syntax. Basically, the developer visually designs the application. This gives a higher level of abstraction (granularity) for learning computer programming.

In this paper, we describe a proposed course that will be used to start by utilizing App Inventor to develop the problem solving skills of new programmers. However, we understand that developing programming skills needs to involve learning syntax and semantics of a computer language. Therefore, once we teach the students how to develop visually, we will start introducing them to the Android SDK [10] that will allow them to start by designing the visual components of an application, but instead of using the block editor, they will start inserting their own code based on applications that they have earlier developed using App Inventor. Moreover, we will start teaching them how to program under other platforms besides the mobile platform. This will allow for higher level of abstraction in learning how to program. The advantages of this proposed course is that any student can take it, and it can be taught for both computer science majors as a first course in computer science. It can also be taught for non-majors taking it as a campus-wide service course. After this course students can either apply what they learned in their major field, or, if they're CS majors, they can continue on to take the second computer science course.

2. PROGRAM WHAT YOU UNDERSTAND

It is essential to start a programming course by building students' confidence in their abilities. They need to understand that they can perform tasks if they understand the program. Therefore, and according to [5], we start the course by giving students a reassurance of their ability to solve problems. We explain to them that before attempting to solve a problem, it is essential that they understand it and understand all the precise requirements of this problem. This means, without knowledge of the language in which instructions are give, solving it cannot be done. Just like a person, a computer can only perform instructions that it understands. Therefore, we must communicate with a computer using its language. However, this is a cumbersome task to learn, given that a computer only understands 0's and 1's. Therefore we need the help of a translator between our language and the computer's. This is where we explain the concept of compiling.

Once we explain the concept of translation, we start by giving the students an example problem to add two numbers. We emphasize that the instructions are given in English because the instructor and the students speak the same language. We start out by picking a single student and asking the question: "Can you please add two numbers together". Generally, the student's response is something like 5 + 2 = 7. Our response to the student, these are not the two numbers to be added. So, the student would ask, what numbers do you need to add? Before giving the student the two numbers, we usually spend the time emphasizing the importance of input in a program. Once the student is given the two numbers, he/she will then add them, and speak the sum back to the instructor. We then tell the student that he/she was not asked for the answer, and has only been asked to add the two numbers. This gives us an opportunity to emphasize the role of the output in a computer program. At this point, we emphasize that a computer will only do what it is told in the order in which it is told.

Once the student finishes the exercise, he/she understands that input, output, and processing are three separate, yet equally important, aspects of writing a computer program. We then involve the student in another exercise where two large numbers (can not be remembered without writing) are given for addition. When the student is unable to add them, we then emphasize the importance of memory.

We believe that continuing to give this exercise to the students in any course involving new programmers is quite useful as it is the precursor to understanding why we use compilers, why we have a CPU, a keyboard, a monitor, and memory in the computer, and what role each one of them plays. This exercise is normally done on the first day of class.

3. ADVANTAGES OF MOBILE COMPUTING

Recent Advances in mobile computing lead to the popularity and wide use of smart phones. A significant portion of cellular telephone users carries smart phones. It is even more prevalent on college campuses. It also seems to be a trend that more students are dependent on smart phones for their daily use. We believe that urging the students to be able to develop applications for their smart phones will motivate them to learn programming. Many of the students are interested in developing games for their phones. Many of them may have ideas about developing games and other applications, yet they cannot implement them due to their lack of programming knowledge. We believe that our course will help them in accomplishing this task.

4. SEMANTICS FIRST – SYNTAX LAST APPROACH

In the beginning of our course, we would start teaching the students how to develop program with little attention to the syntax of a particular language and the features that come along with the language. What Google's App Inventor provides is the ability for one to implement an application with little attention to syntax. We simply think of a computer program from with a high level of abstraction. The requirement for using App Inventor is an account and permission from Google to be able to get access to App Inventor. One does not need to own a cellular phone running the Google Android Operating System to develop an application. An emulator is available for download. The process of developing an application by using the App Inventor involves familiarity with the Designer window within which there is a "Viewer", and a "Palette" [9]. Developers simply pull components from the Palette and place them in the Viewer in their desired location. It is essential that, either a phone is connected to the computer or the emulator is running. A tutorial illustrating the details of how to develop an application is provided in [9]. Moreover, students must be familiar with the block editor provided as well.

In this teaching framework, which will develop into a full course, we will introduce the students to the various concepts of computer programming such as variables, mathematical operations, logic, if-statement, loops, and other control structures as presented in the block editor within App Inventor. Figure 1 shows an example application that was developed in App Inventor with very little effort to understand syntax. Figure 2, shows the semantics of the add button of the program within the block editor. This application contains two text boxes that will accept two numbers. It contains 4 buttons that when clicked will perform the labeled action (add, subtract, multiply, and divide). The result will show up in the third text box below the listed button.

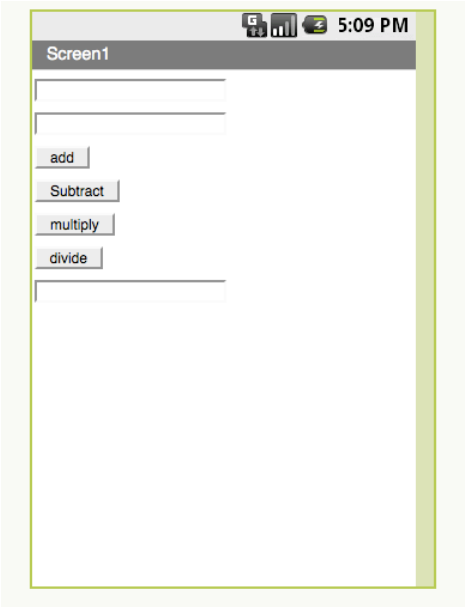

Figure 1: A Simple application developed in App Inventor. It allows for adding, subtracting, multiplying and dividing two numbers.

Once the students grasp the concepts of developing a solution to the simple problems using App Inventor, we will then start by adding more complexity to the logic. For example, we can introduce the concept of the if-statement by adding it to the logic of the fourth button in figure 1. The fourth button is the one that performs division. Therefore, the number in the denominator cannot be a zero. The block editor has blocks containing the if-statement that will allow us to check if the value provided in the second text box is a zero or not. If it is zero, it either will not perform the division, or it could display a large number representing infinity in the third text box.

5. Learning Syntax

In the second half of the course, it would be essential to teach the students that while it is easy to be able to develop with App Inventor, it is necessary for them to learn how to develop for other platforms. Therefore, they must learn how to write code without having to just drag and drop logic blocks into a block editor. At this point, we can start teaching the students how to re-build the applications that they have developed using App Inventor and transform the development under Eclipse [11] and using the Android SDK [10]. We used the Android SDK because it's closely tied to App Inventor and it develops for the same platform, except it requires more programming knowledge, so the transition would be much easier than if we are to move to a significantly different programming environment such C, C++ or Java directly.

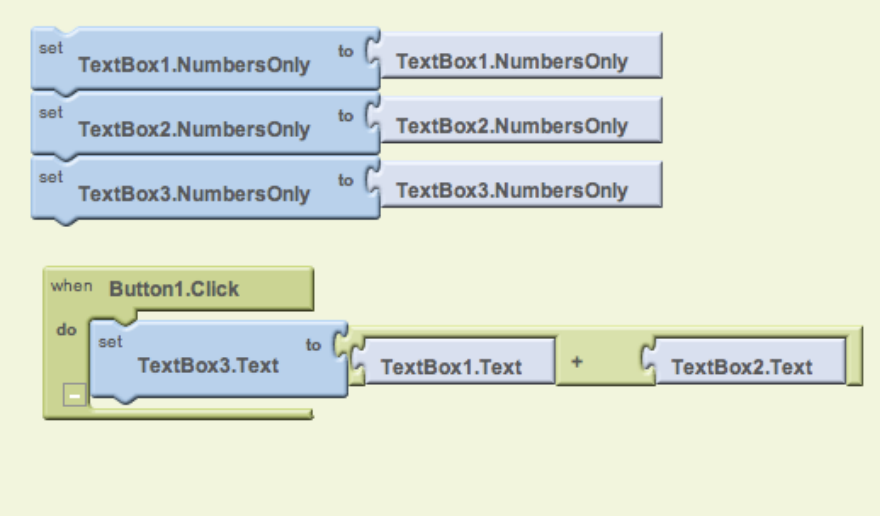

Figure 2: Logic for the text boxes and the add button

It is worth noting that developing for the Android SDK is not a trivial task, and therefore setting up the environment for SDK development under Eclipse is quite essential as we do not want the students to worry about the logistics of setting up the SDK on their own computers. It is even essential to provide a lab session in this course to be able to help the students setup their own computers and help them while using the pre-set lab computers with the SDK. We believe that the students will see the flexibility provided by developing one's own applications with the freedom of writing their own code and further debugging what does not work.

Learning how to develop with the SDK must later lead to the start of teaching the student an introductory programming language such as C, C++ or Java. While it is quite useful to use the Android OS and its development environments, it is essential to emphasize to, and start teaching the students, in the first course, that other languages and other platforms (e.g. Windows, Mac, Linux) are important to learn in addition to mobile environments.

6. Conclusion and Future Work

When teaching students how to program for the first time, the first thing that teachers introduce is the syntax of a language. This usually creates a barrier for the student to learn problem solving as some of them tend to not be comfortable with the small details that are included in a computer language, parentheses, semicolons, and curly

brackets. So, using a semantics-first approach will be quite popular with the students. We believe that utilizing Google's App Inventor in the classroom is quite valuable, as it does not emphasize syntax. It also provides a level of motivation for the students to learn how to program while using their smart phones. Furthermore, migrating the development towards a more traditional programming environment is quite essential as the students need to learn programming in general not only for smart phones and mobile devices.

REFERENCES

[1] R. E. Pattis, Karel the Robot: A Gentle Introduction to the Art of Programming, 2nd ed., Wiley, 1995.

[2] D. Buck, D. J. Stucki, "JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum," In Proceedings of the 32nd SIGCSE Technical Symposium on Computer Sciences Education, Charlotte North Carolina, February 2001, pages 16 – 20.

[3] K. Powers, S. Cooper, K. J. Goldman, M. Carlisle, "Tools for Teaching Introductory Programming: What Works?," In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, Texas, March 2006, pages 560 – 561.

[4] D. Sanders, B. Dorn, "Classroom Experience with Jeroo," Journal of Computing Sciences in colleges, 8(4), 2003, pages 308 – 316.

[5] A. Abukmail, L. Perkins, "Towards the Development of Programming Skill for First-Time-Programmer," ASEE Southeast Section Conference, Blacksburg, VA, April 2010.

[6] S. H. Moritz, G. D. Blank, "A Design-First Curriculum for Teaching Java in a CS1 Course," In ACM SIGCSE Bulletin, 37(2) 2005, pages 89 – 93.

[7] S. Hernandez, "Ball State study shows college students' smartphone usage rising," The Ball State Daily News: http://www.bsudailynews.com/ball-state-study-shows-college-students-smartphone-usage-rising-1.2275899, June 21, 2010. Last Visted: 01/26/2011.

[8] J. Dean, "Smartphone User Survey: A glimpse into the mobile lives of college students," The Digital Media Test Kitchen: http://testkitchen.colorado.edu/projects/reports/smartphone/smartphone-survey/. Last Visited 01/26/2011.

[9] About – App Inventor for Android, http://appinventor.googlelabs.com/about. Last Visited 12/10/2010.

[10] Android Developers, http://developer.android.com/index.html. Last Visited 02/04/2011

[11] Eclipse – The Eclipse Foundation open source community website, http://www.eclipse.org. Last Visited 02/04/2011.

Ahmed Abukmail

Ahmed Abukmail (ahmed.abukmail@usm.edu) is an assistant professor at the University of Southern Mississippi's School of Computing. His primary teaching responsibilities are in computer science. His research includes computer science education, parallel and distributed simulation, and mobile computing.