Enhancing The Teaching Of CS 1 By Programming Mobile Apps In MIT App Inventor

Dr. Kefei Wang, Gonzaga University

Computer Science Department Gonzaga University

Enhancing the Teaching of CS 1 by Programming Mobile Apps in MIT App Inventor

Abstract

This paper presents the development of a curriculum of CS 1 course, which conveys the basics of programming techniques and concepts of Computer Science (CS). To build a solid foundation for prospective CS majors, while also attract students from other subject areas, a new pedagogy of programming was adopted by using MIT App Inventor. It is a blocks-based programming environment that enables students to create apps for Android devices without worrying about syntax. We developed a series of hand-on modules into the course work. Thus, students were motivated to make a grater effort to learn the concepts embodied in the modules. Through this engaging and intriguing method, the freshly designed curriculum helped students master a number of vital skills in problem solving, computational thinking, design, and teamwork. Pre-survey and post-survey were conducted to study the acceptance of the new teaching method. The study indicated the curriculum not only helped motivate students from all majors, but also improved their performance.

1 Introduction

At Gonzaga University, Computer Science 1 (CS 1) is a required course for Computer Science and all engineering majors, including civil, mechanical, and electrical engineering. Each CS 1 class consists of less than 15% Computer Science majors and 70-80% engineering majors. The course covers basic programming concepts, e.g., binary conversions, flow of control, procedural abstraction, and data manipulation by using Python programming language. In addition to programming and Computer Science principles, the course is also emphasized on collaboration, creativity, writing, and communication. Most of engineering students in this course have little to no programming experience and many do not anticipate studying computer science beyond CS 1. While computational thinking is a critical skill for scientists and engineers,[1] engineering students in this course find concepts in Computer Science are rather abstract and irrelevant to their field of study. In this paper, we present the re-design of our CS 1 curriculum that not only prompts to build a solid foundation for prospective CS majors, but also makes the course more attractive and meaningful for students from engineering majors. A series of CS 1 modules are developed by incorporate MIT App Inventor into the the course to enable students to learn programming concepts without being burdened with the syntax of a programming language.

MIT App Inventor, which is developed by Google Inc. and Massachusetts Institute of Technology, is an open-source visual programming environment targeted at novices for creating mobile apps for Android devices. The browser-based graphical drag-and-drop development environment helps students to focus on problem solving and logic thinking without worrying about syntactic errors. In addition, App Inventor provides functionality to use location and motion sensors, as well as the ability to store shared information in the Cloud. By the end of 2013, App Inventor had attracted over two million registered users globally,[2] and the majority of users were educators and students. Many studies have demonstrated that App Inventor is a great way to engage students of varying interest levels and experience. For example, Ericson created an effective and financially sustainable business model for summer camps on App Inventor programming.[3] By teaching App Inventor in a core curriculum course at the University of San Francisco, Wolber believed App Inventor facilitated interactions with the world outside of the classroom.[4] Gray et al. successfully used App Inventor as a platform for teaching CS Principles at University of Alabama.[5] In[6], Hsu and Ching presented their work in developing an online course for mobile app design for graduate students by using App Inventor. These researches reveal the great educational value of App Inventor, which not only increases students' interest and confidence in programming, but also enhances their abilities in computational thinking skills, problem solving techniques, and creativity. Thus, we adopt it in our CS 1 curriculum re-design to deliver a more exciting and engaging learning experience for students from different engineering majors.

The rest of this paper is organized as follows. Section 2 presents the course outcomes and objectives. Section 3 describes the course structure and a series of CS 1 modules consisting of App Inventor projects. Section 4 presents assessment methodology and the assessment findings. Finally, concluding remarks are presented in Section 5.

2 Course Outcomes

In[7], the authors stated: "there is wide agreement that the US engineer of the future will have different attributes than the stereotypical 20th century engineer. These attributes include the technical knowledge that all engineers should have and are included in traditional engineering curricula, and other characteristics related to what have often been called soft skills or professional attributes, including communication, leadership, and entrepreneurial skills." To help instill an entrepreneurial mindset, Kern Family Foundation established a network of colleges called the Kern Entrepreneurship Education Network (KEEN) in 2005. Gonzaga University is one of the 19 colleges in KEEN, and it views the entrepreneurial mindset as a critical need in undergraduate engineering education. Considering students in our CS 1 is predominant in engineering, we propose a curriculum that embedded a set of learning outcomes defined by KEEN with the ABET's student outcomes. In particular, student will learn a number of vital skills that can be transferred to any subject area and contribute significantly to their performance as professionals. These skills include:

- Problem solving skills
- Logic and reasoning skills
- Computational thinking

- Skills to effectively collaborate in a team setting
- Skills to apply critical and creative thinking to ambiguous problems
- Skills to persist through and learn from failure
- Skills to demonstrate voluntary social responsibility

3 Course Structures and Modules Development

The course was design to contain two sessions: App Inventor and Java programming. The App Inventor would be introduced at the beginning of semesters to allow the students to explore basic programming concepts, including logic, conditions, loop, variables, procedures, data abstraction, input and output, and system analysis. The visual approach of App Inventor helped students study event-driven and object-oriented programming without focusing on the syntax of coding. This session consisted of seven modules (see Table1) as supplemental week-long programming projects. All the projects were conducted by students in a pair-programming manner as in[8 9 10] to foster the effective collaboration within a team setting.

The Java programming session would be brought into the class right after the App Inventor session. Java was used as a general programming language to reinforce the proficiency of programming and problem solving skills among students. It also played as a transitional programming language for student to continue their study in Computer Science.

In addition to programming and problem solving skills, the designed modules also include considerable contents on teamwork, social responsibility and ethics. Followings are brief descriptions of designed App Inventor modules:

Module 1 is based on the ”Hello Purr” project from Wolber’s book[11] with a few extensions. It works as a quick start for students to explore the life cycle of app development in App Inventor. The app includes images, sounds, accelerometer sensor awareness, and tap event handlers. Concept of event-driven programming is defined in this module, too.

Module 2 is a free-style ”Drawing” app. Through variables, students can control line styles, colors, and pen shapes. More UI components and screen layouts components are introduced. Drag-and-drop event handler that takes multiple arguments is discussed. Finally, the camera component is embedded in the app so that users can take a picture and then use it as canvas background.

In module 3, students develop a ”Dice Game”, which simulates a casino dice game through random number generators and conditional statements. An accumulator is used to collect the scores. To make the app accessible for persons with disabilities, a *Text2Speech* component is included in this module. Moreover, Students discuss the voluntary societal responsibility, and software as a service for the common good.

A traditional ”Asteroids” game is developed in module 4. In this app the user controls a spaceship using accelerometer. The goal of the game is to avoid the falling asteroids for as long as possible while collecting power-ups that replace lives and increase the score. The score is defined as a

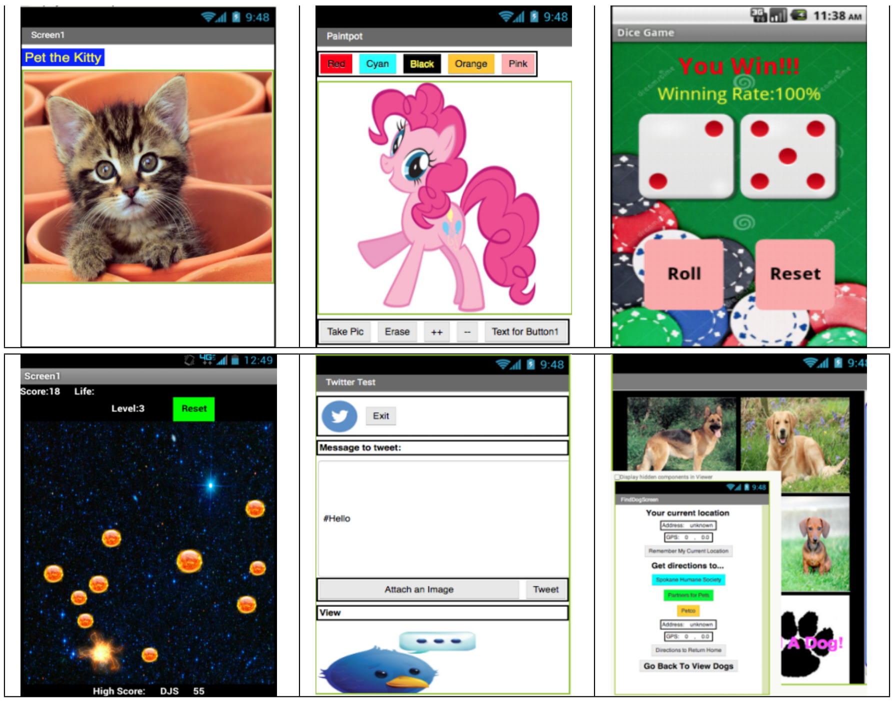

Table 1: Modules 1-6

function of the user's time survived and the number of power-ups collected. Iteration structure is used to go through a list of high scores. Timers are set up to update the positions of the spaceship and asteroids. The game also updates the highest score in a cloud storage through *TinyWebDB* component.

Social networking websites are popular among college students, and module 5 is developed to demonstrate the social networking related technologies through *Twitter* component in App Inventor. Twitter API and OAuth protocol are explored, and students brainstorm to analyze the extensions on the features to enhance user experience. The class also examines potential personal privacy concerns and ethic issues in a social networking community.

Module 6, "Find a Dog", is a multi-screen app to provides information to the user about specific breeds of dogs. This app is helpful for people who are interested in purchasing a dog but want to know more about the different types before making an decision. Dog shelters and pet stores are listed through the *GPS* and *Activity Starter* components. By saving their original location with the *TinyDB* component, the user can find their way back home after successfully navigating to the desired pet store.

Module 7 is the final programming project in App Inventor session. During a three-week period, each student team needs to go through important procedures of software development, including: requirement analysis, software system design, prototypes demo, coding, testing, documentation and presentation. The final report and oral presentation are then assessed by a rubric that is similar to the one described by[12]. In this module, instead of following instructions, students need to demonstrate their creativity and apply skills they have learned from other modules to build an app from scratch. The module 7 is also a good opportunity for students to evaluate their learn-

Survey Questions	Pre-survey	Post Survey
1) Programming is hard.	4.20	3.90
2) I can be creative with computing.	3.80	4.17
3) Learning programming is helpful in my major and career.	3.80	4.24
4) I will stick with a computing problem until I have a solution.	2.92	3.92
5) I am good at solving problems that are ambiguous.	3.67	3.89
6) I am good at working within and contributing to a team.	4.25	4.77
7) I consider going into Computer Science major or minor.	2.21	2.80

Table 2: Mean Scores of the Surveys

ing outcomes, e.g., computational thinking, team work, critical and creative thinking in solving ambiguous problems, and learning from failure.

4 Methodologies and Results

4.1 Surveys for Students Self-evaluation

The modules were deployed in a CS 1 course in the semesters of Fall 2013 and Spring 2014. There were totally 108 students in the classes, and about 80% of them were civil, mechanical and electrical engineering majors. Students conducted each module in team of two, and partners in teams were switched twice during the semester. While most requirements in these modules can be fulfilled on an Android simulator, the sensors, e.g., camera, GPS, and accelerometer are only available on a physical device. About 30% of students in the class used their own Android phones and tablet, and the rest borrowed tablets from the Computer Science department.

To help understand how the App Inventor experience had changed the students' perceptions of computer programming, and also evaluate the effectiveness of proposed new curriculum, pre-survey was conducted at the beginning of the semester, and post-survey was conducted after the module 7 final project. Table 2 briefly summarizes the results of the surveys with mean scores. The scale is 1 through 5, where 1 indicates ”strongly disagree” and 5 indicates ”strongly agree”.

The results from these surveys shows App Inventor modules effectively increased students' interest and confidence in programming. One student praised the new designed modules in the course evaluation: “After studying programming in App Inventor, I have a new interest in Computer Science, and that has sparked my interest in continuing further studies in it”. Results from questions 2-6 show the course not only engaged the students but also developed their abilities in computational thinking, problem solving, and creativity.

Questions 4-6 are KEEN related, and the results indicate that through teamwork, collaboration and reflection, the course allowed students to gain additional insights into entrepreneurship mindset in the context of software development.

Moreover, question 7 shows that after the class, students increase their interest in computer science, thus helps increase the student retention rates in the department.

4.2 Instructor's Evaluation

In addition to the students self-evaluation surveys, the instructor also evaluated students performance on class work, including homework assignments, oral presentations, documentations, midterm and final exams. The scores of the experimented sections were then compared to the sections in Python, which were taught by the same instructor in Spring 2013.

Our summative assessments focused on general programming and problem solving skills. More specifically, the following four categories of learning outcomes were examined:

1. concepts of control flows and subroutines
2. program requirements analysis
3. data structures and data model abstractions
4. algorithm efficiency evaluation

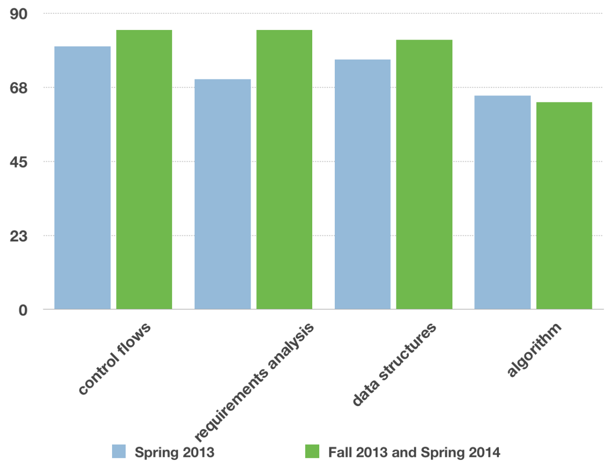

Figure 1: Instructor's Evaluation on Students Performance

In each category, we picked representative questions and calculate the mean scores in the scale of 100% among all the class sections from Spring 2013 to Spring 2014. Figure 1 shows the statistical results of our research.

Clearly, students from the newly designed classes outperformed the students from the traditional Python sections in the category of program structures, requirement analysis, data abstraction and usage of data structures. Students from the App Inventor class showed solid ability to tackle problems of data abstractions and modeling, which is a fundamental and important programming skill in computer Science. As a visual programming language, App Inventor gave students a high-level overview of system structure and data abstraction. Thus helped the students develop problem solving skills in a broader point of view without getting involved into details too early.

Moreover, the nature of event-driven and object-oriented design within App Inventor also contributed to a better understanding of system design among students. This nature encouraged students to explore the possibility of code reuse, objects composition, and inheritance.

Note that in the category of algorithm efficiency evaluation, the App Inventor sections slightly underperformed, but the difference was marginal.

5 Conclusion

Many engineering students have little or no programming experience when they come to CS 1 class. To create an effective and dynamic learning environment, we developed a series of hand-on modules into the course work. MIT App Inventor was adopted in the new designed curriculum to help increase students' interest and confidence in programming. In this environment, students learned programming concepts and computational thinking skills by building mobile apps. The results of the work described in this paper indicate that the new learning approach impact students performance positively and effectively. It also helps develop engineering students' entrepreneurship mindset and societal responsibility. Further improvement of the curriculum design includes additional App Inventor modules to utilize various sensors, new tools to help establish a easier transition from App Inventor to Java, and development of a set of students assessment rubrics.

References

[1] Chenglie Hu. Computational thinking: What it might mean and what we might do about it. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 223–227, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0697-3.

[2] Shaileen Crawford Pokress and José Juan Dominguez Veiga. MIT app inventor: Enabling personal mobile computing. *CoRR*, abs/1310.2830, 2013.

[3] Barbara Ericson and Tom McKlin. Effective and sustainable computing summer camps. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 289–294, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1098-7. doi: 10.1145/2157136.2157223.

[4] David Wolber. App inventor and real-world motivation. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 601–606, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0500-6. doi: 10.1145/1953163.1953329. URL `http://doi.acm.org/10.1145/1953163.1953329`.

[5] Jeff Gray, Hal Abelson, David Wolber, and Michelle Friend. Teaching cs principles with app inventor. In *Proceedings of the 50th Annual Southeast Regional Conference*, ACM-SE '12, pages 405–406, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1203-5. doi: 10.1145/2184512.2184628.

[6] Y. C. Hsu and Y. H. Ching. Mobile app design for teaching and learning: Educators' experiences in an online graduate course. In *The International Review of Research in Open and Distance Learning*, pages 117–139, 2013.

[7] John-David Yoder, Robert Kleine, Don Carpenter, and Cynthia Fry. Spreading the fire: Broadening faculty support for the entrepreneurial mindset. In *OPEN 2013: NCIIA's 17th Annual Conference*, NCIIA 2013, 2013.

[8] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. Improving the cs1 experience with pair programming. *SIGCSE Bull.*, 35(1):359–362, January 2003. ISSN 0097-8418. doi: 10.1145/792548.612006. URL `http://doi.acm.org/10.1145/792548.612006`.

[9] Alan Shaw. Teaching peer-to-peer programming methodologies in introductory computer science courses to facilitate collaborative programming paradigms. *J. Comput. Sci. Coll.*, 27(2):156–165, December 2011. ISSN 1937-4771. URL `http://dl.acm.org/citation.cfm?id=2038836.2038859`.

[10] Charlie McDowell, Brian Hanks, and Linda Werner. Experimenting with pair programming in the classroom. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '03, pages 60–64, New York, NY, USA, 2003. ACM. ISBN 1-58113-672-2. doi: 10.1145/961511.961531. URL `http://doi.acm.org/10.1145/961511.961531`.

[11] David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney. App inventor: Create your own android apps. 2011.

[12] Michael J Peeters, Eric G Sahloff, and Gregory E Stone. A standardized rubric to evaluate student presentations. *American journal of pharmaceutical education*, 74(9), 2010.