AC 2012-4258: ACCELERATING K-12 INTEREST IN COMPUTER SCIENCE USING MOBILE APPLICATION-BASED CURRICULUMS

Mr. Korey L. Sewell, University of Michigan, Ann Arbor

Korey Sewell received his B.S. in computer science from the University of California in 2004, and his M.S. in computer science and engineering in 2007 from the University of Michigan, Ann Arbor. He currently is a doctoral candidate at the University of Michigan, Ann Arbor. He has research interests in high-performance microprocessor design, on-chip interconnects, and simulation modeling. His teaching interests include languages and tools for introductory programming, as well as computer science curriculum design for pre-college and college engineering students.

Dr. Jeff Ringenberg, University of Michigan

Jeff Ringenberg is a lecturer at the University of Michigan's College of Engineering. His research interests include mobile learning software development, tactile programming, methods for bringing technology into the classroom, and studying the effects of social networking and collaboration on learning. He holds B.S.E., M.S.E., and Ph.D. degrees in computer engineering from the University of Michigan.

Accelerating K-12 Interest in Computer Science using Mobile Application-Based Curriculums

Abstract

Exposing students to Computer Science at an early age is critical to the continued growth and development of the Computer Science community. Once students are shown the pervasive impact of modern computing, long-term interest in Computer Science can continue to be stimulated through curriculums built around popular, age-specific topics. The advent of mobile computing through virtually every age group makes mobile application design and development an attractive topic for future K-12 Computer Science curriculums. The first part of this work details our experiences using an Android-based development platform to teach basic Computer Science principles. The second part of this work leverages that experience to propose a range of Introductory Computer Science curriculums targeted at the 9th through 12th grade age groups. Using a variety of user-experience surveys, we have found that combining mobile application development with introductory Computer Science concepts nets a great deal of positive feedback from students. As such, we provide a framework for extending this type of curriculum for future educators.

Introduction

As the information technology field continues to mature, the need for Computer Science (CS) principles to be introduced at early ages has become imperative. When students are exposed to computing at early ages, the more likely they are to have long-term interest in the subject and be willing to take the time to master advanced CS concepts such as parallel programming or object-oriented design. It is also well known that engineering students are more likely to pursue and complete CS degrees if they perform well in their freshman programming courses. Consequently, the importance of stimulating long-term CS interest at the K-12 level cannot be understated.

K-12 CS programs that dwell on the high-level benefits of a CS career can sometimes overwhelm new students. Typically, these programs will introduce students to recent research projects or high-end products in the market. While these methods inspire interest in CS, they can also be discouraging when students realize their introductory work (e.g. basic programming) is so far away from the advanced projects that were introduced.

Alternatively, educators have stimulated interest in CS through the use of K-12 friendly programming environments. Visual programming languages (VPLs) such as Scratch[1] provide a more aesthetic alternative to the traditional text-based programming environment and typically provide animation or other multimedia to reward student progress. The drawbacks of VPLs are that they can become limited when applying advanced CS concepts such as functions, recursion, or object-oriented design. As a result, students often do not see the real-world application in their assignments and projects.

The accessibility and interest of the K-12 age group in mobile computing platforms has provided educators with the opportunity to teach CS in an environment that is becoming increasingly

native to early age groups. Considering that the popularity of mobile computing devices amongst teenagers continues to grow, teaching CS through the programming of these devices promises to have a great impact.

This work seeks to combine the teaching benefits of mobile applications with the ease of K-12 ready programming interfaces. Ideally, the result will provide curriculum that will stimulate long-term CS interest in K-12 students. We first present the design of summer programs that target programming Android mobile devices using the Google App Inventor platform. While prior works have looked at the use of App Inventor for undergraduate level curricula[2], we target K-12 students and specifically a range of students ranging from 9th to 12th grade.

Our Android-based programs were piloted with 100 students at the University of Michigan, Ann Arbor and we evaluated them using a variety of student-experience surveys. We found that the majority of the students discovered the work to be both challenging and stimulating. We also found that there was a class of students that had previous experience in CS and wanted more depth in the work and/or text-based programming interfaces. Finally, this work presents a refined version of our mobile application-based curriculum for future use in K-12 education. The proposed curriculum presents an educational framework designed to facilitate lasting interest in CS across the wide-ranging needs of 9th-12th grade students.

Pilot Curriculum: Mobile App Development for K-12 Students

In the summer of 2011, one hundred 9th to 12th grade students participated in introduction to engineering camps at the University of Michigan, Ann Arbor. Students were introduced to Computer Science in one of the following programs:

- Program 1 (P1): A 1 week course for 30 underrepresented minority and women students entering the 9th grade. Students received CS instruction from 8am – 5pm each day in the form of individual labs combined with group projects.

- Program 2 (P2): A 12-day course for 10 underrepresented minority and women students entering the 10th and 11th grades. This was part of a larger engineering camp where the CS course was taught for 90 minutes each day along with Mathematics and Professional Development courses.

- Program 3 (P3): A 90-minute workshop for 60 rising 12th graders. Students interested in various engineering disciplines attended the workshop for a general introduction to CS. There were also workshops in Mechanical Engineering, Chemical Engineering, and Industrial & Operations Engineering. After all the workshops, students were split up into teams according to discipline and used their workshop skills in an engineering challenge format.

We designed our pilot Computer Science curriculum around the following concepts:

(1) Algorithms
(2) Input and Output
(3) Variables
(4) Boolean Conditions and Selection
(5) Functions
(6) Iteration
(7) Lists
(8) Software Development (Brainstorming, Specification, Planning, and Development)
(9) User Interface Design (e.g. Screen Layouts, Buttons, Textboxes, etc.)
(10) Multimedia (Sound, Video, and features specific to Android Mobile Applications)

The particular topics covered depended on factors such as the length of each program as well as the age group for the students in the program. After the topics list was selected for a particular program, a curriculum was designed consisting of lectures, tutorial-based lab assignments, and a team project (specific concepts of each pilot curriculum are available in Appendix A).

Short, picture-heavy lectures were designed in Microsoft PowerPoint and intended to last for no more than 15 minutes. Each lecture focused on a high-level concept, presented the concept using a real-world analogy and provided examples showing how that concept manifested into the target programming language (Google AppInventor). As an example, the "Variables" lecture featured pictures of backpacks, wallets, and safes to illustrate to the students that variables were storage elements. During the lectures, students were prompted by the instructor to guess the relationship between a concept and a set of images. The interactive nature of the lectures served to provide a fun and engaging environment to learn Computer Science.

Students worked on their own Windows workstation during individual lab assignments and collaborated with their peers for group projects. Lab work consisted of tutorial-like instructions to start the lab and then challenging extensions to the tutorial to complete the assignment. In the "Variables" lab, students were first given step-by-step directions on how to use variables to calculate the area of a rectangle. This included instruction on how to prompt the user for the necessary input (the base and height) values. After this was completed, students were then challenged to use variables to create applications that calculated correct results for the quadratic formula.

Group projects allowed students to be as creative as they liked to develop their own mobile application. Groups of 2 or 3 students submitted project brainstorming and project proposal worksheets to begin their projects. After instructor approval, students used the remaining lab time to work with their group to complete the mobile application. Some of the notable applications that the students created were a Coloring Book featuring their favorite carton characters, a GPS-Aware Tour-Guide application, and extensions to pre-made video games (e.g. the Wack-A-Mole game was modified to Wack-A-Celebrity featuring their favorite pop-culture figures). Appendix B contains a listing of sample project ideas for future iterations of these programs.

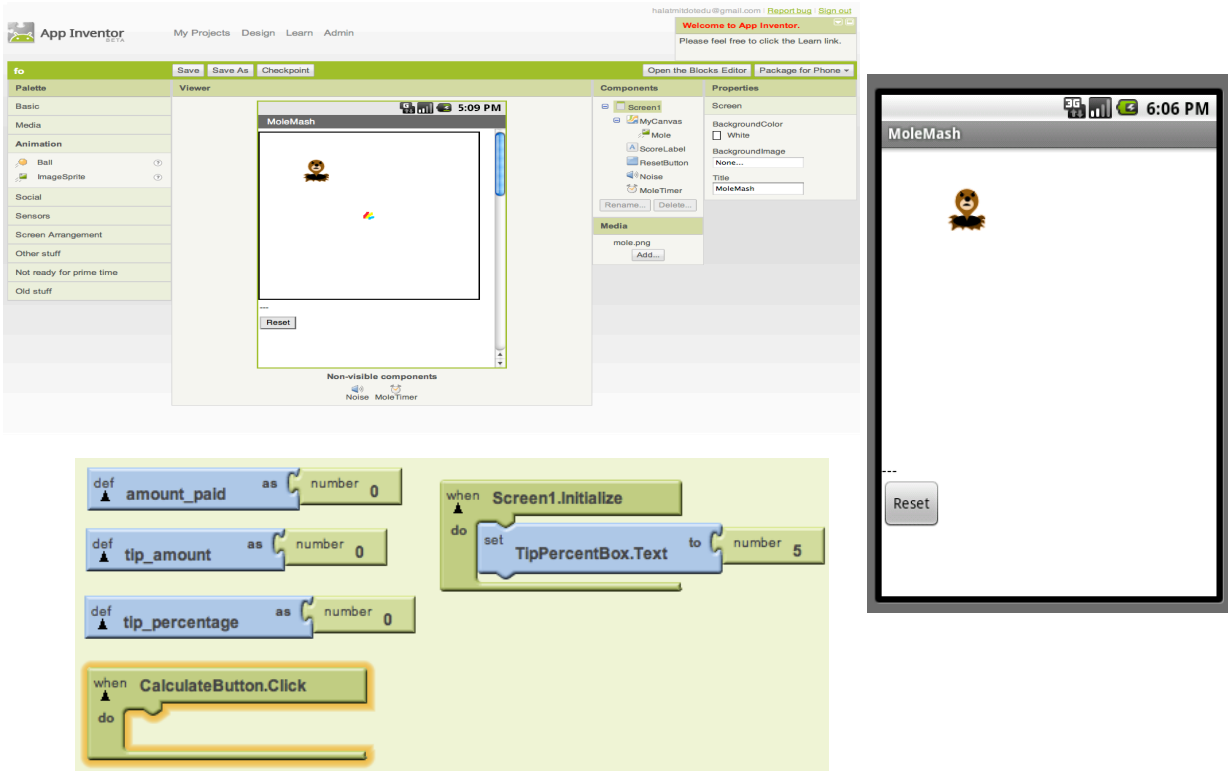

Figure 1: Design Editor, Emulator, and Block-Based Programming in App Inventor

In Programs 1 and 2, we began our studies using the puzzle programming game "LightBot" [4, 5]. LightBot introduces algorithms to students by challenging them to navigate a robot to a square on a matrix using only a finite number of steps. The fun, web-based game serves as an icebreaker to the students and helps prep them for App Inventor based studies.

All programs used Google AppInventor[3] to facilitate the programming of Android-capable mobile phones. Figure 1 shows a picture of the Android design window, the block-based programming interface and the phone emulator. AppInventor provides two features that enable students to get started quickly with mobile phone development:

(1) A drag-and-drop interface that allows students to customize the look of their mobile applications
(2) An easy, block-based programming environment that minimizes the editing mistakes that often frustrate beginning programmers.

AppInventor programs are capable of being tested using both an Android emulator which comes preloaded with AppInventor as well as actual Android-enabled devices. After testing their programs on the emulator, the students were allowed to try their applications on either a mobile phone or tablet. Five Android devices were made available to students: 2 Virgin Mobile LG Optimus V phones, 1 Coby Kyros Tablet, 1 ViewSonic G-Tablet, and 1 Entourage eDGe Tablet. The phones were additionally augmented with external memory (16 MB SD Cards) and prepaid data/voice plans.

Evaluation Methodology

The effectiveness of each program was evaluated by surveying the students after the completion of each program. Students were asked to rate the accuracy of each of the following statements on a scale of 1 – 5 (1=Strongly Disagree, 2=Disagree, 3=Neutral, 4=Agree, 5=Strongly Agree):

A. The computer science workshop was academically challenging.
B. The tasks and assignments in the workshop were interesting.
C. I learned a lot about computer science in the workshop.
D. The workshop generated excitement about technology and computer science.
E. The program made me more interested in a computer science career.
F. The program helped me improve my teamwork skills. (Only Program 1)

Additionally, the following qualitative questions were asked to each student:

1. What did you like most about the workshop?
2. What did you like least about the workshop?
3. What would you do to improve future workshops?

We were also able to gather information on whether or not students had programmed before when collecting the data for Programs 2 and 3.

Program Evaluations

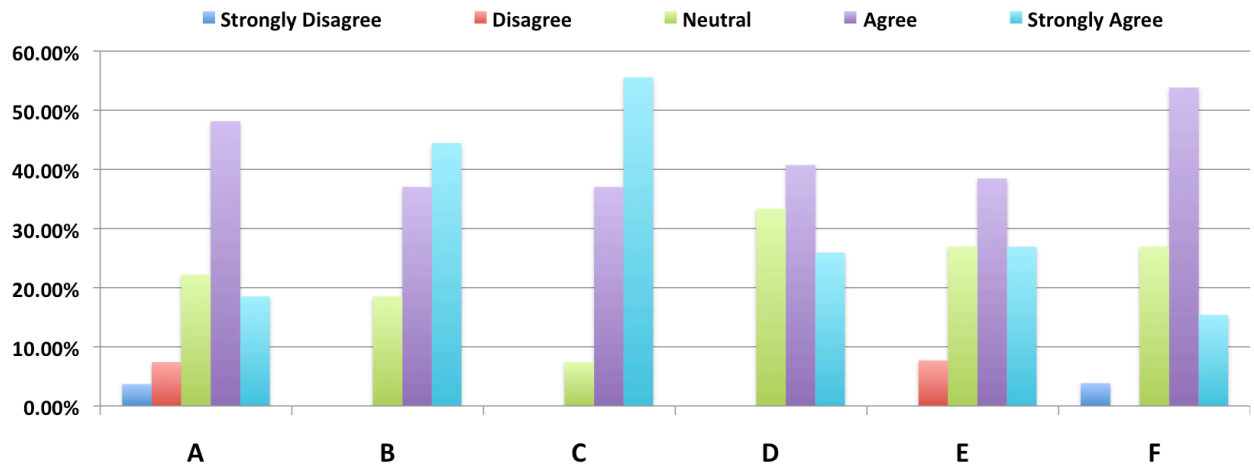

Figure 2: Program 1 Evaluation - 30 students composed of rising 9th graders

The results from Figure 2 show predominantly positive feedback from the rising 9th grade students in Program 1. An especially encouraging result was that 81% of students felt that the mobile application based tasks and assignments were interesting. Undoubtedly, teaching through a topic of interest helped produce positive results for questions D and E where ~66% of the students affirmed that using Android-based curriculums would excite students about technology and increase their interest in computer science careers.

Answers from the qualitative portion of our surveys suggested that students wanted simplified AppInventor lab instructions, more time to finish their projects, and more group activity time to socialize with their peers. Because many of the AppInventor labs were based off examples on the AppInventor website (e.g. the "Hello Purr" project), the students and instructors found that some of the lab instructions were too technically dense even though they were in tutorial form. The basic instructions seem to target a reading level for entering college freshman and some of the website examples were actually derived from introductory college curriculum[2].

While it was great practice for the students to learn attention to detail, future iterations of this workshop should strive toward creating lab assignments with more intermediate questions and checkpoints that verify the student is doing the work correctly. Additionally, instructors noticed that after about a hour of lab instruction during a 1.5 hour lab sessions, the students became academically fatigued, so another suggestion for improvement would be to target learning sessions of about 1 hour for the entering 9th grade level of student.

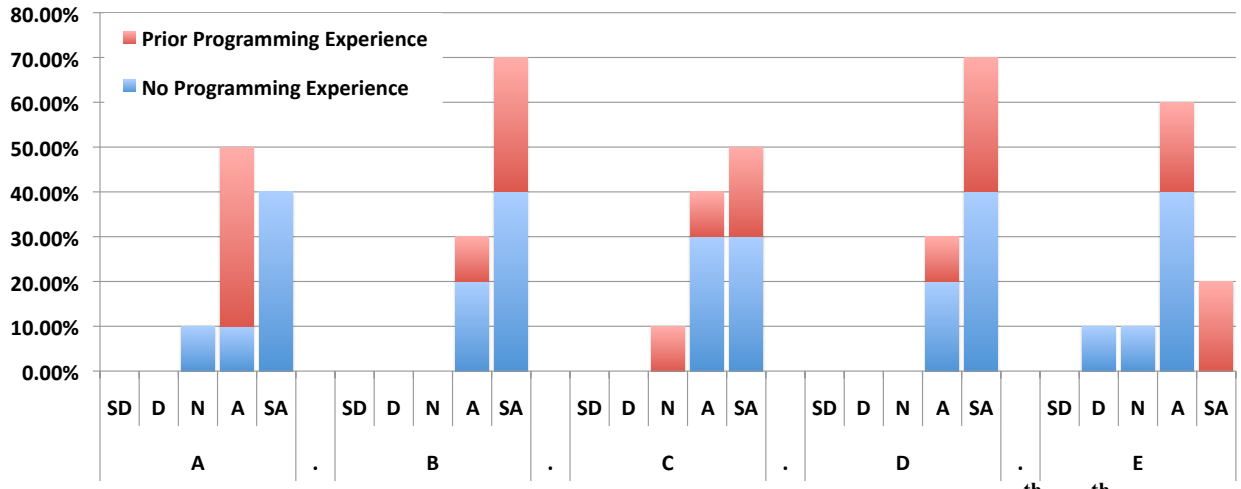

Figure 3: Program 2 Evaluation - 10 students composed of rising 10th /11th graders (6 students did not have prior programming experience)

Program 2 also produced positive results regarding the use of Mobile App Development for K-12 instruction. The results from Figure 3 showed that our 2-week Computer Science program was unanimously interesting to the students and provided all of our students with future excitement about technology.

In this group, 6 of the 10 students had not had previous programming experience. Of those 6, 4 marked that they had interest in a CS career, 1 was neutral on CS and 1 student said the program did not make them more interested in a CS career. The 4 students who had previous CS experience all declared future interest in CS careers.

The qualitative surveys from the students requested more time for finishing their final projects or more step-by-step instructions for lectures and lab assignments. This commentary supports the quantitative results that showed 9 out of 10 students felt that the work was academically challenging. Unfortunately, a sample size of 10 students is too small to make definitive judgments about 10th/11th grade teaching, although we still feel that the amount of material that

was covered in the labs and projects during the 12-day span was challenging enough that we don't suggest a significant increase in assignments for future versions of this course.

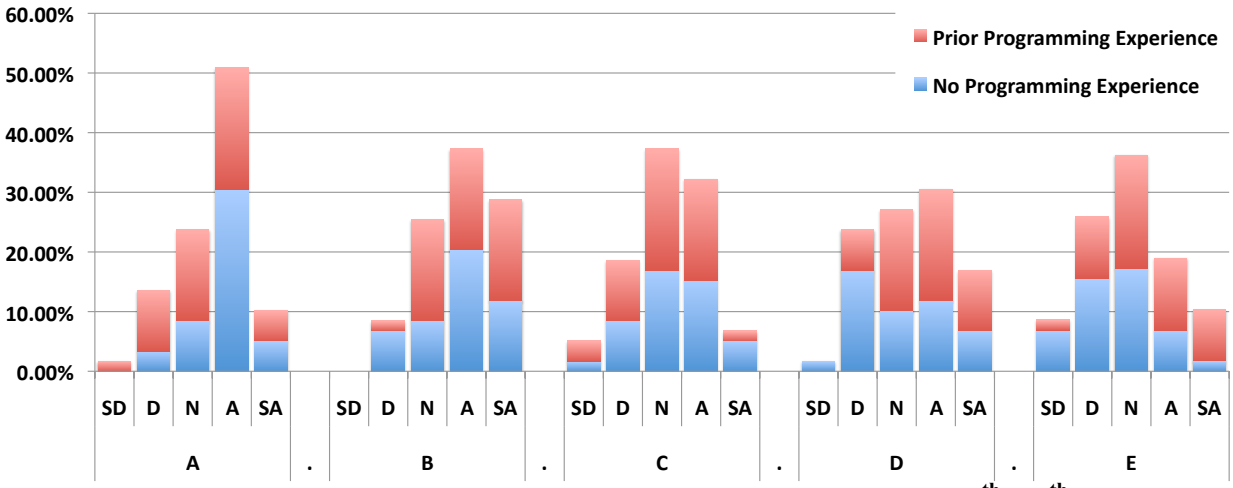

Figure 4: Program 3 Evaluation - 60 students composed of rising 11th /12th graders (28 students did not have prior programming experience)

The 90-minute format of Program 3 produced a much more diverse set of feedback from the students entering 12th grade. Figure 4 illustrates the encouraging result that 63% of the students found the topic of CS interesting. Additionally, 47% of students responded that the workshop increased their interest in technology and CS.

Unfortunately, only 29% of the students reported that our short workshop made them more interested in CS careers. The qualitative question answers lead us to believe that this result was primarily because of 3 factors: time limit, prior experience, and age group.

Because of the short format of this workshop, some of the students became discouraged when they could not setup their mobile application quickly or they ran into problems. Students from other programs had multiple days to learn the environment, but the time limit of the single-day workshop added extra pressure to those students who "saw their friends getting everything working".

Additionally, 47% of the students in Program 3 did not have prior programming experience in this workshop. From Figure 4, we see that a higher percentage of these students, as compared to the students with prior experience, were excited about the topic of Mobile Apps for CS (67%). However, a lower percentage of the students without prior CS experience left the workshop more excited about CS (39%) or interested in a CS career (18%). This result suggests to us that the topic was indeed interesting but more time was needed to allow students to make mistakes and learn.

Lastly, the students who did have prior experience programming were largely unsatisfied with the block-based programming presented in AppInventor. They felt limited by the challenges of the assignments and by the productivity of the block-based programming interface compared to a more traditional text-based programming environment. For these students, it can be argued that

either more challenging work and applications need to be presented (e.g. Android SDK development) or that these students' prior experience indicated that they are outside of the target group of students that we want to introduce to Computer Science.

Proposed K-12 Computer Science Through Mobile Application Development Curriculum

Appendix B details our proposed curriculum for engaging K-12 students in Computer Science concepts. We target the curriculum to students with little or no programming experience. Lectures should be 15-20 minutes in length and labs should be done using a block-based mobile application interface such as Google AppInventor.

Each of the 9 topics proposed consists of a lecture analogy to use for the class and a suggested lab component to teach the students. The labs are adopted from well-known examples in the proposed textbooks or commonly known programming problems. Topics 1-4 are what we termed as "core concepts". These first four concepts will be needed for students to do their group projects and should not be passed over.

Instructors can use the suggested length of each lab to generate a customized curriculum for their K-12 students. For instance, in a one-week course for rising 9th graders, an instructor may choose topics 1-4 and topic 9 to create project. Alternatively, a group of students who are grasping concepts quickly may choose to go from topic 4 to topic 7 (layout design) to project work. Since every classroom's students and needs are different, we leave it up to the instructors to choose what works best for their students.

Conclusion

Mobile devices have become pervasive in our society and extremely popular amongst the K-12 age groups. The popularity of mobile devices provides educators with a unique opportunity to teach CS within a framework that a majority of students already have access to and are keenly interested in. We theorize that leveraging mobile application development in computer science curriculum will both prepare K-12 students for advanced computer science topics and help create long-term interest in Computer Science amongst those students.

This work uses Android phones and Google App Inventor to teach CS to rising 9th through rising 12th grade students. Using a variety of survey questions, we find that mobile application development is indeed an effective tool for teaching Computer Science amongst K-12 students. More specifically, we found that over 60% of our 100 students surveyed found the topic of mobile application development interesting. In programs where students were able to focus on Computer Science for a week or more, that percentage jumps up to over 80%.

Consequently, we confirm that this is an area of extreme interest and potential to Computer Science educators. In addition, we also found that the majority of students who were exposed to mobile application development for a week or more had increased interest in technology and computer science careers. Although the populations from the programs varied in age range and length of time, we feel this result should encourage future work that seeks to determine the optimal timeframe that K-12 students should be exposed to CS curricula. These works could

gauge the CS interest levels of K-12 students when variables such as the programming environment, target applications, and length of study change.

We additionally identified that lab time and technical complexity are issues when using block-based programming labs for mobile application development. Students with little to no programming experience became frustrated when they were only given 1.5 hours to complete their labs while students with programming experience often found the interface too simplistic and/or too restrictive for their skill set.

The lessons learned from the 100 students surveys allowed us to produce a revised curriculum for K-12 Computer Science courses using mobile application development. Using this curriculum as a base, we feel that future educators will be able to engage students for long-term interest in Computer Science as well as teach them important concepts for future careers involving the field of computing.

Bibliography

1. David J. Malan and Henry H. Leitner. 2007. Scratch for budding computer scientists. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education* (SIGCSE '07). ACM, New York, NY, USA, 223-227.

2. Ellen Spertus, Mark L. Chang, Paul Gestwicki, and David Wolber. 2010. Novel approaches to CS 0 with app inventor for android. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10). ACM, New York, NY, USA, 325-326.

3. Abelson, H. 2009. App Inventor for Android. Google Research Blog, July 31, 2009. URL= http://googleresearch.blogspot.com/2009/07/app-inventor- for-android.html.

4. LightBot 1.0. http://armorgames.com/play/2205/light-bot

5. LightBot 2.0. http://armorgames.com/play/6061/light-bot-20

Appendix A: Pilot Curriculum

Program 1: 1-week course for rising 9th graders.

	Day 1	Day 2	Day 3	Day 4	Day 5
Lecture	Algorithms and Functions	Variables, Conditions, & Selection – Pt. 1	Variables, Conditions, and Selection – Pt. 2	N/A	N/A
Labs	(1) Algorithms (LightBot) (2) Intro to AppI. (3) Algorithms – AppI.	(3) Algorithms – AppI. (4) Variables, Selection, Functions Pt. 1	(5) Variables, Selection, Functions Pt. 2	(6) Team Website and Forms Creation	N/A
Projects	N/A	- Team Groupings (of 3) -Brainstorming Worksheets	- Project Proposal - Project Design	- Project Debugging - Project Submission	- Project Poster Boards - Project Demos

Program 2: 12-day course for 10th/11th graders.

	Day 1	Day 2	Day 3	Day 4	Day 5
Lecture	Algorithms	Input/Output and Variables	Conditons and Selection I	Conditions and Selection II	Functions
Labs	(1) LightBot Programming	(2) Intro to AppI. / AppInventor Math	(3) Computer Drawing	(4) Data Collection & Selection	Repeat (4)
	Day 6	**Day 7**	**Day 8**	**Day 9**	**Day 10**
Lecture	Lists and Iteration	Software Development	User Interface Design – "Virtual Screens"	Multimedia I	Multimedia II
Projects	- Project Brainstorming	- Concepts Quiz - Project Proposal	Team Projects	Team Projects	- Concepts Quiz Retake - Team Projects
	Day 11	**Day 12**			
Lecture	Team Website Design	N/A			
Projects	Team Projects	- Project Poster Boards - Team Demos			

Program 3: 90 minute workshop for rising 12th graders.

	Day 1
Lecture	- Algorithms - If-Else Conditonal Blocks
Lab	- Hello Purr! App: Students place a picture of a cat on the screen with a meow sound. If the user touches the cat picture a meow sound plays. (Students were free to use any picture or sound in place of the cat) - SuperFan Quiz Application: Picture, TextBox, Button and a Label. Students ask the user a question and use if/else blocks to determine the correct answer

Appendix B: Proposed K-12 Mobile Application Development Curriculum

Textbooks/Reference:

- "App Inventor for Android", Jason Tyler, Wiley & Sons 2011
- "App Inventor: Create Your Own Android Apps", David Wolber, Hal Abelson, Ellen Spertus, and Liz Hooney, O'Reilly 2011
- Website: http://appinventor.googlelabs.com

Syllabus:

Topic	Lecture Analogy	Suggested Lab	Lab Length (Hrs)
1. Algorithms*	What's an algorithm to buy a Mother's Day gift?	"Hello Purr!" Introductory Lab	2
2. Variables*	Relationship between a wallet, a backpack and a safe (Storage)	Computer Math: Calculate area of a rectangle. Calculate volume of a sphere. Program common formulas and plug in values (e.g. compound interest or quadratic formula)	2-4
3. Input/Output*	"Black Box" analogy using a vending machine	Basic I/O using Buttons: When the user clicks a button, change the color of the screen or play a sound	2
4. Boolean Conditions and If-Else Blocks*	Breaking down our lives into T/F decision-making (e.g. Is it time to wake up in the morning? Do I have enough money to buy a snack?)	"Super Fan" Quiz: Students pick a topic that they are a fan of and create 1-3 questions.	2-4
5. Functions	Why use a calculator instead	"Here or There?": One bank	2-4

	of manually doing a 10-digit multiplication each time?	requires a 5 year deposit, but another bank requires a 10 years. Allow a user to compare which bank should a customer deposit their money in.	
6. Iteration using While Loops	What's the algorithm to eat a bowl of spaghetti? (answer: while there is spaghetti left, keep eating food)	"Quarter Only": Use a while loop to calculate change for a vending machine that only dispenses quarters	2-3
7. Layout Design: Virtual Screens	Changing a Channel on TV (i.e. substituting one picture for another)	"Super Fan II": Students edit their "SuperFan" lab to change screens (images) once a user answers a question correctly	2-4
8. Animation	What happens when you flip a book of "stick- figure" drawings really fast?	"Wack-a-Mole" Game	4-6
9. Multimedia Extensions	Extend analogy from Functions or Input/Output Sections.	Text Messaging, Text-to-Speech, or Web Browser	4-6

*Core Concepts: Required for Project Work

Projects Ideas:

- Coloring Book
- Joke Collections
- Important Events Calendar
- "Wack-a-Mole" Game extension with new characters and levels
- Text-to-Speech Robot
- Automatic Text Message Response
- Twitter-Based Application
- GPS-Aware Applications (Advanced)