Can Android App Inventor Bring Computational Thinking to K-12?

Ralph Morelli
ralph.morelli@trincoll.edu

Trishan de Lanerolle
trishan.delanerolle@trincoll.edu

Pauline Lake
pauline.lakealmeida@trincoll.edu

Nina Limardo
nina.limardo@trincoll.edu

Elizabeth Tamotsu
betsy.tamotsu@meriden.k12.ct.us

Chinma Uche
cuche@crec.org

Computer Science Department
Trinity College
Hartford, CT, USA

ABSTRACT

App Inventor for Android is a new visual programming platform for creating mobile applications for Android-based smart phones. This paper reports on the summer component of an ongoing project aimed at addressing whether App Inventor would be a suitable platform for bringing computational thinking to K-12 students. The project brought together a team consisting of two high school CS teachers, two novice undergraduate computing students, a community outreach leader, and a college CS instructor. The students were easily able to develop complex mobile apps completely on their own initiative. Overall, the team found App Inventor to be an accessible and powerful platform that could well support introductory level courses at the college and K-12 levels.

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: Computer and Information Science Education—*Computer science education, Curriculum*

General Terms

Human factors

Keywords

Computational thinking, App Inventor for Android, K-12 curriculum development

1. INTRODUCTION

App Inventor for Android is a new visual programming platform for creating mobile applications (apps) for Android-based smart phones [7]. It was developed at Google Labs by a team led by MIT's Hal Abelson. It was released to the general public in July, after being available in alpha and beta versions to a group of invited developers, many of whom were college faculty interested in exploring App Inventor's potential as a teaching platform for CS0 and K-12 students.

To develop apps in App Inventor you do not write code. Instead you visually design the way the app looks and use blocks of interlocking components to control the app's behavior. In this respect App Inventor is comparable to Scratch (http://scratch.mit.edu) and Alice (http://www.alice.org). Like these languages, App Inventor aims to make programming enjoyable and accessible to novices. The difference, and perhaps an important reason for the attention it has gained, is that App Inventor lets you create apps for smart phones. Given the popularity and ubiquity of mobile phones among today's generation of students, App Inventor seems to hold great potential for attracting a new generation of students to computing and computational thinking.

This paper reports on a summer project aimed at addressing the following question: Would App Inventor be a suitable platform for bringing computational thinking to K-12? The project was sponsored by the Humanitarian Free and Open Source Software (HFOSS) project (http://www.hfoss.org) and funded by the National Science Foundation as part of its effort to engage K-12 teachers interested in bringing computational thinking into their classrooms. It brought together a small team consisting of two undergraduate students, two high computer science school teachers and an undergraduate computer science instructor to learn App Inventor and assess its suitability for classrooms at the CS0 and K-12 levels. The teachers were chosen from schools that serve underrepresented minorities and the team focused specifically on whether App Inventor would have the potential to engage students from this underrepresented demographic – i.e., girls, African Americans, and Hispanics – in the study of computing. In keeping with the community service goals of the HFOSS project, the team also focused on apps that could be seen as socially beneficial in some way – e.g., an app that uses the phone to teach children about nutrition – and partnered with the director a community-based technology outreach program who is working with the team to develop a pilot after school program that will involve App Inventor instruction.

The term *computational thinking* (CT) was coined by Jeannette Wing, Professor of Computer Science at Carnegie Mellon University and Assistant Director of the NSF's Computer and Information Science and Engineering Directorate

SIGCSE'11, March 9-12, 2011, Dallas, Texas, USA.

(CISE)[13], as a way to call attention to the pervasiveness of computing in all aspects of contemporary human life and knowledge. As described by the Center for Computational Thinking at Carnegie Mellon University, CT refers to a way of solving problems that involves *abstraction*, *algorithmic thinking*, and other concepts acquired from the study of computer science [5]. In her role as a director of NSF's CISE, Professor Wing as given many talks at computing conferences to promote the view that despite the ubiquitous of computational thinking in modern science, social science, and the humanities, it remains completely absent from the K-12 education. Her grand vision for the 21st Century is to see CT become as well integrated into K-12 curriculum as mathematics is in today's schools.

The recent K-12 focus by NSF is an important one for undergraduate computing education. At the undergraduate level we are increasingly confronted with the challenge of recruiting students to our introductory courses. As reported in the most recent Taulbee Survey, although undergraduate enrollments in CS were up slightly during the 2007-8 – the first increase in six years – diversity remains a problem, with only 11.8% of bachelor degrees going to women and only around one-third going to minority students [15]. The fact is that entering first year students are not giving CS close consideration in their academic plans and part of the reason for this is that K-12 students, for the most part, receive little or no exposure to CS.

The summer project focused on learning to build mobile apps that would appeal to the K-12 demographic and developing curricular ideas and lesson plans for incorporating App Inventor into the K-12 classroom. As the following sections describe, the team came to a very positive conclusion regarding App Inventor's potential as a teaching platform for CS0 and K-12 students. A number of follow-up activities are now planned for the 2010-2011 academic year, including a workshop for computer science teachers, an App Inventor project for the computing club at a local high school, an after school program in Hartford, CT for students and their families, and an undergraduate CS0 course.

2. CONTEXT AND RELATED WORK

App Inventor has been presented at a number of recent workshops and has already created considerable buzz within the CS education community.. During Fall 2009 an alpha version of the software was used in CS0-like courses to introduce non-programmers to computing at 11 colleges and universities and one high school [1]. All of the trials were considered successful, and App Inventor garnered praise from the professors and students alike. As one instructor noted, "App Inventor has the potential to transform Computer Science education. It is the best tool I've seen in terms of allowing 'non-techies' to write computer programs, and I've been teaching Computer Science for seventeen years!"[14].

Google sponsored an App Inventor presentation and demonstration at the 2010 SIGCSE symposium[2]. The presentation met with an enthusiastic response from the audience of computing educators, several of whom expressed their views that App Inventor has tremendous potential to attract non-computing students, including high school students, to computational thinking and the study of computer science.

Three-hour hands-on workshops for CS faculty were conducted at the SIGCSE Symposium in March [**?**] and at the CCSCNE conference in April [**?**]. Both of these were filled to capacity and attendees expressed enthusiasm and optimism about App Inventor's potential for appealing to introductory level students.

App Inventor was also used to engage Bay Area high school girls in an eight week after school program entitled the Technovation Challenge which ran from February 23, 2010 to April 22, 2010 [8]. The students used App Inventor to create their own mobile apps and showcased them at a concluding 'Pitch Night.' Although the Technovation Challenge was geared towards confidence building and teaching entrepreneurial skills, the positive experiences achieved with App Inventor can't but have helped enhance the students' perceptions of computing as a possible area of study.

The common theme emerging from these various efforts was that App Inventor's ability to enable novices to build apps for a cell phone is an extremely attractive development – perhaps a game changer. It allows students to focus on the interesting aspects of programming – solving interesting design problems and building creative mobile apps that are socially useful and fun to share with one's friends. Based on a review of these experiences, App Inventor, like Scratch and Alice, seemed to us to be an attractive platform for engaging students at all levels in the computing curriculum, K-12 through university.

3. WHAT WE DID

Our project explored ways of using App Inventor to introduce computational thinking to K-12 teachers and students. One student (Limardo) was a rising senior Theater and Dance major. Her previous CS experience consisted of a CS0 course that taught a little PHP scripting. The other (Lake) was a rising sophomore Computer Science and Education major who had completed a CS1 course based on Python and Java. Both teachers (Tamotsu and Uche) have had training in computer science and familiarity with a number of programming languages and both have K-12 experience in computer science and other subjects, including mathematics. Both are members of the Connecticut Chapter of the Computer Science Teachers Association (CSTA) and one (Uche) currently serves as the the CSTA chair. Morelli and de Lanerolle served as project leaders and mentors, although this was their first experience with App Inventor. And the remaining team member (Espinosa) helped the team develop follow-up plans.

3.1 The Students' Experience

Before the teachers joined the group, the students spent the first four weeks teaching themselves App Inventor. They started with the Google online tutorials and progressed quickly into writing their own applications. They kept daily logs, documenting all of their work on the project's WIKI [9]. While the project leaders helped with design concepts and timeliness, the students were really on their own in developing a number of mobile apps. When they ran into bugs or problems with the App Inventor system, they solved them almost completely on their own.

It should be noted that App Inventor was itself still under development during the project. During the first four weeks App Inventor went from an Alpha to a Beta release, and didn't go into a general release until the 7th week of the project. Periodically the App Inventor development site would come down and bugs in the development platform were fixed during the day while the students were working.

Occasionally this led to some lost code. But the students took this all in stride. If they had trouble getting something to work, they figured it out on their own, and over the course of the summer they became the in-house experts on App Inventor.

The students found App Inventor to be very accessible and quickly learned how to develop apps of their own design. They also created their own set of tutorials, which would be used to teach the teachers. Table 1 provides a summary of the applications they developed. Their first effort was based on the App Inventor *Hello Kitty* tutorial, an example that was used in the several App Inventor workshops. This app consists of an image of a cat and a button with a "Pet me!" label. When the button is pressed the cat purrs. From this start the students made a simple interactive guessing game, in which the user tries to match a pet's sound with its image. In addition to using more resources (images, sounds, labels), their app involved considerably more control logic. This app took them one day to complete.

For their second app the students developed an interactive map of the campus by overlaying clickable icons on an image of the campus map. When an icon is clicked, a dialog pops up presenting information about a building or other feature of campus. In anticipation of a visit to our lab by the Academic Dean, the students incorporated the Dean's image and one of her favorite songs, which popped up when the Dean clicked on her office building! This map included several advanced features, including (eventually) the ability to superimpose the user's location onto the map by polling the phone's GPS service. The original version (without the GPS) took about a week to build.

Their third major app was entered in Michelle Obama's Apps For Healthy Kids Contest [12]. One of the requirements of the contest was to use online health and nutrition data from the Department of Agriculture that would help kids learn about healthy eating and activity. The students' *Work it Off!* app creates a game for kids which matches various food items – e.g., a hamburger, an apple, an ice cream cone – with an equivalent about of calorie-burning exercise.

The app starts by the user naming (orally) a type of food. A built-in voice recognition app is used to match the voice input with the name of the food which is then looked up in an on-phone database. The app reports the number of calories (using the USDA data) and suggests an activity that the user can do to "work off" the calories – e.g., run a mile, swim 100 meters, watch TV for 8 hours, etc. The user can select an activity or say "Nah" and another choice will be presented. The user gets points based on their selection and the points are entered into a database on a remote server, which the user can access to see how they did compared to their friends. Some of the choices lead to humorous feedback such as "Brushing your teeth for 3 hours won't be good for your smile."

This app, which was developed in around 2 weeks, uses some very advanced features – a relational database, a client-server architecture, interfacing with other apps (the voice recognition app). The fact that such a sophisticated app could be developed in a couple of weeks by introductory-level students is clear evidence of App Inventor's accessibility and power. What's also important is the viral effect these kinds of apps have compared to, say, Alice and Scratch apps. Showing your app to your friends is as easy as pulling your phone out of your pocket.

Perhaps the best illustration of App Inventor's potential as a teaching platform was its ability to encourage students to focus on problem solving rather than coding syntax. The *Work it Off!* app used code that was adapted from one of the App Inventor tutorials. The tutorial code interfaced with back-end database code written in Python. However, the Python server did not exactly meet the needs of the students' app. One of the students, who had no prior experience with Python, was able on her own initiative to modify the server code to successfully interface with their application.

This example illustrates that programming skills learned in the context of App Inventor's visual, block-coding model, can be successfully transferred to more traditional programming languages. Moreover, this example also suggests that the App Inventor development approach – working through tutorials, studying other people's code, adapting existing code – promotes resourcefulness, creativity, confidence, and self-initiative that all educators value.

Overall, the speed with which the students learned App Inventor, together with the enthusiasm they displayed towards the programming challenges, and the obvious pleasure they took in showing their "cool apps" to friends was truly remarkable. The students' successful experience reinforces our hypothesis that App Inventor could successfully be used in K-12 and in introductory college courses to attract students to the joys of programming and motivate them to develop their problem solving skills. In addition to learning the basic programming constructs, the goal of an introductory programming experience at the CS0 or K-12 level should be to demonstrate to students that learning to program is an enjoyable and creative activity that helps improve one's ability to solve problems and meet challenges. Judging by the students' response – albeit a small sample – App Inventor would appear to be an excellent platform for achieving this goal.

3.2 The Teachers' Experience

After four weeks, when the teachers joined the project, the students switched gears and turned part of their effort to mentoring and helping the teachers learn App Inventor. The teachers started off working through the Google tutorials and the supplemental tutorials developed by the students. Both teachers had taught introductory and advanced courses in computer science at the high school level on Windows style PCs. Neither of the teachers had prior experience writing mobile applications. The team worked together to design and code simple apps that could be incorporated into a K-12 computational thinking curriculum.

After two weeks the teachers and students worked together to design and build a prototype of the *Mall Madness* app, a multi application framework designed to teach mathematics, language, and computing skills (Figure 3.2. Mall Madness allows students to add their own "App Stores" to the mall. These could be stores that "sell" apps on English or Algebra. The Mall Madness project would serve the dual role of promoting learning of computational skills along with traditional subjects. Students would build mobile apps that would not only help them develop computational thinking skills but would also help them and other students learn the standard school subjects.

The teachers view the Mall Madness project as a way to showcase many of the concepts that they like to teach in an introductory Computer Science course. It is designed to be

Table 1: App Inventor Apps.

Name	Description	Development Time	Form Elements	Logic
Animal App	Quiz game matching animal pictures to animal sounds.	1 day	Buttons, labels, media	Loops, conditionals
Trinity Tour	Virtual tour of campus.	1 week	Maps, clickable icons, sounds, graphics.	Event handling, GPS updates
Work it Off!	Interactive game to teach children the correlation between calories consumed and calories burned through activity	2 weeks	Web browser, multimedia	Database, speech recognition, client-server.
Mall Madness	Virtual mall application with stores representing various educational challenges in Math and Language	N/A	Multiplayer components, shared documents	Database, client-server.

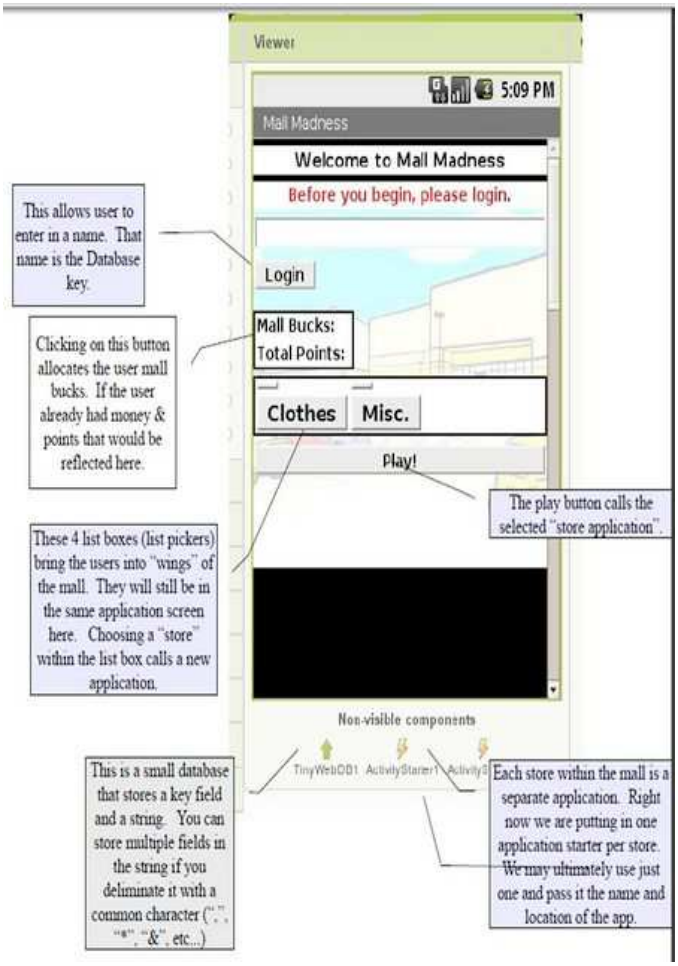

Figure 1: The Mall Madness 'App Store.'

extendible. Teachers can add code modules to exhibit additional programming concepts. Students can extend it as well. It teaches students how to work within an existing program. It enables them to create apps that help themselves and their classmates. It could be a platform that grows from school year to school year and, if distributed under a free software license, it could be shared among schools.

In addition to getting hands-on experience with App Inventor the teachers also developed a series of lesson plans based around using App Inventor to introduce students to computational thinking [**?**]. For example, Lesson 1, "Getting Started," focuses on getting the student's development environment set up, including the following:

- Obtaining a gmail account
- Registering for an App Inventor Account
- Knowing the location of the web based tutorial information from HFOSS
- Knowing the location of the web based tutorial from Google App Inventor team
- Creating list of software that they need to download to their phones.
- Completing the phone setup.

Lesson 2, "Component Design," introduces App Inventor's user interface component designer. It includes the following objectives:

- Access App Inventor's Component Designer
- Using the Pallet
- Using the Viewer
- Using the various components
- Using the properties of components
- Understanding the difference between visible and non-visible components.
- Understanding the naming conventions for components.

The teachers' assessment of App Inventor was very positive. Both teachers were pleased that it had all of the components of a real programming language and that it would allow students to create applications that could be used in the real world. The use of blocks instead of a written programming language would enable students to avoid getting bogged down in syntax problems. The learning curve was reasonable and quick. The documentation provided by Google was good. And the fact that the code is Open Source means that there is a likelihood that the documentation will continue to grow and improve over the life of the language.

At the end of their eight-week summer project, the teachers presented a demo and short tutorial to high school teachers at the CT CSTA summer symposium. The feedback received from their colleagues matched pretty well the summer teachers' own observations about App Inventor's strengths and potential as a teaching K-12 platform.

4. FUTURE PLANS

During the coming academic year, teachers will continue the experiment within their respective schools by introducing their colleagues and students to App Inventor through a variety of activities. Assuming that App Inventor continues to prove its attractiveness as a teaching platform for K-12, a follow-up NSF proposal will be developed that will incorporate App Inventor into a more comprehensive and structured multi-year program. We think such a program would have phases:

1. Working with TrInfo Cafe and with local high schools, HFOSS students and faculty together with high school faculty will use App Inventor in after-school programs to introduce high school students to computational thinking.

2. The App Inventor experienced would be followed by more in-depth computing course, using a language such as Python, in an after-school program in the spring.

3. Students would then participate in a six week summer program at Trinity College, during which they continue to build their knowledge and computing skills through projects, mini courses, and tutorials.

As students progress through these phases, they would help mentor new students who enter the program. The goal of the project would be to get participating students – recruited from underrepresented demographic groups – to apply to college – possibly after taking the AP exam in computer science – or, in some other measurable way, demonstrate an interest in the further study of computing.

As a step toward preparing for such a long-term program, our immediate follow-up plans include the following activities:

- The TrInfo Cafe, Trinity College's community technology center, will pilot a multi-week App Inventor immersion program for six to eight high school students throughout the fall 2010 semester. The goal is to expose students to computational thinking through an enrichment program targeting their interests. Students will use App Inventor to develop their own Android phone applications. The program will use the lesson plans developed during the summer. At the end of the program an evaluation of the effectiveness of the lessons and the overall program itself will be produced.

- During the Fall semester students in the computing club at a local high school will work through and help refine the lesson plans developed during the summer. This will be a voluntary after school activity and will be a second way (in addition to TrInfo Cafe) to get feedback from K-12 students.

- In November, the Greater Hartford Academy of Math and Science (GHAMAS) will devote one of its annual professional development days for a workshop entitled "App Inventor: a visual programming tool for Android phones." GHAMAS staff will lead teachers from around 35 district schools in a hands-on exploration of App Inventor as a teaching tool. The participants will complete an evaluation of App inventor as a tool for promoting student engagement, teaching problem solving, teaching programming in high schools, and raising students' levels of achievement. Furthermore, App inventor will be discussed as a possible replacement for the programming unit in the Exploring Computer Science (ECS) curriculum, a UCLA initiative aimed at increasing access to college preparatory high school CS instruction [6].

- In Spring 2011 an undergraduate CS0 course, "Computing with Mobile Phones," will be offered at Trinity College. The course will use many of the tutorial materials developed during the summer and will incorporate the study of App Inventor into study of traditional introductory CS topics.

5. CONCLUDING OBSERVATIONS

This paper described a summer project that is part of an ongoing effort to evaluate the feasibility of using App Inventor as an means of promoting computational thinking among K-12 students.

It would be premature to draw many strong conclusions from such a limited experiment among a small number of students and teachers. However, the following observations seem reasonably well supported by the combined experiences of the students, teachers, CS faculty, and resource people involved in the 10-week summer experience.

- **Accessible and Powerful**. Teachers and students alike were aware that App Inventor had been touted as powerful and easy to use, but the ease with which sophisticated mobile Apps could be written came as a surprise. It was not only easy to follow and reproduce already written apps, it was also straight forward to develop completely new apps based on the principles acquired through the tutorials and demonstrations. Students progressed quickly from writing "Hello Kitty" to developing apps using database, interactive maps, client server communication, and other advanced concepts. And they appeared to understand the concepts they had acquired.

- **Object Oriented Programming Model** App Inventor provides an effective object-oriented framework for designing visual interfaces and event-driven control structures. It presents application development as

two main activities – analyzing what components (objects) make up the app and designing and coding the actions (behavior) that these objects take during the app's execution. This model provides a good introduction to the object-oriented paradigm and provides a good foundation for acquiring more advanced computing skills in Python and other languages,

- **Problem-driven learning**. Programming with App Inventor helps novice students focus more on problem solving and less on language syntax. As CS educators well know, problem solving is infectious. You could see the pleasure that the students took in in finding out how to do things and fixing their own bugs. And you could see them gaining in confidence and taking on larger problems, including problems with the alpha version of the development environment itself. Given App Inventor's focus on problem solving rather than syntax, students were able to transfer their programming skills to new types of problems – databases, client-server communication, GPS – and new languages (Python). It would be premature to attribute too much of the success we observed to App Inventor – the students themselves were self-starters and good problem solvers – but App Inventor's potential for reinforcing such thinking and instilling such confidence was apparent.

- **Motivational Potential**. The presentation to the CT-CSTA was received positively. In discussing the Grade 9 curriculum, CT-CSTA members agreed that App Inventor, Web Design, and Robotics are all tools that can be used to keep the magic in computing education. However, the portable nature of App Inventor apps and their usefulness, unlike Web Design and Robotics, were highly regarded by the teachers. As one member observed "App Inventor would likely be a stronger motivator than Alice or Scratch because a phone is much more "hands-on" and real than the fantasy worlds of Alice or Scratch."

- **Relevance**. The relevance of programming a phone, something important in students' lives, should also work well as a motivator. App Inventor's built-in emulator means that only a few actual phones would be needed to introduce App Inventor at the K-12 level. But as smart phones become less expensive, it will certainly become feasible to acquire them to support such instruction. This view was shared by other teachers who attended the CT-CSTA workshop. There was a general consensus among attendees that App Inventor has enormous potential for the future for high schools.

- **Support for learning**. The resources accompanying App Inventor, due to its association with Google, provide unquestionable advantages for its use by both teachers and students. In particular, the App Inventor Google group provides enormous support for all.

6. ACKNOWLEDGMENTS

This Humanitarian FOSS Project is supported in part by the National Science Foundation under grant #0930934. We would like to acknowledge Hal Abelson (MIT) and Franklyn Turbak (Wellesley College) for their encouragement and support and for providing us access to the pre-release versions of App Inventor.

7. ADDITIONAL AUTHORS

Carlos Espinosa, carlos.espinosa@trincoll.edu

8. REFERENCES

[1] H. Abelson. App Inventor for Android. *Google Research Blog*, http://googleresearch.blogspot.com/2009/07/app-inventor-for-android.html, July 31, 2009.

[2] H. Abelson and M. Friedman. App Inventor – A view into learning about computers through building mobile applications. *Proceedings of the 2010 SIGCSE Symposium*, http://www.sigcse.org/sigcse2010/attendees/supporter_sessions.php, March 2010.

[3] H. Abselson, M. Chang, E. Mustafaraj, F. Turbak. Mobile phone apps in CS0 using App Inventor for Android. *Proceedings of the 2010 SIGCSE Symposium*, March 2010.

[4] H. Abselson, M. Chang, E. Mustafaraj, F. Turbak. Mobile phone apps in CS0 using App Inventor for Android. *Proceedings of the 15th CCSCNE Conference*, March 2010.

[5] Center for Computational Thinking. Carnegie Mellon University. http://www.cs.cmu.edu/ CompThink/, 2010.

[6] J. Goode. Exploring Computer Science: A University/K12 Partnership to Increase Access to Engaging and College-Preparatory High School Computer Science http://www.csta.acm.org/ProfessionalDevelopment/sub/CSIT10Pres August 2010.

[7] Google. App Inventor for Android http://appinventor.googlelabs.com/about/, 2010.

[8] Google. Technovation Challenge 2010. http://sites.google.com/site/technovationchallenge2010/, March 2010.

[9] N. Limardo and P. Lake. App Inventor Notes. http://notes.hfoss.org/index.php/AppInventor, August 2010.

[10] N. Limardo and P. Lake. Work it off. http://wio.hfoss.org/, August 2010.

[11] B. Tamotsu Lesson Plans. http://notes.hfoss.org/index.php/AppInventor:Teachers

[12] U.S. Department of Agriculture. Apps for Healthy Kids. http://www.appsforhealthykids.com/, August 2010.

[13] J. Wing. Computational Thinking. *CACM*, 49(3):33–35, March 2006.

[14] D. Wolber. Teaching App Inventor. *App Inventor for Android Blog*, http://www.appinventor.org/teaching-android, no date.

[15] S. Zweben. 2007-2008 Taulbee Survey. *Computing Research News*, http://www.cra.org/resources/taulbee/, May 2009.