Using App Inventor in a K-12 Summer Camp

Amber Wagner, Jeff Gray, Jonathan Corley
Department of Computer Science
University of Alabama
{ankrug, jggray, corle001}@bama.ua.edu

David Wolber
Department of Computer Science
University of San Francisco
wolber@usfca.edu

ABSTRACT
Educators are often seeking new ways to motivate or inspire students to learn. Our past efforts in K-12 outreach included robotics and media computation as the contexts for teaching Computer Science (CS). With the deep interest in mobile technologies among teenagers, our recent outreach has focused on using smartphones as a new context. This paper is an experience report describing our approach and observations from teaching a summer camp for high school students using App Inventor (AI). The paper describes two separate methods (one using a visual block language, and another using Java) that were taught to high school students as a way to create Android applications. We observed that initiating the instruction with the block language, and then showing the direct mapping to an equivalent Java version, assisted students in understanding app development in Java. Our evaluation of the camp includes observations of student work and artifact assessment of student projects. Although the assessment suggests the camp was successful in several areas, we present numerous lessons learned based on our own reflection on the camp content and instruction.

Categories and Subject Descriptors
D.2.6. [**Software Engineering**]: Programming Environments

Keywords
App Inventor, Java Bridge, Summer Camp

1. INTRODUCTION
Students learn best when the learning objective is contextualized with topics related to their daily activities [14]. Most teenagers are frequent users of smartphones and texting, which provides a unique context for engagement. The adoption rate of mobile computing among teenage students in the US continues to grow and has been estimated at 58% for smartphone adoption [11]. Based on current US census results [17], this suggests that there are 12.1M teenage students with smartphones. Educators can take advantage of these devices as a springboard for motivating topics involving CS [1], which has always been driven by a love of tinkering and creative exploration using computational themes and artifacts.

SIGCSE'13, March 6-9, 2013, Denver, Colorado, USA.

Since 2004, we have offered multiple one-week summer camps to K-12 students. As a context for motivation and learning, we have used Lego Mindstorms robots [10], Alice [2], and Media Computation [9] in our past camps. In Summer 2011 and 2012, we offered three one-week camps focusing on Java [1] across multiple themes. The topics and attendance at these camps is summarized in Table 1. Students had the opportunity to attend all three weeks or one week. A dormitory option allowed us to offer the camps to students across the United States (the 2012 camps were attended by high school students from 12 states) in addition to students from Hong Kong and Beijing.

The third week was focused on App Inventor (AI) [3], which enabled us to take advantage of the interest that high school students have in smartphones. AI is a visual programming language that allows users to write apps using a block-oriented drag-and-drop interface to create both the user interface of an app, as well as to specify the app's behavior and functionality. An emulator is available for AI so that apps can be executed on a local desktop. AI also integrates with Android smartphones and tablets, which enables the user-made applications to be tested on a physical device. There is also an alternative to using the block language of AI: the Java Bridge, which allows for the creation of Java programs that use the same components available in the AI block language. From our experience in teaching Android programming to college seniors using the traditional Android SDK with Java, we found the Java Bridge to be a much simpler way to program Android apps and an excellent transition tool for students learning Java.

Table 1. Summary of three camps taught in Summer 2011/2012

Camp	Pre-requisites	Content	Attendance
Week 1: Intro to Java	No experience needed; Grades 9-12	Taught intro to Java using Greenfoot and Media Comp	2011: 28 2012: 33
Week 2: Robotics	Existing knowledge of Java or attendance of the first camp	Taught Java using Lego NXT and Lejos	2011: 12 2012: 25
Week 3: Android App Inventor	Existing knowledge of Java or attendance of the first camp	App Inventor Block Language and App Inventor Java Bridge	2011: 16 2012: 24

There are other mobile platforms that support educational outreach [8]. However, because our camps were for students new to programming, the visual language of AI was found to be a positive way to introduce students to problem solving and computational issues. We observed that after students became

[1] http://outreach.cs.ua.edu/camps/

familiar with the components and events of the block language of AI, they were then more prepared to learn and use the Java Bridge to rewrite the same programs in Java. In Section 3.2, we present our approach for easing the students into the Java Bridge to improve their practice in using Java. The first two days of the camp were taught using the block language of AI, the next two days were focused on using Java, and the final day was focused on student projects (this gave students the ability to be creative and write a program that they wanted to use in their daily lives). In Section 4, we summarize our thoughts on why we believe that teaching the AI block language helped students to be more successful in programming apps using the Java Bridge.

This paper presents related instruction at other institutions (Section 2), the layout and content of the camp (Section 3), observations of student reactions, strengths and weaknesses of the camp (Section 4), and how the summer camp will evolve in future offerings (Section 5).

2. RELATED WORK

The popularity of mobile devices has inspired much interest as a context for teaching computation [13]. In fact, a new learning model has emerged, LOCAL (Location and Context Aware Learning) [4], which combines mobile devices and wireless networks to create a new learning context. In this section, we summarize a portion of the work that has emerged in using smartphones as a context for teaching CS.

At the University of San Francisco (USF), co-author Wolber has integrated AI into a general education course over the past three years and reported that AI was "his most satisfying teaching experience in seventeen years" [19]. Before teaching AI, this course used Lego Robots [10] and Media Computation [9]. The course focuses both on programming and the real-world impact of mobile applications [19]. Because students were able to start building applications immediately with AI, they were motivated as the semester progressed to learn how to solve hard logic problems. Although the students attending the course were not CS majors, they successfully learned how to solve problems and, more importantly, felt empowered at the conclusion of the course (they even presented along with Senior and Master's project students at USF's annual CS Night). There were 11 students, out of 41 total, who went on to take the next course in the CS sequence. In the past, this CS0->CS1 bridge has been very rare, because the students that enroll in CS0 often take it because they have little confidence in mathematics. A video describing student feedback on this course is available[2].

As described in [7], Fenwick and Kurtz (from Appalachian State University) and Hollingsworth (from Elon University), taught their senior classes using the Android SDK [6] and AI. The Elon University course started as a lecture-based course with hands-on activities followed by a project-based approach. The Appalachian State University course was project-based from the start. Both universities found that "students enjoyed the course" and students were once again exhibiting an "entrepreneurial and independent spirit" [7].

To assist in teaching the importance of human-computer interaction, Loveland used AI to motivate students [12]. At the conclusion of the course, one student commented: "It is cool that the course applied the latest technologies in software development to mobile and web design." This statement summarizes other reports [4, 7, 8, 13, and 19] about why mobile computing is being used as a teaching context. Students are able to learn about programming using tools that are very personal and applicable to their daily lives.

A few college courses on mobile computing have been described above; however, these courses focused on college-level students in beginning or non-major programming courses. Roy [15], from Valdosta State University, organized summer camps for high school students using AI in 2011 and found AI to be an effective tool for novice programmers due to the visual environment being similar to Scratch. Moreover, Roy noted that transitioning from AI to Java would be easy due to the Java Bridge functionality, but does not provide an assessed study.

The primary difference between these examples and our summer camp is our approach of presenting a visual language followed by a strategic transition to Java. The gap among first learners using graphical languages and textual languages was also noticed by Cheung et al. [5], who describe an approach for easing into textual languages from a graphical language. The new version of Alice 3 also enables such a transition. However, the benefits of using a common set of examples to support the transition from a graphical to textual programming language is not described deeply in the literature.

3. FROM BLOCK LANGUAGE TO JAVA

AI was the focus of our third and final week of camp, meeting for a total of 35 hours over a 5-day period. Among the 40 students across both summers that attended the AI week of camp, 14 had Java experience from their high school and 26 were introduced to Java in our first week of camp. This section introduces our approach for providing a transition path from a visual language to Java for writing apps.

Given the varied Java experiences of the students, the camp started with two days of using the AI block language, which allows novice programmers to build Android apps quickly (using a blocks metaphor, such as in Scratch [16]). After the students gained confidence with events, components, and the general AI environment, students were then introduced to the Java Bridge. The learning objectives for the camp were to teach the students the following concepts:

> 1. Writing Android apps; 2. Objects; 3. Programming environments; 4. Events; 5. Decision statements; 6. Loops; 7. Method calls; 8. Method creation; 9. Being able to interpret documentation; 10. Creating a GUI and coding the components of the GUI; 11. Using phone sensor components and services (camera, text messaging)

3.1 Starting with the Visual Block Language

The AI environment, components and visual language were covered in days one and two by using examples from co-author Wolber's textbook [18]. Although the students already had some previous experience with Java (either from our week 1, or from an offering at their home school), it was important to introduce the students to AI via the block language due to the complexity of the various components, events, GUI, and general metaphors associated with programming Android apps. The block language helped familiarize the students with these components before jumping into the Java Bridge. The apps we selected from the textbook represent a progressively complex combination of applications that offer benefits to society (e.g., "No texting while driving") as well as entertaining apps such as games (e.g., "MoleMash"). Rather than a traditional lecture, the camp was approached with an inquiry-based hands-on approach, where a

[2] http://www.appinventor.org/

problem was presented and the instructor walked the students through the problem while answering questions as the instructor and students completed the app together. Next, a similar problem was presented that covered one or more new concepts. The students were presented with each exercise and were given approximately thirty minutes to implement each app. For those who implemented an app quickly, an additional challenge was assigned. After the allotted time, students were asked to volunteer to present their solution to the class. To emphasize that more than one solution/approach exists, multiple students were asked to present their solutions, and the instructor aided the students in explaining options for various possibilities.

On Day One, students were introduced to the AI interface and began creating applications immediately. The apps the students focused on throughout the day are from [18]:

1. "HelloPurr" (see Figure 1; notice that only three lines of "code" were required to create this app) – this application is a simple button with an image of a cat. When the button is clicked, a "meow" sound is played, and the phone vibrates for 500ms simulating a purr.
2. "PaintPot" – a paintbrush app that has options for changing pen color, wiping the canvas, drawing circles/lines, and changing the background image through a call to the camera component (see Figure 3 for an emulator image).
3. "MoleMash" – this application simulates the "Mole Mash" game at arcades where a "mole" appears in a random hole, and the player must hit the "mole" before it disappears.

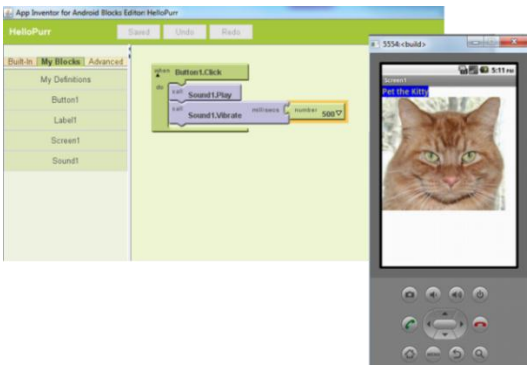

Figure 1. Block editor and emulator (left to right)

Day Two began by asking the students to explore on their own several projects from the textbook to continue their familiarity and confidence in using the visual block language. We then introduced the week-long project and asked the students to design and create their own application to present to the class on Day Five. Students were told they could use either the visual block language or the Java Bridge (described in Section 3.2) that would be introduced on day three. Each student gave a 5-minute presentation to the class of their project idea. Please refer to Figure 2 for an example of the timeline for the 2012 camp.

3.2 Mapping to the Java Bridge

As noted in Section 2, many universities are beginning to offer a mobile computing course focusing on app development. The University of Alabama offers a college senior-level design course focused on Android development that is taught using Java and the Android SDK. Although this is a feasible option at

ID	Task Name	Start	Finish	Aug 2012				
				30	31	1	2	3
1	Introduction to AI: Hello Purr, Paint Pot, and Mole Mash	7/30/2012	7/30/2012					
2	More work on projects from the text [14]	7/31/2012	7/31/2012					
3	Begin final project ideas and proposals	7/31/2012	7/31/2012					
4	Introduction to Java via the Java Bridge: repeat Paint Pot	8/1/2012	8/2/2012					
5	Time to work on final projects	8/2/2012	8/3/2012					
6	Final project presentations	8/3/2012	8/3/2012					

Figure 2. Timing of topics presented in 2012 camps.

the college level, we felt it would be overwhelming for high school students (with limited Java experience) to learn the core Android SDK. Thus, our approach transitioned the students from the visual language of AI into the Java Bridge, which assists in writing a Java-based app that targets the same AI components (the Java Bridge is a .jar file containing all of the AI components that is included in the build path of a Java app in Eclipse). The Java Bridge was first introduced on Day Three of the camp. Our approach was to take the same apps that were developed in days one and two, and show the equivalent implementation in Java. Our reasoning for this strategy is that the students would be able to form a mental mapping to the equivalent Java representation. In fact, the code completion of the Java Bridge components in the Eclipse editor reveals that the component interfaces provided in Java are the exact same as those components in the visual language of AI. We believe that the familiarity with the same components from the first two days of the camp is an asset to students learning to use the Java Bridge.

3.2.1 PaintPot with the Java Bridge

The first exercise the students completed using the Java Bridge was the HelloPurr app. This provided the details on the structure of a Java Bridge application. From that experience, the students were then asked to implement the PaintPot app in Java. We demonstrated how to create the initial setup with the background image and the creation of the first button to select the pen color, including both the GUI and the event method code for the button. Students were asked to create the two remaining pen buttons and a "Clear" button. Those who completed the problem quickly were challenged to add a button, which would take a picture and replace the background image with the newly taken picture. Six students completed the app within nine minutes, and all students completed the initial problem within thirty minutes. Three students successfully implemented the camera challenge without help from the instructors within that same time period. We found this to be a remarkable accomplishment – we estimate that for ourselves to create a similar app using the traditional Android SDK would take approximately six hours. The fact that students with little Java experience were able to do this within 30 minutes attests to the power that the AI Java Bridge offers as a teaching environment, which abstracts many of the accidental complexities found in the Android SDK.

Figure 3 illustrates a portion of a completed implementation of the PaintPot app in Java. As can be seen, in the Java Bridge the "main" method is replaced with `$define`. The code that is shown in Figure 4 is focused on the creation of the GUI. Java components are created and added to the default `Form` object (i.e., the `this` object). Other parts not shown in this figure include the import section, the registration of events for dispatching, and the implementation of event handlers.

The complete implementation of the PaintPot, as well as the other apps given to students, is available at our Google code sites at both the University of San Francisco[3] and the University of Alabama[4].

3.2.2 Skeleton of a Java Bridge App

The skeleton of a Java Bridge app is shown in Figure 4. The required packages from the Java Bridge .jar file include classes for handling the event dispatching, as well as the importation of all the GUI components used in the app. The class representing the app must extend the `Form` class within the Android SDK and implement the interface specified by the event dispatcher. The main class also declares all of the GUI components that will be used, which are then constructed at the beginning of the `$define` method and inserted into the form. At the end of the `$define` method, the events that the app is interested in are registered (sample events include things like Click, Dragged, Touched). A separate dispatching method is then called whenever an event occurs in the app. Those events that are filtered by the dispatcher have their appropriate handlers called, which represent the remaining methods in the class.

We have observed that students very quickly pick up on the mechanism for registering events in a manner that is much more intuitive to them than the listener approach required in the core Android SDK. An important aspect of AI is its event-based nature; i.e., an app is a set of event-handlers. The only challenge that students faced was uncovering the specific names of the handlers, which are available in documentation.

At USF, co-author Wolber explored a different way for Java Bridge programmers to specify event handlers that is a bit higher level than registering them. In this approach, a slightly modified Java Bridge is used, one where the components are not final classes. The programmer subclasses the components (e.g., Canvas), then can override the event methods (e.g., Dragged) to build the event-handler. This simplifies things for beginners because they do not have to worry about the low-level event names and event registration. The event-handler method headings are already in the superclass and the programmer need only override them to program the handler (Eclipse even generates the override method header). For example, consider the following code:

```
class GameCanvas extends Canvas {

  @Override
  public void Touched(float x, float y, boolean touchedSprite) {
```

The downside with this approach is that inheritance is used, but in limited testing novice students have grasped the approach readily and with less problems than the event registration scheme of the traditional Java Bridge.

As an extension of the Java Bridge idea, students at USF have created an application to generate the Java code automatically for an app created within AI[5]. This assists students transitioning from AI to Java from a different perspective. The students can create their own app and then see the Java code required to build the app.

[3] https://sites.google.com/site/mobileprogrammingusf/paintpot-java-bridge

[4] http://code.google.com/p/appinventor-java-translation/downloads/list

[5] http://usfaicg.appspot.com

3.3 Final Project

In addition to the exercises that we used from [18], the students were asked to work on their final project that was initially introduced on Day Two. The primary time for project design and implementation occurred on Days Four and Five. The student projects were often collaborative (done in teams) and very creative. The projects varied from applications that would benefit the community to games and entertainment apps. The final projects serve as the artifacts that are assessed in Section 4.2. These apps are publicly available on our camp website.

One beneficial app created using the Java Bridge assists environmental scientists in collecting information about samples from photos taken of the environment. This app allows a citizen scientist to help collect information by taking a photo of some reactive test and then report details about the image by touching various parts of the image. This project actually turned into a regional science fair project for the student, with continued mentoring by one of the co-authors. The app will return various properties about the pixel that is touched. Another final project, created using Java, acts as a study aid for science. The application asks the user for initial and final data for one or more of the following fields: pressure, temp, moles, molecular mass, and/or density. Based on the information the user enters, the app fills in the remaining information according to laws of gases: Boyle's Law, Charles's Law, Gay-Lussac's Law, Combined Gas Law, and Ideal Gas Law (true and false). This application required the use of data structures, loops, decisions, custom methods, events, objects, and components. Although this student had previous Java experience, this was her first time using Java to program Android phones. Based on the complexity of the project, she mastered the material thoroughly and quickly.

4. EVALUATION AND DISCUSSION

This section introduces two assessment approaches to determine the impact of using AI in our summer camps. In Section 4.1, we summarize our observations made throughout the week; Section 4.2 provides some insight from an artifacts assessment that we performed on the publicly available final student projects.

4.1 Observations from the AI camp

Of the three different camps, students were noticeably most excited for the start of the AI camp. During the PaintPot exercise, students were not initially impressed with the difficulty-level of the application; however, as soon as they completed the app and ran it on their phones, smiles began appearing throughout the classroom. The students enjoyed working with AI and learned how to use the programming environment within a day of its introduction.

When transitioning to the Java Bridge, the students were challenged, but because we repeated the same applications (HelloPurr and PaintPot), they were able to create a mental map between the Java syntax and what they previously created with the blocks language. The majority of the students appeared to understand how to use Java in creating an app. Some of the students preferred using the blocks language to using the Java Bridge, due to the ease of creating GUIs in the blocks language (as compared to programmatic creation of the GUI in Java).

Overall, the students were more engaged this week than in previous weeks. Because students were so enthusiastic about creating their own apps, which they could install on their phone, they wanted to arrive early to the camp and stay later. We did not see this level of engagement in previous weeks.

```
private int count = 0;
private int dotSize = 30;

void $define()
{
    img = new BufferedImage(this);
    img.Visible(false);

    cam = new Camera(this);

    cnv = new Canvas(this);
    cnv.BackgroundImage("resize.jpg");

    Label space = new Label(this);

    btnTakePic = new Button(this);
    btnTakePic.Text("Take Picture");

    Label space1 = new Label(this);

    HorizontalArrangement hrz1 = new HorizontalArrangement(this);
    HorizontalArrangement hrz2 = new HorizontalArrangement(this);
    HorizontalArrangement hrz3 = new HorizontalArrangement(this);
    HorizontalArrangement hrz4 = new HorizontalArrangement(this);
    Label space2 = new Label(this);
    HorizontalArrangement hrz5 = new HorizontalArrangement(this);

    lblX = new Label(hrz1);
    lblX.Text("X: ");
    lblXPosition = new Label(hrz1);
    lblXPosition.Text("");
```

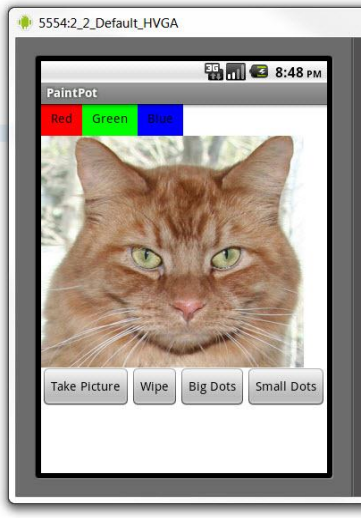

Figure 3. PaintPot application written in Java

```
import com.google.devtools.simple.runtime.components.HandlesEventDispatching;
import com.google.devtools.simple.runtime.events.EventDispatcher;
import com.google.devtools.simple.runtime.components.android.*;

public class MyApp extends Form implements HandlesEventDispatching
{
        /* Declaration of GUI components goes here */

        void $define()
        {
            /* GUI components constructed and added to Form, similar to code in Figure 3 */

            EventDispatcher.registerEventForDelegation( this, "MyApp", "SomeEvent" );
        }

        public void dispatchEvent(Object component, String id, String eventName, Object[] args )
        {
                if( eventName.equals( "SomeEvent" ) )
                        if( component.equals( someComponent ))
                                someComponent_DoAction();

                /* Other event dispatching goes here */
        }

        private void someComponent_DoAction()
        {
            /* Some specific event handler code here */
        }
}
```

Figure 4. Skeleton of a Java Bridge app

4.2 Student Project Assessment

To evaluate what the students learned throughout the week, we reviewed the 23 (9 from 2011 and 14 from 2012) complete and submitted final projects (students gave permission to post online) based on the eleven learning objectives presented in Section 3.1 (several of the projects were team-based). Some students did not submit a project because they were unable to complete it, which indicates either the project was too ambitious or we should start the project time earlier in the week. Table 2 lists the given learning objectives and the number of projects where each learning objective was met (the learning objective had to be used correctly to be counted). The projects used in this assessment were student selected projects; therefore, the difficulty level is indicative of the student's understanding. The results are positive considering only 35% of the students had exposure to Java prior to this summer (i.e., 65% of the students learned Java just from our first week of camp).

It is clear from Table 2 that iteration was not a common need in the final projects (only 3 projects used a looping construct), but many students had no need for loops in their applications because the event processing did not require iteration. 15 projects included decision statements, which is also a critical

construct used for problem solving; therefore, most students felt confident using decision statements in their applications. The majority of students also felt comfortable with the use of objects, learning a new environment, working in teams collaboratively, using and programming events, calling methods, reading documentation, creating GUIs, and using components based on project assessment and observations.

Table 2. Number of projects meeting the learning objectives

Learning Objective	Number of projects
How Android apps can be written	23
Objects	23
Programming environments (Block Language and Java)	23
Events	23
Decision statements	15
Loops	3
Method calls	23
Method creation	7
Understanding how to read documentation	14
Creating a GUI	23
Using components	23

5. CONCLUSION

In order to inspire students to study CS and understand its relevancy to their lives, educators should identify meaningful learning contexts. From our experience, we found that mobile computing provides a powerful new context for motivating computational thinking. The 40 students attending our AI week of camp across two years were exposed to both the AI block language and the Java Bridge to familiarize them with multiple environments and solutions, GUI development, and Java.

It was clear that students needed solidification of Java concepts at the beginning of the week. Based on information in Table 2, we feel the approach of teaching a visual language followed by Java (using common examples in each) allowed students to gain the confidence necessary to build applications, while reinforcing core concepts such as objects, decision statements, and methods. In a separate week, we also used AI in a teacher-focused workshop (supported by a Google CS4HS grant) where science teachers were taught how to create apps focused on their classroom needs. Several of these teachers are introducing AI into new CS Principles classes to highlight the way in which computing affects many disciplines[6]. At the University of Alabama, we are also Piloting a CS Principles course with the College Board that uses AI as a core focus.

Moving forward, this camp will be offered in Summer 2013 with improvements made based on student feedback. Our observations after reflecting on the past two years has led us to several activities that we plan to try, including: 1) introduce the Java Bridge earlier (afternoon of Day Two); 2) homework challenges with notes for those who need more help or more challenges; and 3) for those students staying in our dormitory, offer evening sessions for those interested, providing additional help to those who are behind or deeper challenges for advanced students. As a contribution to the AI community, we also developed a prototype tool that will convert AI programs in the block language into a complete Eclipse project representing the same app as translated to Java[7]. This allows an integrated mapping mechanism where students can compare their block language programs to the Java equivalent.

[6] http://outreach.cs.ua.edu/camps/

[7] http://code.google.com/p/appinventor-java-translation/

ACKNOWLEDGMENTS

We acknowledge the support of a Google CS4HS award. Thanks to Josh Swank for assisting in the conversion of several of the AI programs to the Java Bridge.

REFERENCES

[1] Abelson, H., Mobile Ramblings. *EDUCAUSE Quarterly*, vol. 34, no. 1, 2011.

[2] Alice. Carnegie Melon University. http://www.alice.org.

[3] Android App Inventor from MIT Center for Mobile Learning, http://appinventor.mit.edu.

[4] Barbosa, J., Hahn, R., Rabello, S., and Barbosa, D., LOCAL: A model geared towards ubiquitous learning. *SIGCSE Symposium on Computer Science Education*, Portland, OR, March 2008, pp. 432-436.

[5] Cheung, J., Ngai, G., Chan, S., and Lau, W., Filling the gap in programming instruction: A text-enhanced graphical programming environment for junior high students, *SIGCSE Symposium on Computer Science Education*, Chattanooga, TN, March 2009, pp. 276-280.

[6] Eclipse. ADT plugin for eclipse. http://developer.android.com/sdk/eclipse-adt.html.

[7] Fenwick Jr., J., Kurtz, B., and Hollingsworth, J., Teaching mobile computing and developing software to support Computer Science education. *SIGCSE Symposium on Computer Science Education*, Dallas, TX, March 2011, pp. 589-594.

[8] Goadrich, M. and Rogers, M., Smart smartphone development: iOS versus Android. *SIGCSE Symposium on Computer Science Education*, Dallas, TX, March 2011, pp. 607-612.

[9] Guzdial, M. and Ericson, B., *Introduction to Computing and Programming in Python, A Multimedia Approach*. Pearson Education, 2009.

[10] Lego Mindstorms. http://mindstorms.lego.com.

[11] Nielsen Wire, Young Adults and Teens Lead Growth Among Smartphone Owners. Written September 10, 2012, retrieved December 7, 2012 from http://blog.nielsen.com/nielsenwire/online_mobile/young-adults-and-teens-lead-growth-among-smartphone-owners/

[12] Loveland, S., Human computer interaction that reaches beyond desktop applications. *SIGCSE Symposium on Computer Science Education*, Dallas, TX, March 2011, pp. 595-600.

[13] Mahmoud, Q., Best practices in teaching mobile application development. *ITiCSE Joint Conference on Innovation and Technology in Computer Science Education*, Darmstadt, Germany, June 2011, pg. 333.

[14] Perrone, V. Teaching for understanding: How to engage students in learning. *Educational Leadership*, Vol. 51(5), February 1994.

[15] Roy, K. App inventor for Android: report from a summer camp. *SIGCSE Technical Symposium on Computer Science Education*, Raleigh, NC, March 2012, pp. 283-288.

[16] Scratch. Massachusetts Institute of Technology. http://scratch.mit.edu.

[17] U.S. Census Bureau. Age and Sex Composition: 2010. 2010 Census Briefs, May 2011. Retrieved July 9, 2012. http://www.census.gov/prod/cen2010/briefs/c2010br-03.pdf.

[18] Wolber, D., Abelson, H., Spertus, E, and Looney, L., *App Inventor: Create Your Own Android Apps*. O'Reilly, 2011.

[19] Wolber, D., App Inventor and real-world motivation. In *Proceedings of the 42nd ACM SIGCSE Technical Symposium on Computer Science Education*, Dallas, TX, 2011.