Interplay of Desktop and Mobile Apps with Web Services in an Introductory Programming Course

Amit Shesh
Illinois State University
ashesh@ilstu.edu

Douglas P. Twitchell
Illinois State University
dtwitch@ilstu.edu

ABSTRACT

This paper describes a case study of a second-semester introductory programming course for information systems (IS) students that combined desktop and mobile application development and consumption of existing web services. Our aim was to provide students with a holistic view of how different types of applications can be developed and combined to solve real-world problems, as the students learned the basics of programming. Students progressively built a desktop Java application with a graphical user interface for a local public transit system. It combined the use of basic algorithms, existing web services for geo-coding and mapping to illustrate a recommended route on the system. Students then ported this application to the Android platform re-using most of the code they had already developed. Along with fulfilling the traditional objectives of an introductory course, this course also demonstrated the possible interplay of stand-alone components and web services in desktop and mobile applications and kept the students motivated and engaged throughout the semester.

Keywords

Java programming, Android

INTRODUCTION

Many students entering an IT-related major have little experience as IT developers, but most of them are avid IT users. They heavily use and switch between stand-alone and web applications on their computers and smart devices. We feel that a cohesive exposure to such different kinds of applications from the point of view of developing them may lead to a unique and holistic understanding of their role in solving real-world problems. We attempted to provide such an exposure in an introductory programming course in order to motivate students early on to think about IT development in such a hybrid way. This paper describes our experience in the form of a case study.

Introductory programming courses typically concentrate on stand-alone “desktop” programs. The concepts of web services and applications are introduced in possibly higher-level courses, whereas “non-traditional” applications like mobile apps often do not receive formal classroom time in many curricula (recent exceptions are discussed in Related Work). Thus, unless a student participates in a project that involves integrating these technologies, he/she has little understanding or exposure to the utility, synergy and implementation of applications that are hybrid in nature. Even though many curricula include a “starter” course on IT that discusses its ubiquitous and polymorphic nature, can such an introduction be provided when the discussion moves on to software development?

Our primary motivation was to develop and use material that includes a combination of stand-alone and web functionality to solve a real-world problem progressively in a manner suitable to an introductory programming course. We briefly included mobile app development to further convey how such applications, despite being unique, converge in utility. We believe one of the unique features of our experience is accomplishing this within a single semester-long course. We introduced this material in a second-semester introductory programming course for undergraduate students of the information systems (IS) major. We feel such exposure to the inner workings of different types of applications is especially useful for the future system analyst.

This case shows that likely because of increased enthusiasm, students in the course sections with the progressive assignments performed better than others in similar sections without the treatment and reported that the course fulfilled all relevant course objectives.

Week	Topics
1,2	Review: loops, methods, 1D arrays
3	Classes and multidimensional arrays
4,5	Inheritance, polymorphism, interfaces, abstract classes
6	Exception handling
7	GUIs using Swing
8, 10	Generics and collection classes
11, 12	XML in Java (SAX, DOM)
13, 14	Android development

Table 1: Overview of our course

RELATED WORK

This paper should be considered part of a large body of research showcasing experiences teaching CS1/2 classes using various methodologies. Some studies have explored non-traditional pedagogical methods such as the *Studio* method (Hendrix, Myneni, Narayanan, & Ross, 2010) or pair programming (Simon & Hanks, 2008). Others have discussed the advantages and disadvantages of the objects-first method of teaching Java in CS1 (Reges, 2006). Still others avoid pedagogical arguments in favor of using some "real-world" or "exciting" problem as a framework for teaching the concepts. For example, CS1/2 have been taught using games (Bayliss & Strout, 2006), web applications (Schaub, 2009), web services (Lim, Hosack, & Vogt, 2010) and image processing(Shesh, 2011; Wicentowski & Newhall, 2005). Likewise, CS2 has been taught using bioinformatics (Cutter, 2007), the color image quantization problem (Necaise, 2011). Using these "real-world" problems to teach CS1/2 has, according to the experiences of the instructors involved, increased student engagement. We hoped for the same when using Android as a final project in our second-semester programming course. Since IS students are typically required to take fewer and relatively less rigorous programming courses but are expected to have a holistic understanding of how information systems work, we feel our method of having students implement and integrate different types of applications is uniquely suited to this major. Within the context of the IS major Newby et al.(Newby & H., 2010) test the effectiveness of using the same problem for all assignments, each time using different programming constructs. While we also ask students to implement the same problem definition in multiple ways, our focus is at a higher level (different platforms, frameworks and user interaction).

Courses have been built previously that specifically target mobile application development using Android (Fenwick, Kurtz, & Hollingsworth, 2011; Hu, Chen, Shi, & Lou, 2010; Matos & Grasser, 2010). Android App Inventor (Google Android App Inventor), a web GUI based on Scratch (Scratch) has also been used to introduce programming for CS as well as non-CS/IS majors (Abelson, Chang, Mustafaraj, & Turbak, 2010; Spertus, Chang, Gestwicki, & Wolber, 2010). Although neither the first to introduce a new framework for teaching CS1/CS2 nor the first to use Android in the classroom, we believe our classroom experience is unique as it details the use of Android using Java (rather than visually using Scratch) and combines technologies to solve a single problem progressively in an introductory programming course.

CONTEXT AND OVERVIEW OF OUR COURSE

The Course

The course in question, titled "Computer Application Programming", is offered by the School of Information Technology at our university that, among other IT-related majors, offers the Information Systems (IS) major for undergraduates. This is an "IS2" course, i.e. the second of two mandatory programming courses for all IS majors. The programming language used is Java, and the syllabus includes topics related to Java and object-oriented (OO) concepts (summarized in Table 1). Typically students in this course only have one semester's worth of programming experience (the IS1 course before it) that includes basics such as control statements, loops and simple OO concepts. This course lays the foundation for higher-level courses involving system analysis and design and other technical topics that assume knowledge of Java and OO concepts. Both sections offered in this particular semester used the same assignments, discussed below.

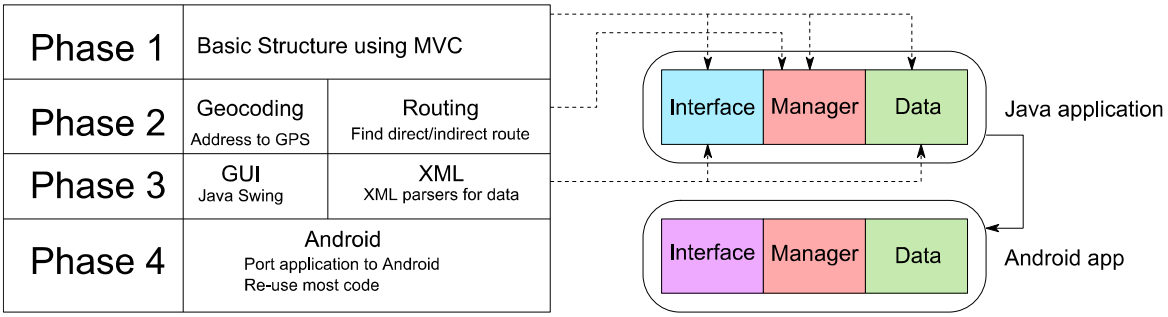


Figure 1: Illustration of flow of the assignments. The overall application development was divided into 4 phases, carried out in sequence. Phases 1,2 and 3 culminated in a Java GUI-based application complete with a GUI, while Phase 4 ported most of its code to the Android platform. The dotted arrows show how various phases contributed specifically to various layers of the application.

Overview of Assignments

We designed the assignments of this course so that they progressively built a single application related to the local public transit system. Our public transit system plies buses on 11 routes throughout the city. This application assists users to find appropriate bus routes to go between two street addresses in the city. Although all the route times and maps are available electronically (as static pdf files) and in paper form and bus locations can be tracked in real-time, currently there is no official computer-based application of any kind that lets users plan routes based on source-destination and intended time of travel. Thus our application addresses a practical, real-world requirement.

Our assignments divided the application into four unequal phases. In the first phase, students created the necessary classes to represent relevant aspects of the application in a model-view-controller (MVC) design. They were provided bus stop and route data in simple formatted text files. In the second phase, they used existing web services to convert street addresses into geo-locations (i.e. latitude and longitude) in order to determine appropriate starting and ending bus-stops. Using this information they implemented a simple greedy algorithm to find direct and indirect routes between two bus-stops. In the third phase, they implemented a graphical user interface that allowed a user to input relevant information and see the details of the relevant route in text and also plotted on a map. Also, they wrote XML parsers to read the bus stop and route data from XML files. In the final phase, they ported their Java application onto the Android platform, by re-using most of their code and creating a new interface suitable for Android devices. Figure 1 illustrates the progressive development of this application.

DETAILS OF ASSIGNMENT

This application was implemented in 5 assignments out of a total of 8 assignments in the semester (the other 3 were unrelated assignments).

Assignment 1 (Phase 1)

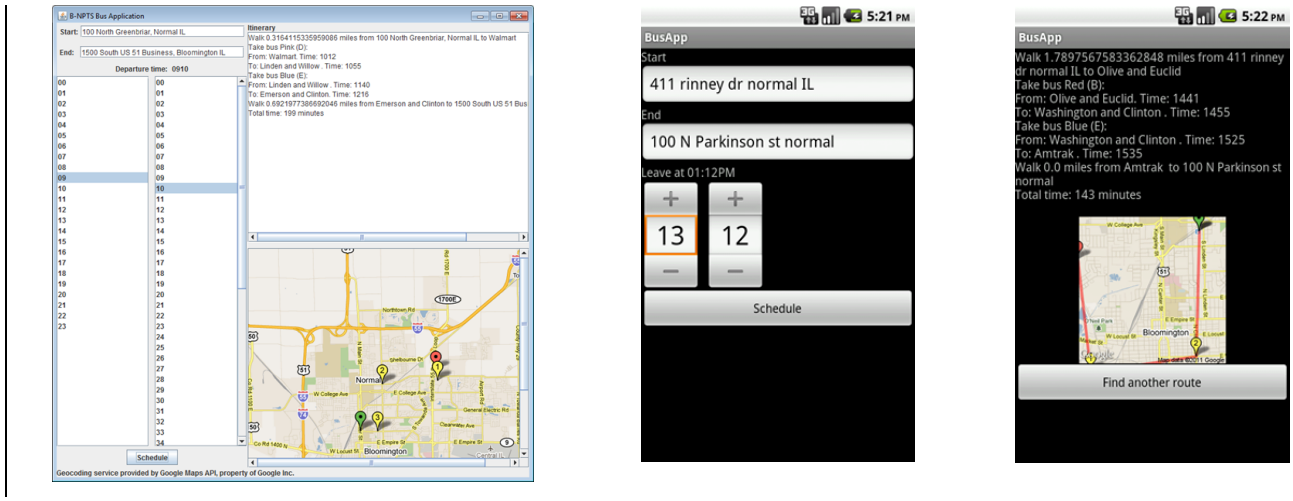
This was the first and smallest assignment of the application, and asked them to create the overall structure of a simple bus application. Specifically students were asked to:

1. Create a *BusStop* class that represented a single bus stop, comprising of its name, its location in GPS coordinates and a unique application-specific ID.
2. Create a *BusRoute* class that represented a single bus route, comprising of its name, a list of stop IDs along that route, and a table of arrival times whose rows were bus stops and columns were physical buses plying on that route.
3. Create a *BusManager* class that manages all the bus stops and routes read from provided text files.

The structure imposed a very simple model-view-controller architecture with the first two of the above classes representing the model, the third class representing a controller and a simple “Main” class serving as a placeholder for the view. This class tested the program by providing names of data files and printing out the details of a given bus route. The data files were simple text files in a pre-defined format. The GPS locations of all the bus stops were obtained manually by the instructors using an available free online geocoder (GPS Visualizer).

Assignment 2 (Phase 2)

This assignment implemented most of the application logic and also used web services for geo-coding. Specifically students were asked to:



(a) (b)

Figure 2: Screen captures for GUI development. (a) The Swing GUI for our application for Assignment 3. (b) Recommended screen captures for the Android GUI.

1. Create a *GPSLocation* class that represents one GPS location, and write a method to compute the Euclidean distance between two locations.
2. Write a method in the *BusRoute* class that accepts as parameters two stop IDs and a time, and returns the earliest time after the provided time that a bus could be taken from the first stop. A similar method was written for the latest bus that reached the destination before the provided time.
3. Learn the use of a *Geocoder* class provided to them that used the OpenStreetMap geo-coding service(Nomatim).
4. Write a method in the *BusManager* class that would accept a GPS location and return the nearest bus stop to it.
5. Write a method in the *BusManager* class that would accept a starting and ending address and an intended departure time, convert the addresses into GPS locations and return the details of a direct route between the two, if one exists.
6. Write a method in the *BusManager* class that would accept a starting and ending address and an intended departure time, convert the addresses into GPS locations and return the details of a indirect route (exactly one intermediate stop) between the two, if one exists.
7. Write a method in the *BusManager* class that would accept a starting and ending address and an intended departure time, returned either a direct route between the two, or an indirect one if a direct route does not exist (using the above two methods).

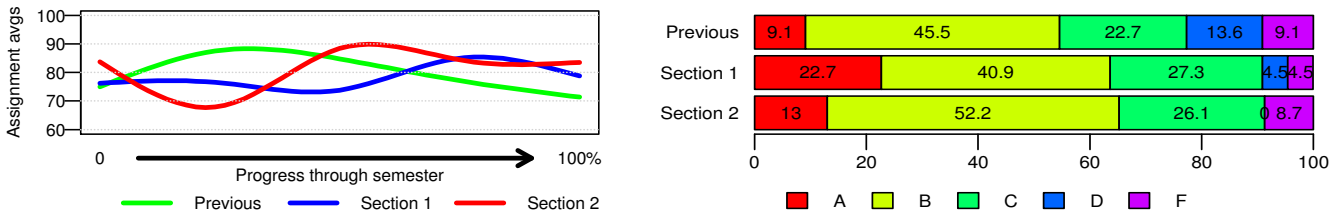
For extra credit, students could write methods that returned routes that considered an intended arrival time.

Geocoder

A *Geocoder* class was directly provided to them for use in this assignment as writing one would require knowledge of Java that was outside the scope of this course (in later assignments they were asked to extend it or use it to write other code). This class accessed a web service and parsed the returned XML to extract the GPS location. Students were required to be connected to the internet for the geocoder to function. Students faced some difficulty with the OpenStreetMap service as its address-parsing was not very robust (e.g. occasionally it would not return a GPS location if a street direction was not mentioned). This was addressed in a later assignment.

Routing

Students were asked to implement a brute-force greedy algorithm for indirect routes: given that the user wishes to go from A to B, consider each stop C as a potential intermediate bus stop and determine if there exist direct routes (A→C) and (C→B) and return the first one that is found. Thus the returned route was not guaranteed to be optimal in any way. We considered asking students to implement a shortest path algorithm (e.g. Dijkstra's shortest path). In this problem it would require creating a more complicated graph. Since IS students typically lack the knowledge of algorithms required for this problem we could



(a) (b)

Figure 3: Performance comparison with a previous semester (without described assignments). (a) Trend of average performance in assignments. Although the ranges of average performance are comparable, both sections with the described assignments show a slight increase in performance towards the end of the semester. (b) Plot comparing percentage of letter grades per section. Sections with described assignments showed a moderate increase in the number of A's, B's and C's over the previous semesters.

not frame it as a manageable assignment without disclosing other details that would be outside the scope of the course. Nevertheless students regarded this as one of the more time-consuming and difficult assignments of the semester.

Assignment 3 (Phase 3)

This assignment asked students to create a graphical user interface for their program. They were provided with a screen capture (shown in Figure 2(a)) and were expected to replicate its look-and-feel using Java Swing. They had to write all code from scratch--the use of any drag-and-drop GUI design tool was strictly forbidden. Also, an updated *Geocoder* class was provided that used a web service to fetch a map centered at the mean of the GPS locations given to the geocoder, such that all the locations were visible in it. Using Java, the map was available as an *InputStream* object that could be directly plotted on a *JLabel* like an icon.

The geocoder in this assignment used the Google Geocoding API (Google Geocoding Service API) which offers a robust address-parsing service. Google's terms of service mandate that any results of its geocoding service be shown on a map. As earlier versions of our program could not display a map, we could not use this web service earlier. This geocoder also used the Google Static Maps API (Google Static Maps API) to get the map. Students were referred to its documentation and were asked to extend this method for extra credit so that it plots the suggested bus route(s) (by drawing lines between source, destination and intermediate bus stops, although it would be possible to trace the route using other bus stops on those routes).

Assignment 4 (Phase 3)

This assignment asked students to migrate the bus stop and route data to XML, and then write SAX (Simple API for XML) XML parsers using standard Java classes(Parsing an XML file using SAX) so that their application read data from them. For the former, students had to simply "print" all the data read from the text files in valid XML format. Since the XML parsers in Java check for invalid XML formats, their own parser verified whether their XML writing was correct. Thus at the end of this assignment, students had developed a GUI-based application that read XML data and determined and plotted a suitable itinerary using the local bus transit system.

Assignment 4 (Phase 4)

In the final assignment, students had to "port" their application so that it ran successfully on an Android device. Screen captures were provided to students (Figure 2(b)), although they were given the freedom to create different-looking GUIs so long as they sufficed to run the application (i.e. specify addresses and a departure time, show the route and the map, and find another itinerary without having to restart the app). Once again they were discouraged from using drag-and-drop GUI designers for Android.

Several choices made earlier made this assignment tractable and meaningful. First, since the rest of the course was taught in Java, a very short introduction to Android development (approx. 2 weeks) sufficed for this assignment. Secondly, as we used the MVC architecture, the separation between parts that needed to be changed was very clear (students had to simply replace the GUI without changing anything else). Thirdly the prior use of XML made this mobile app efficient as Android (like other mobile platforms) seems fairly optimized to work with XML data. Moreover as Android GUIs are typically specified in XML, a prior exposure to XML helped students in creating Android GUIs without drag-and-drop design tools. Lastly as we used the Eclipse (Eclipse) IDE throughout the course, the fact that it is one of the recommended IDEs for Android development made the transition very smooth for students.

Question	Rating	% students rating (out of 5)			
		Section 1		Section 2	
		1-2	4-5	1-2	4-5
Gaining factual knowledge (terminology, classifications, methods, trends)	Important	0%	90%	6%	78%
Learning fundamental principles, generalizations, or theories	Important	5%	75%	6%	83%
Learning to apply material (to improve thinking, problem solving, and decisions)	Essential	5%	85%	11%	72%
Developing specific skills, competencies, and points of view needed by professionals in the field most closely related to this course	Essential	0%	80%	24%	71%

Table 2: Results from IDEA evaluations. The rating of each objective was given by faculty who regularly teach this course. The responses show that most students perceive that the objectives considered most important were fulfilled in this experimental way of teaching it. (Note: Every row does not add up to 100% because (i) IDEA does not report statistics about the rating of 3 (ii) not all surveyed students answered every question.)

IMPACT AND FEEDBACK

In general students responded enthusiastically to the assignments. Many students initially felt that the bus transit system was too difficult for a course at the beginner's level, but after the GUI assignment it was satisfactory for many to see a complete program functioning properly. Android development was an unofficial inclusion in the course: the instructors announced initially that some Android development will be included the course if they felt the students were capable of grasping the material, based on overall class performance. This kept students motivated as the instructors occasionally got “update queries” in class from students wondering if Android would be covered in the end.

We introduced these assignments in two sections of the course with 23 students (after initial dropouts) each. Figure 3 compares these sections to a section of the same course with 22 students taught in a previous semester. Figure 3(a) plots the trend of average performance of all students in assignments in each section (10 assignments in the previous semester were grouped into 5 sets of 2 consecutive assignments each to compute averages). As the objectives of assignments in the previous semester could not be correspondingly matched with those described here this plot meaningfully illustrates only the progress in a semester. It shows that although the ranges of performances are comparable, sections in our trial performed better towards the end of the semester. Figure 3(b) shows how sections with the described assignments saw a moderate increase in the percentages of A's, B's and C's over those in the previous semester. A more thorough analysis over several semesters with comparisons to control courses could reveal finer performance trends among various student demographics and provide stronger evidence of the utility of this approache. Some details of standardized IDEA evaluations (The IDEA Center) for these two sections (Table 2) show a favorable student perception about whether they satisfied the essential course objectives identified by our faculty.

LIMITATIONS AND VARIATIONS

We realized some limitations of the way we implemented our approach, many of which we plan to address in future. First although students were enthusiastic about Android, we could not devote enough time to it during the semester. Secondly dividing some of our assignments into smaller ones could have made them more focused and tractable. Thirdly, although unsuitable for our course, students could have achieved more within the same time by working in groups.

Our assignments can be varied in several ways to suit other types of courses while retaining our hybrid theme. First, the advantage of smooth Java-Android transition can be accrued for other programming languages (C# with Windows Mobile (Windows Mobile Development), C++ with Android NDK (Android NDK) or Qt mobile development (Qt)). Secondly, more emphasis could be placed on using better algorithms to determine optimal, multi-hop routes based on shortest distance, time or other constraints. Implementing these algorithms efficiently on a mobile platform would present unique challenges. Such an approach would be more suitable in a formal algorithms course for computer science students. Thirdly, the role of web services can be increased by using interactive maps and relevant points of interest along a given route. Alternatively the entire implementation could be offered as a web service feeding both client-side web and mobile apps, essentially making the latter independent of platform. Fourthly the design of such a system could be a subject of an analysis-and-design course: how best to design a system that integrates stand-alone and web services and adapts easily to many traditional and mobile platforms. Lastly if a course concentrates on mobile development, features like GPS can be utilized to create an app that adapts to a user's location in real-time.

REFERENCES

1. Abelson, H., Chang, M., Mustafaraj, E., & Turbak, F. (2010). Mobile phone apps in CS0 using App Inventor for Android: pre-conference workshop. *J. Comput. Small Coll. , 25*, 8-10.
2. *Android NDK*. (n.d.). Retrieved 2013, from http://developer.android.com/sdk/ndk

3. Bayliss, J. D., & Strout, S. (2006). Games as a "flavor"' of CS1. *SIGCSE Bull. , 38*, 500-504.

4. Cutter, P. (2007). Having a BLAST: a bioinformatics project in CS2. *SIGCSE Bull. , 39*, 353-357.

5. *Eclipse*. (n.d.). Retrieved 2013, from http://www.eclipse.org

6. Enbody, R. J., & Punch, W. F. (2010). Performance of Python CS1 students in mid-level non-Python CS courses. *Proc. SIGCSE*, (pp. 520-523).

7. Fenwick, J. J., Kurtz, B. L., & Hollingsworth, J. (2011). Teaching mobile computing and developing software to support computer science education. *Proc. SIGCSE*, (pp. 589-594).

8. *Google Android App Inventor*. (n.d.). Retrieved 2013, from http://beta.appinventor.mit.edu/about/

9. *Google Geocoding Service API*. (n.d.). Retrieved 2013, from https://developers.google.com/maps/documentation/geocoding/

10. *Google Static Maps API*. (n.d.). Retrieved 2013, from https://developers.google.com/maps/documentation/staticmaps/

11. *GPS Visualizer*. (n.d.). Retrieved 2013, from www.gpsvisualizer.com/geocoder/

12. Hendrix, D., Myneni, L., Narayanan, H., & Ross, M. (2010). Implementing Studio-based learning in CS2. *Proc. SIGCSE*, (pp. 505-509).

13. Hertz, M. (2010). What do "CS1" and "CS2" mean?: Investigating differences in the early courses. *Proc. SIGCSE*, (pp. 199-203).

14. Hu, W., Chen, T., Shi, Q., & Lou, X. (2010). Smartphone Software Development Course Design based on Android. *Intl. Conf. Comp. & Info. Tech.* , 2180-2184.

15. Lim, B. L., Hosack, B., & Vogt, P. (2010). A web service-oriented approach to teaching CS/IS1. *Proc.SIGCSE*, (pp. 131-132).

16. Matos, V., & Grasser, R. (2010). Building applications for the Android OS mobile platform: a primer and course materials. *J. Comput. Small Coll. , 26*, 23-29.

17. Necaise, R. D. (2011). Using the color image quantization problem as a course-long project in CS2. *Proc. ACM-SE*, (pp. 54-59).

18. Newby, M., & H., N. T. (2010). Using the same Problem with Different Techniques in Programming Assignments: An Empirical Study of its Effectiveness. *Journal of Information Systems Education , 21* (4), 375-382.

19. *Nomatim*. (n.d.). Retrieved 2013, from http://nominatim.openstreetmap.org/

20. *Parsing an XML file using SAX*. (n.d.). Retrieved 2013, from http://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html

21. *Qt*. (n.d.). Retrieved 2013, from http://qt.digia.com/

22. Reges, S. (2006). Back to basics in CS1 and CS2. *SIGCSE Bull. , 38*, 293-297.

23. Schaub, S. (2009). Teaching CS1 with web applications and test-driven development. *SIGCSE Bull. , 41*, 113-117.

24. *Scratch*. (n.d.). Retrieved 2013, from http://scratch.mit.edu/

25. Shesh, A. (2011). High-Level Application Development for non-Computer Science majors using Image Processing. *Proc. EG - Education Papers*, (pp. 37-41).

26. Simon, B., & Hanks, B. (2008). First-year students' impressions of pair programming in CS1. *J. Educ. Resour. Comput. , 7*, 5:1--5:28.

27. Spertus, E., Chang, M. L., Gestwicki, P., & Wolber, D. (2010). Novel approaches to CS0 with App Inventor for Android. *Proc. SIGCSE*, (pp. 325-326).

28. *The IDEA Center.* (n.d.). Retrieved 2013, from http://www.theideacenter.org/

29. Wicentowski, R., & Newhall, T. (2005). Using image processing projects to teach CS1 topics. *SIGCSE Bull. , 37*, 287-291.

30. *Windows Mobile Development*. (n.d.). Retrieved 2013, from http://msdn.microsoft.com/en-us/windowsmobile/bb264318