7-2013

Teaching and Assessing Programming Fundamentals for Non Majors with Visual Programming

William L. Honig
Loyola University Chicago, whonig@luc.edu

Recommended Citation

William L. Honig. 2013. Teaching and assessing programming fundamentals for non majors with visual programming. In Proceedings of the 18th ACM conference on Innovation and technology in computer science education (ITiCSE '13). ACM, New York, NY, USA, 40-45. DOI=10.1145/2462476.2462492 http://doi.acm.org.flagship.luc.edu/10.1145/2462476.2462492

Teaching and Assessing Programming Fundamentals for Non Majors with Visual Programming

William L. Honig
Loyola University Chicago
Department of Computer Science
Chicago, Illinois 60611 USA
+1.312.915.7988
whonig@luc.edu

ABSTRACT

Visual programming tools and mobile device applications are a natural tool to engage university students; but, are they effective in teaching quantitative thinking skills to non computer science majors? Answering this question can be based on careful assessment of the learning outcomes.

This paper reports the results from teaching over 100 students mobile app development with App Inventor in a university core course. Results were measured using an assessment process motivated by Bloom's Taxonomy that included student self assessment, ratings by instructors, and comparisons of the two results. The categories in the assessment were mapped to specific levels of skills with various App Inventor components.

Results presented here confirm App Inventor's effectiveness and ability to motivate students. App Inventor features and components that most impacted the student learning are noted.

The assessment results show the course was very successful particularly in the three assessment categories of Remembering, Understanding, and Application (Lower Order Thinking Skills) and acceptably successful in Analysis, Evaluating, and Creating (Higher Order Thinking Skills). The paper concludes with suggestions on continued improvement of the course content and additional App Inventor features that should become part of the assessment process.

Categories and Subject Descriptors

D.1.7 [**Programming Techniques**]: Visual Programming.

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *self-assessment.*

General Terms

Measurement, Experimentation.

Keywords

App Inventor, assessment, Bloom's Taxonomy, mobile apps.

ITiCSE'13, July 1–3, 2013, Canterbury, England, UK.

1. INTRODUCTION

This paper presents an approach to teach and assess learning outcomes in a university quantitative knowledge course for non computer science majors. The course uses mobile applications created with App Inventor, a visual programming tool, to develop skills in symbolic reasoning, problem solving, and consequence prediction.

App Inventor has become popular in the classroom for introductory courses for computing students [5, 10, 12, 13] and to a lesser extent with general students [4]. The course structure reported here is similar to these other uses of App Inventor; this paper matches particular features of App Inventor to different types of learning and suggests a structured assessment process to better evaluate student`s learning outcomes.

The assessment process was developed using Bloom's Taxonomy and includes evaluations by both students and instructors. Although Bloom's Taxonomy inspired assessment has been used in computer science [7, 9, 11], it could be used more often and potentially improve learning assessment and comparison. The work presented here includes mapping specific skills with App Inventor features to assessment categories and ratings. The assessment approach is new for this type of course and is suggested as a more rigorous method for determining the success of similar courses.

Current results from the assessment of over 100 students in both online and traditional classroom courses indicate that the course and App Inventor are successful: over two-thirds of students met the most demanding measure of success. The findings to date also show that student self assessments are generally lower than instructor ratings. The results also suggest potential improvements of the assessment process by using additional App Inventor features mapped to certain skill levels.

2. COURSE STRUCTURE AND MOTIVATION

2.1 Course Purpose and Intent

The course, titled Visual Information Processing, is a undergraduate course students may take to meet a university wide requirement for "quantitative knowledge" learning. All courses meeting this requirement teach students to understand and analyze information presented in multiple formats, determine various ways to solve problems, and learn to interpret information and predict consequences. The course has no prerequisites and is open to, and attracts, students from all backgrounds.

In the past the course used the Visual Basic programming language. The course was restructured and modernized to increase its appeal to students while maintaining the necessary leaning goals to satisfy the university quantitative knowledge core requirements. Due to the popularity of mobile devices and applications and the timely availability of the App Inventor visual programming tool, the course was redone to teach quantitative skills using mobile apps. The course was *subtitled Mobile Apps with Google Android using App Inventor.*

2.2 Instruction Topics and Programming Concepts

The instruction and student assignments in the course were selected to teach the following programming and systems concepts:

- Basic elements of human computer system interface including buttons, labels, text boxes, and selection lists
- Information storage in text strings, numeric form, program variables, and persistent data bases
- Media beyond text and numbers including sound, video, graphics, and drawings
- Key structure of object oriented systems including abstract classes, object instances, global variables, attributes, behaviors, and basic control structures
- Events and timing as mechanisms to drive app processing and responses
- Basic analysis and design techniques to convert a desired functionality to working software.

Students built their understanding of mobile apps by completing a common set of assignments and then, in the final third of the course, creating a custom app to do something they found interesting or useful. The initial common assignments were designed to cover multiple ways of getting and using information and introduced students to the abilities and limitations of the App Inventor tool. The custom app was open to student's imagination to encourage experimentation and exploration of programming concepts similar to studio learning and tinkering courses [3, 4, 8].

Figure 1 is an example of a student's custom course project incorporating the above programming concepts. The student conceived, designed, implemented, tested, and extended an app to create graphical greetings with color, text and drawings, and the option to save and send the resulting greeting to others.

2.3 Visual Programming with App Inventor

App Inventor is a graphically rich visual programming tool for Android mobile apps. It is intended to allow anyone, including those with no background or interest in programming to develop mobile apps. There are three major parts of App Inventor all of which run on most personal computers with support from cloud servers (hence doing cross development of apps to eventually run on mobile devices). App Inventor is descendent from Scratch and was developed by Google and later transferred to MIT's Center for Mobile Learning. App Inventor and its graphical programming tools are not described in detail here (see [5, 10, 12] for more on App Inventor.)

The Designer allows the selection of "components" (objects of a certain class) to be added as parts of an app. Once a part is added to the app, by dragging it onto a screen layout, its static attributes or properties can be set.

Figure 1. Student Paint-Tastic App Running on Emulator

The Blocks Editor uses jig saw puzzle "blocks" (methods of the related class) which snap together to define the dynamic behavior of the app in response to various events such as the user touching the screen or the selected system updates (e.g. a location update). App Inventor does not require the developer to distinguish between types (numbers and strings may be interchanged).

App Inventor allows the app to run on Android phones or an emulator of a phone which runs on the personal computer. Changes to the app are automatically pushed to the connected phone or emulator as soon as they are made (there is no concept of compiling).

App Inventor is limited compared to a normal programming environment. It does not allow creation of new components (new classes) and does not allow control over priorities of events. It is, however, very feature rich and able to generate sophisticated mobile apps in a visual programming language.

3. EVALUATION TECHNIQUE

The course was conducted as an experiment to determine if visual programming with App Inventor could both interest students and achieve the quantitative knowledge requirements for the university core curriculum. Student learning of basic programming concepts (which had formerly been taught in Visual Basic) was part of the evaluation.

A formal evaluation incorporating both student and instructor input was created based upon Bloom's Taxonomy. Bloom's Taxonomy was conceived to improve communication and comparison of test results by giving better precision to terms such as "thinking" and "problem solving" [2] and later updated to

support standards-based curriculum planning and evaluation tools[1].

In the work reported in this paper, each of six cognitive categories or types of learning in Bloom's Taxonomy were related to skills and abilities to use App Inventor and rated from 1 to 5. The same numeric ratings apply to each skill category with the meanings shown in Table 1. Ratings of 3 and above indicate successful learning outcomes. Ratings of 1 or 2 generally indicate less the adequate knowledge and a lack of success in achieving the university core course goals.

Table 1. Ratings Levels, General Portrayal

Numeric Rating	Description (for all Categories)
1	No or Limited Evidence of Learning
2	Less Than Workable Knowledge
3	Adequate Knowledge of Desired Learning
4	Full Mastery of All Basic Skills
5	Advanced Skills

While each category uses numeric ratings with the same general meanings, the categories address learning at different skill levels (Table 2.) The categories increase in complexity of learning demanded from the initial Remembering to the sixth and final Creating.

For the course assessment student skills with specific App Inventor components were matched with each category. Table 2 summarizes the assessment tool used. Due to space limitations the entire assessment tools does not appear here (for more details on the ratings and relationship to App Inventor concepts see the online version [6].)

For example, the Application category is intended to evaluate the student's ability to apply something learned to a new situation or need. It was measured using the events concept in App Inventor which are either user inputs to a mobile app or caused internally by something in the phone or network. The ratings were detailed as follows for this category:

1. Lack ability to apply known concepts of events to new problems.
2. Able to use single events to perform new functions, e.g. changing different properties of a known component such as a text box.
3. Ability to use appropriate events for a consistent GUI and app execution. Includes GUI events and others such as Screen Initialize, and Clock timer.
4. Able to use multiple events to perform tasks that requires multiple GUI inputs.
5. Ability to demonstrate use and application of known complex component events (e.g. sensors and animation) and to utilize them to solve a new problem.

The full assessment matrix was known to students and instructors during the course. However, to prevent bias the students did not have the general meaning of the numeric ratings shown in Table 1.

Table 2. Visual Programming Assessment Summary

Cognitive Category	Measure	Ratings
Remembering	Ability to recognize App Inventor components used in mobile apps	(1)unable to identify, (2) basic components (labels), (3) most components (colors), (4) complex components, (5) beyond those used such as orientation sensor,WebDB.
Understanding	Ability to explain and compare usefulness of components used in apps	(1) unable to explain, (2) explain some (text box), (3) explain correct use, (4) explain and select multiple components, variables, (5) understanding and explanation of complex components, animation.
Application	Ability to apply the events concept (WhenDo) to unfamiliar problems	(1) lack ability, (2) able to use single event, (3) use events for consistent GUI and execution, (4) use multiple events, (5) use known complex component events such as sensors for new problems
Analysis	Skill in analyzing apps, selecting appropriate implement-tation components, dividing needs into elements	(1) lack ability to define approach, (2) able to use some but not all implementation approaches, (3) decompose needs into appropriate elements , (4) ability to suggest alternatives to others, (5) able to select good approaches from among alternatives
Evaluating	Skill in making judgments about your app and implement-ation including identification of limitations	(1) unable to evaluate problems on own app, (2) perceive problems but not always in correct terminology, (3) communicate limitations and problems in apps, (4) discuss how and why others have used components, (5) skillfully judge App Inventor limitations and test for bugs
Creating (applies to custom project only)	Ability to define and construct apps and features not covered in class, including defining app needs and exploring solutions.	(1) no ability to create new approaches, (2) able to create new apps with help and advice, (3) understand design, planning, and implementation, (4) able to extend and combine examples, (5) Devise own original solution and integrate Android activities

Students worked from the detailed descriptions of App Inventor capabilities[6] in an online survey form.

Students evaluated their own learning at the end of the class with assurances that their evaluation did not impact grading. The student's assessment of each category was both a numeric rating (1 to 5) and a text explanation for their rating using examples from their work. Students were given credit for fully and thoughtfully completing the assessment. Faculty and teaching / research assistants involved in the class independently evaluated each student's learning and the results were combined and compared (Section 4.)

4. FINDINGS AND LEARNING OUTCOMES

The course has been taught several times to over 100 students in both traditional and online versions. The traditional sessions take place in a lab with students sitting at a computer or using their own computer for hands on work. The online classes use video materials and regular synchronous sessions with instructors and students online together using screen sharing and audio/video conferencing.

4.1 Student Self Evaluation

Figure 2 shows the distribution of ratings in each category when students evaluated themselves at the end of the course. For example in the first category, Remembering, over 20% of students measured their leaning at the top level of 5 (able to recognize an App Inventor component even if it had not been seen before) and only a few percent gave themselves a rating of 2 (remembering only very basic components such as a GUI button).

The Remembering, Understanding and Application categories (sometimes termed Lower Order Thinking Skills) are basic to working with the mobile apps in the course; no students were unable to work at this level and only a few percent rated themselves 2 (knowledgeable limited to basic and individual components and events).

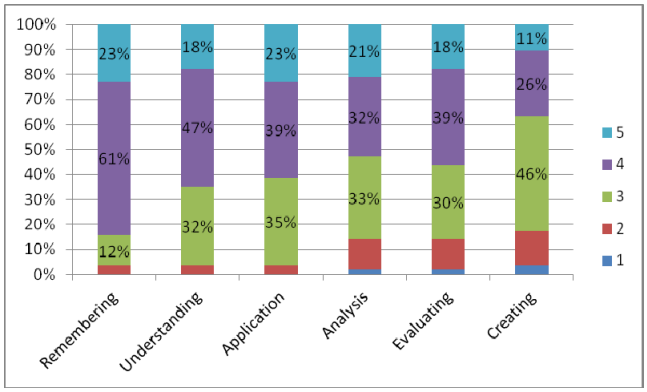

Figure 2. Student Self Evaluation in Assessment Categories

In the assessment students were required to give examples from their work in support of the ratings. Their comments show that students understood the assessment criteria fairly well:

- Remembering (self rating 3) I would be able to recognize the components themselves if such as buttons, text boxes, clock etc but when it comes to activity starter or SMS components, it would take me long to be able to figure out what they are…
- Remembering (self rating 4) with each assignment, we built upon the information from the previous assignment…we learned about the tinyDB component after we knew very well how to program what we were trying to save.
- Understanding (self rating 4) I can explain pretty much everything but the sensor area is still kind of fuzzy…
- Application (self rating 4) …I knew how to initialize events when the screen was opened, when certain buttons were clicked, when a math total was met (and if it wasn't met I knew how to make another event happen (if/else)…

The graphical screen interface items and text messaging components of App Inventor were areas many students mentioned as examples of their success. TinyDB was also mentioned often as a difficulty by some and as a positive learning by others.

For the final three categories, Analysis, Evaluating, and Creating, (Higher Order Thinking Skills), more students applied the lower two ratings to themselves; for example, 17% admitted that they were unable to fully create new custom applications without considerable help from others.

Similarly, the student's comments showed they were making reasoned choices on their ratings:

- Analysis (self rating 2) I'm no expert and usually am only able to figure out what I am having trouble with or get help from others more than me helping them …
- Evaluating (self rating 3) When I was watching the application demos of my peers I was often able to envision the blocks and components before they brought up their block editor. For example the application that was Catch Phrase I recognized that all his random words would…
- Creating (self rating 4) I'm not at pro level yet but… having a better idea of my own interests for art and implementing it into my Paint-Tastic app…

App Inventor's activity starter component (to allow app to call other apps including those built into the phone) was most often mentioned as a challenging part of the course. Students made many favorable comments about their ability to conceive and create a complete app from their own requirements.

Over 80% of students rated themselves 3 or higher across all evaluation categories indicating they felt they had successfully met all the learning objectives of the course. For the initial three categories this figure raises to over 95%. Combined these outcomes would represent a very satisfactory accomplishment of the goals for the university core course.

The self assessment is the student's own opinion; how would they be evaluated by others? Although the evaluation process was designed to be separate from grading to encourage honest student feedback, it is possible students were overly optimistic about what they learned in the course.

4.2 Instructor's Evaluation and Comparison to Self Assessment

Each student was also separately evaluated by the faculty member responsible for the course and independently by one or more advanced students working with the course as teaching assistants or research assistants. These students were directly involved with the course by participating in assignment preparation, lectures, grading, and classroom activities.

The instructor evaluation used the same assessment materials (the fully extended version of Table 2 [6]) and was based on actual student performance on class work (primarily app development projects but also including exams and student presentations). Each student was evaluated independently by multiple instructors (faculty member or advanced student); the multiple instructor ratings were usually quite similar differing by no more than one point. Instructors completed their ratings without seeing student's self assessment.

Comparison of student self assessment and instructor evaluations is one test of the effectiveness and precision of the evaluation matrix. Section 5 suggests ways to potentially improve this comparison and further explore how to achieve the correct and best possible learning assessment.

The teaching staff generally rated the student learning outcomes at higher levels than in the self assessment the students performed (see Figure 3.) More students were rated 5 as having shown evidence of advanced skills in all categories than in the student's self assessment. This rating required student success at using aspects of App Inventor beyond what was covered in class and assignments.

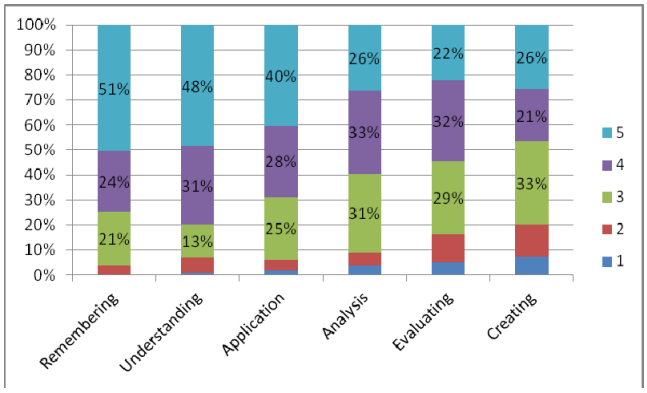

Figure 3. Faculty and Staff Ratings of Students

The differences were largest in the first three categories. Each of Remembering, Understanding, and Application had twice as many students rated 5 as in their self assessment. For example, the Application category, which was evaluated with the event concept of App Inventor rated over 40% of students at 5 (Advanced Skills) including some students who self rated at 3 (Adequate Knowledge).

In the Analysis, Evaluating, and Creating categories, instructor and student assessments were closer to each other. Figure 4 compares the two assessments in the Analysis category by looking at differences on specific students (whereas Figures 2 and 3 above shows totals for each category and rating). For this figure multiple instructor ratings were averaged and rounded to integers for ease of presentation. Here, as in other Higher Order Thinking Skills, instructor ratings diverged lower and higher than the self assessment; however, instructor rating were still typically higher.

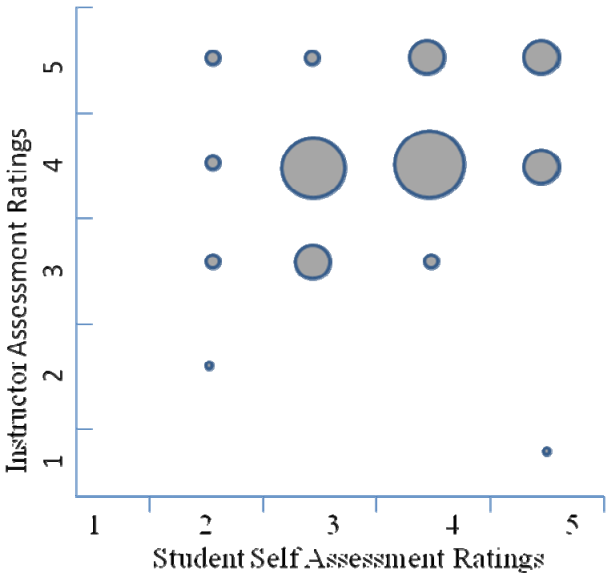

Figure 4. Analysis Category Comparison, Individual Students (Size of Circle Gives Number of Students)

Students who rated themselves 2 in Analysis were least aware of their standing in the course; while students that the instructors thought were twos agreed, most who self rated at two were judged better by their instructors.

4.3 Overall Course Outcomes

For a final measure of the learning effectiveness of Visual Programming with App Inventor, the performance of all students was compared to the goal of a rating of 3 or higher in all categories. The measure chosen was the minimum of the student self assessment and the instructor ratings. The measure is very conservative as it requires both student and instructors to agree that a minimum level of learning was achieved.

Table 3 shows this evaluation; almost all students achieved ratings above 3 in the first three categories and approximately 70% to 80% in the higher three categories. Considering all categories two thirds of the students achieved a rating of 3 or higher in all categories.

Table 3. Minimum Evaluation, Percent of Students

Rating	Remembering	Understanding	Application	Analysis	Evaluating	Creating
1	0%	0%	0%	2%	4%	8%
2	4%	4%	4%	15%	15%	19%
3	17%	34%	43%	36%	43%	45%
4	64%	49%	38%	36%	32%	25%
5	15%	13%	15%	11%	6%	4%

4.4 Changing Impressions

Although not part of the formal assessment materials, students were also surveyed on their attitude toward programming among other things. Not surprisingly for a course that attracts students seeking to fulfill a university common requirement, 68% had never considered a career in computing or programming. The course changed their impression of what programming is about; of those originally not at all interested in software careers the majority (over 60%) became positive indicating they were more likely to consider a computing career. Several students from the course are now majoring in computer science.

5. NEXT STEPS AND FURTHER COURSES

Future courses in similar topics can benefit from the experiences reported here including how to use App Inventor to teach and assess learning of quantitative thinking skills.

5.1 App Inventor and Learning

Students demonstrated again that App Inventor is a compelling and effecting tool for learning. Students were generally seeking even more time to work with their apps, more opportunity to work with actual phones and sensors, and anxious to attempt team or multiple person app development.

The student comments confirmed that many features of App Inventor are easy and enjoyable to use (graphics, integration of media files, lists, and text manipulation all were often noted). Difficulties arose for students in some areas, particularly those requiring somewhat deeper access into the underlying Android operating system (Activity Starter, multiple screen apps, and TinyWebDB were most mentioned).

5.2 Improvement of the Assessment Matrix

The course assessment techniques, based on Bloom's Taxonomy worked effectively to enable both students and instructors to distinguish levels of student learning. However, it may be improved to add precision in each rating category with additional examples of App Inventor components that apply. The current assessment matrix does not include any evaluation of some traditional programming elements (control structures including loops and decision logic, subroutines and parameters). Adding these to the assessment, likely in categories Analysis, and Evaluating, would expand the assessment process.

Team work assignments seem an obvious addition to the Evaluating and Creating categories. As well as responding to student desire, such an addition would allow student participation in each other's assessment.

6. SUMMARY

Visual programming with App Inventor has again been shown popular and effective with students, here with a wide variety of non computer science majors. The assessment results, based upon a mapping of App Inventor usage skills to the categories of Bloom's Taxonomy, effectively demonstrates success in teaching quantitative knowledge to university students. While the assessment process can be improved to add precision, the results endorse the use of the cognitive categories to assess computer science related learning.

7. REFERENCES

[1] Anderson, L., Krathwohl, D., Airasian, P., Cruikshank, K., Mayer, R., Pintrich, P. Raths, J., and Wittrock, M. 2000. *Taxonomy for Learning, Teaching, and Assessing*. Pearson, New York.

[2] Bloom, B., Engelhart, M., Furst, E., Hill, W., and Krathwohl, D. 1956. *Taxonomy of Educational Objectives Handbook 1: Cognitive Domain*. Longman, New York.

[3] Brand, E. A., Honig, W. L., and Wojtowicz, M. 2011. Intelligent systems development in a non engineering curriculum. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (ITiCSE '11). ACM, New York, NY, USA, 48-52. DOI= 10.1145/1999747.1999764

[4] Gestwicki, P., and Ahmad, K. 2011. App inventor for Android with studio-based learning. *J. Comput. Sci. Coll.* 27, 1 (October 2011), 55-63.

[5] Gray, J., Abelson, H., Wolber, D., and Friend, M. 2012. Teaching CS principles with app inventor. In *Proceedings of the 50th Annual Southeast Regional Conference* (ACM-SE '12). ACM, New York, NY, USA, 405-406. DOI= 10.1145/2184512.2184628.

[6] Honig, W.L. 2012, Visual Programming BloomsAssessment, https://docs.google.com/file/d/0BwsTRjvLbRNOX3FOWkp6cUVnT3c/edit, Retrieved Dec. 10, 2012.

[7] Johnson, C. G., and Fuller, U. 2006. Is Bloom's Taxonomy appropriate for computer science? In *Proceedings of the 10th conference on computing education research,* ACM, New York, NY, 120-124. DOI= 10.1145/1315803.1315825.

[8] Mahmoud, Q. H., and Dyer, A. 2008. Mobile Devices in an Introductory Programming Course. *Computer* 41, 6 (June 2008), 108-107. DOI=10.1109/MC.2008.200

[9] Scott, T. 2003. Bloom's taxonomy applied to testing in computer science classes. *Journal of Computing Sciences in Colleges*. 19,1 (Oct. 2003), 267-274.

[10] Spertus, E., Chang, M. L., Gestwicki, P., and Wolber, D. 2010. Novel approaches to CS 0 with app inventor for android. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10). ACM, New York, NY, USA, 325-326. DOI= 10.1145/1734263.1734373.

[11] Thompson, E., Luston-Reilly, A., Whalley, J., Hu, M., and Robbins, P. 2008. Bloom's taxonomy for CS assessment. In *Proceedings of the 6th Baltic Sea conference on Australasian computing education (ACE '08),* Australian Computer Society, Darlinghurst, Australia, 155-161.

[12] Uludag, S., Karakus, M., and Turner, S. W. 2011. Implementing IT0/CS0 with scratch, app inventor forandroid, and lego mindstorms. In *Proceedings of the 2011 conference on Information technology education* (SIGITE '11). ACM, New York, NY, USA, 183-190. DOI= 10.1145/2047594.2047645

[13] Wolber, D. App Inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11),* ACM, New York, NY, 601-606. DOI= 10.1145/1953163.1953329.