Introducing Object Oriented Programming to Engineering Technology Students with an App Development Tool

Rolfe Josef Sassenfeld, Michael Morrell, Luke Nogales
Department of Engineering Technology
New Mexico State University
Las Cruces, New Mexico, United States
rolfe@nmsu.edu

***Abstract*—** Object oriented programming concepts are frequently a difficult topic for Engineering Technology educators to teach to students that have no previous object oriented programming experience. With the recent rise of mobile computing, a powerful and robust tool is now available to easily develop software for the Android mobile device operating system. Through "App" development with a highly interactive interface and real-time device feedback, difficult programming concepts are conveyed in a highly visual and tactile learning environment.

Keywords*-*Object Oriented Programming; App Development; Engineering Education; Engineering Technology

*****

I. INTRODUCING A NOVEL WAY TO TEACH OBJECT ORIENTED PROGRAMMING

Introducing object oriented computer programming (OOP) to Engineering Technology students who have never programmed is often difficult. Even those students who have programmed before in other non-object based languages are often apprehensive or completely lost when introduced to Java and related OOP languages. This apprehension inhibits the learning process and has shown a need for a more inviting development environment aside from the blank text document with which the student is presented. A modern mobile platform application (app) development tool offers an innovative and exciting approach to the introduction of many basic programming skills. This free software development tool called "MIT App Inventor" [1], [2], or simply "AppInventor" provides an inviting and lush development environment that provides the student with immediate visual, and at times, tactile feedback about their programming changes.

This highlights a paradigm shift from teaching OOP programming on a desktop computer versus a mobile electronics device such as an Android [3] smartphone. The Android phone is considered a smart device for the following reasons: the smartphone "knows" when you touch it, where it is located on the planet, how it is physically moving; knows where it's at, where it's, going, where it's been. It can see you, it can hear you, it can feel you, it can talk to you, and it can also "number crunch" just as fast as a personal computer. All of these sensing capabilities offer the programming instructor an abundance of object related concept examples. Each student's program will have a demonstrable working app that stimulates all the creativity of the senses.

II. A BRIEF HISTORY OF THE TOOL

MIT App Inventor was an open source software project sponsored by Google, and released to the public in December 2010. Hal Abelson and Mark Friedman from the Massachusetts Institute of Technology led the AppInventor team. In 2011 Google terminated its direct relationship with AppInventor and sponsored MIT to carry out the research and development of AppInventor program. MIT is host to annual summits to share the latest news and implementations of related tools and exceptional examples of apps created with AppInventor. MIT AppInventor is regularly updated and improved and has a knowledgeable user community that is an invaluable resource to students, developers, inventors, and entrepreneurs.

III. SIGNIFICANT DEVELOPMENTS FOR AI2

The AppInventor tool has been divided into two logical categories; the design aspect (layout, object creation, etc.) and the programming aspect of the event driven app. AppInventor therefore has two main views; the design view and the blocks editor. In the blocks editor the user defines reactions to events initiated by the user of the app. One of the more significant updates in the latest version "AppInventor 2" or simply "AI2" is the blocks editor.

The block editor is the most innovative part of the AppInventor tool. It allows a visual perspective of the programming relationship between objects and variables. As we will see in the examples below, the block editor offers the student a unique and logical view of the programming process. Many updates have been incorporated into AI2, but perhaps the most significant is the incorporation of the blocks editor into the actual user's browser. No longer a separate Java program the AI2 blocks editor is fully incorporated into the user's browser. This major improvement cannot be understated.

IV. HOW TO VISUALLY TEACH PROGRAMMING

Most students benefit from tactile involvement in the learning process. With the AI2 block editor the students get immediate visual feedback to their programming. The blocks ingeniously only allow relevant blocks to attach. So anything that is not allowed isn't even an option. This limitation to a menu of options provides the student with accuracy and helps to underscore the understanding of logical object relationships and actions. When a procedure is missing parameters block

 ISSN: 2321-8169
3875 – 3878

editor provides instant feedback in the form of a red X and a hover over description of the problem.

The block colors also have significance and help the student to put together relevant functionality. A separate color is assigned to procedures, mathematical operations, text operations, events, and more. This enables the learner to immediately associate relevant tasks and visually group logical operations.

Block programming has many benefits from a student's perspective [4]. It is colorful, offers instant comment, and allows students to group programming functionality spatially on a palette without restriction. The "clickable" blocks provide the student with immediate visual and aural feedback as to which blocks work and do not work. When moving blocks the user has the impression of holding or grasping the block. This promotes a highly interactive environment that aids the student tremendously with respect to concept retention, logical thinking, and originality. The interface invites experimentation and exploration. The fact that the student immediately sees the changes on the wirelessly connected mobile device facilitates iterative development and problem solving. The immediate reward factor offers learners positive reinforcement and a quick guidance [5]. The friendly and rich interface of AI2 offers innovators and inventors an easy way to tinker.

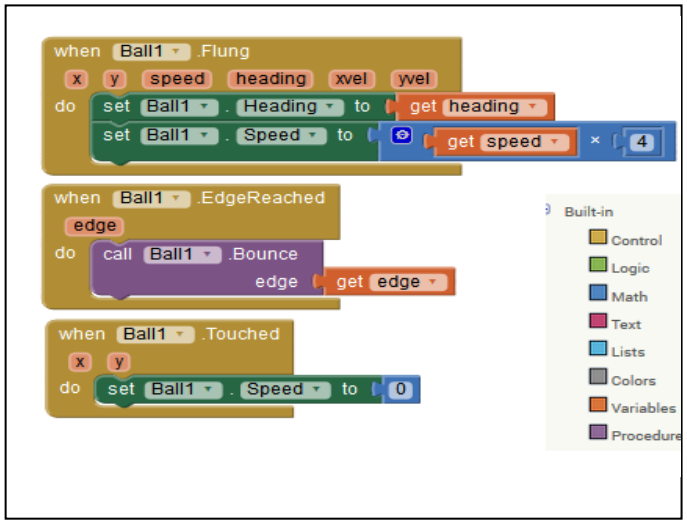

Figure 1. Control blocks for handling the motion of an on screen object.

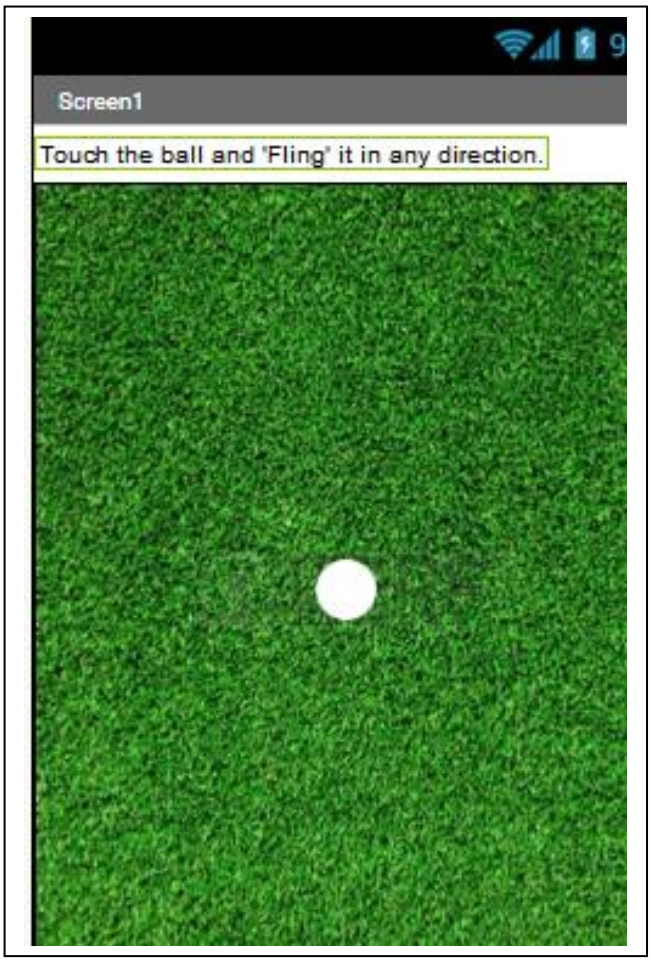

Figure 2. Example of the built n Android App emulator.

V. WHAT ABOUT 'REAL' PROGRAMMING AKA JAVA?

A common criticism of AppInventor is that it isn't very customizable. The nature of the block programming inherently leads the user at some point to reach for a block that doesn't exist. Therefore the objection is valid since there is currently no easy way to create purely custom blocks. This is one of the tradeoffs with the block programming method. For the ease of use, you lose some flexibility. There are numerous ingenious ways around these limitations. For instance, although AI2 has a long list of mathematical operators, it can be cumbersome at best and impossible at worst to perform some mathematical operations or to arrange a functional expression. However, an elegant work around is to use the remote server/scripting functionality of Ai2 to perform all the heavy lifting at the server and use the app as an interface. This approach is actually becoming the most common type of app. One in which data is processed remotely and only the display of data is performed locally. This is really a prevalent operating mode on the most popular apps from Facebook to Netflix to Amazon etc. The app is only an interface to a server providing data or content back to the user's device via the app interface.

The previous example notwithstanding, there is demand for an export function in AppInventor to allow a conversion of the app to the Java/Eclipse development environment. This functionality would allow entrepreneurs that develop

AppInventor prototype apps in to move readily more their project into a larger Java development environment. It would also very much facilitate the teaching of Java by allowing instructors to demonstrate Java concepts visually with the block editor. There have been efforts at a separate tool to perform these functions, however a direct export integrated into Ai2 would be simplest from an educator's standpoint. There is currently a working Google group that is prototyping a process for converting AppInventor to JAVA. However, this conversion process is cumbersome and still very inaccurate with respect to converting object relationships from one language to the other.

VI. BENEFITS TO ENGINEERING TECHNOLOGY EDUCATION

- Faster quicker grasp of complex programming paradigm.
- Can be modified "on the fly" to provide the student with immediate live feedback to their code modifications.
- Tactile and visual
- High quality, no cost teaching resources available.
- Active development community.
- Easily integration of entrepreneurship monetary incentive to students.

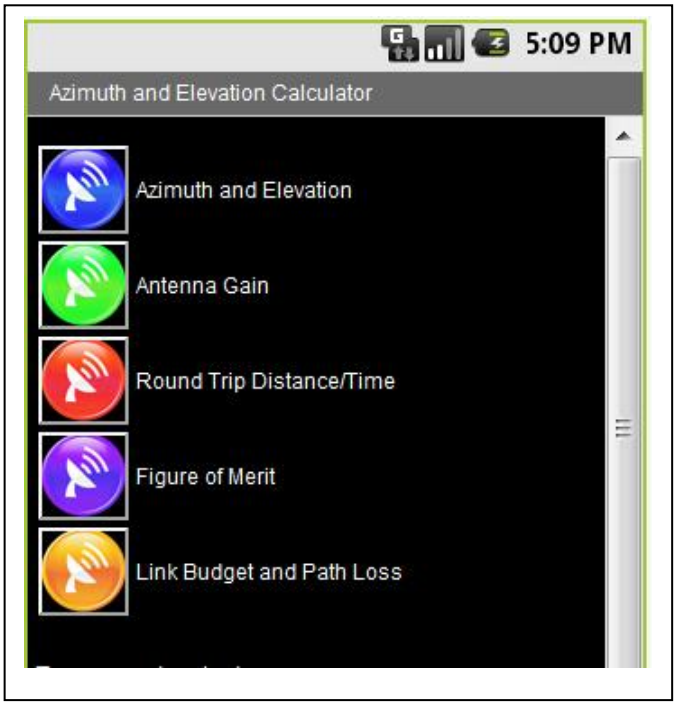

Figure 3. Touch menu interface example.

VII. CONCLUSIONS

The AppInventor development tool has become more robust and offers educators a highly stimulating learning environment to teach OOP computer-programming concepts to their students.

The portability of the platform provides for an ever-increasing audience. With the continued support of MIT AI2 offers a great long-term potential to improve engineering technology's programming needs. The immediate and often tactile feedback of working with a mobile device provides students with an immersive programming experience with multi-sensory feedback [6] Whether you are teaching Computer Science students, technologists, or hobbyists; the visual block editor and immersive development environment of AppInventor merits consideration.

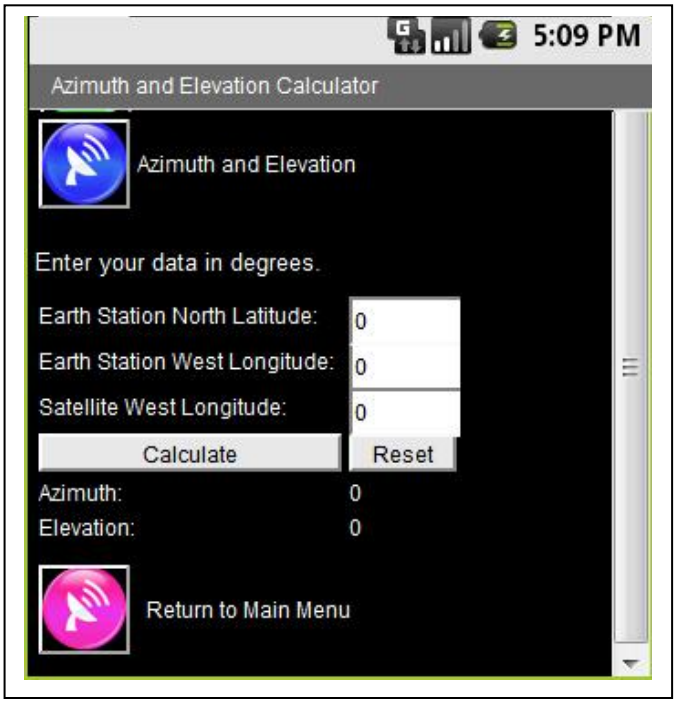

Figure 4. Example of simple scientific calculator interface.

ACKNOWLEDGMENT

The authors would like to acknowledge the work of the MIT Mobile Computing Laboratory and the AppInventor Development team. We would also like to acknowledge Dr. Jeffrey Beasley our department head for his support of this research. We also offer thanks and gratitude to our fathers, Dr. Helmut Sassenfeld Sr., Malcolm Morrell, and John Nogales.

REFERENCES

[1] Abelson, H. (2011). *MIT App Inventor*. (M. I. Technology, Producer) Retrieved 2011, from MIT App Inventor | Explore MIT App Inventor: http://appinventor.mit.edu/explore/

[2] Wikipedia. (2014). *Hal Abelson - Wikipedia, the free encyclopedia*. (Wikedpedia, Producer) Retrieved from Wikipedia, the free encyclospedia.: http://en.wikipedia.org/wiki/Hal_Abelson

[3] Google. (2014). *Android*. (I. GOOGLE, Producer, & Google, Inc,) Retrieved from Android: http://www.android.com

[4] Fischer, G. B. (1992). Adding rule-based reasoning to a demonstrational interface builder. *Proceedings of ACM Symposium on User Interface Software and Technology (UIST '92), Monterey, CA* (pp. 89-97). New York: ACM Press.

[5] Goldberg, A. (.-W. (1984). *Smalltalk-80: The interactive programming environment*. (Vol. 1). Reading, MA: Addison-Wesley.

[6] Stephanidis, C. (2001). User Interfaces for All: New perspectives into Human-Computer Interaction. Mahwah, NJ, USA: Lawrence Erlbaum Associates.

ISSN: 2321-8169
3875 – 3878

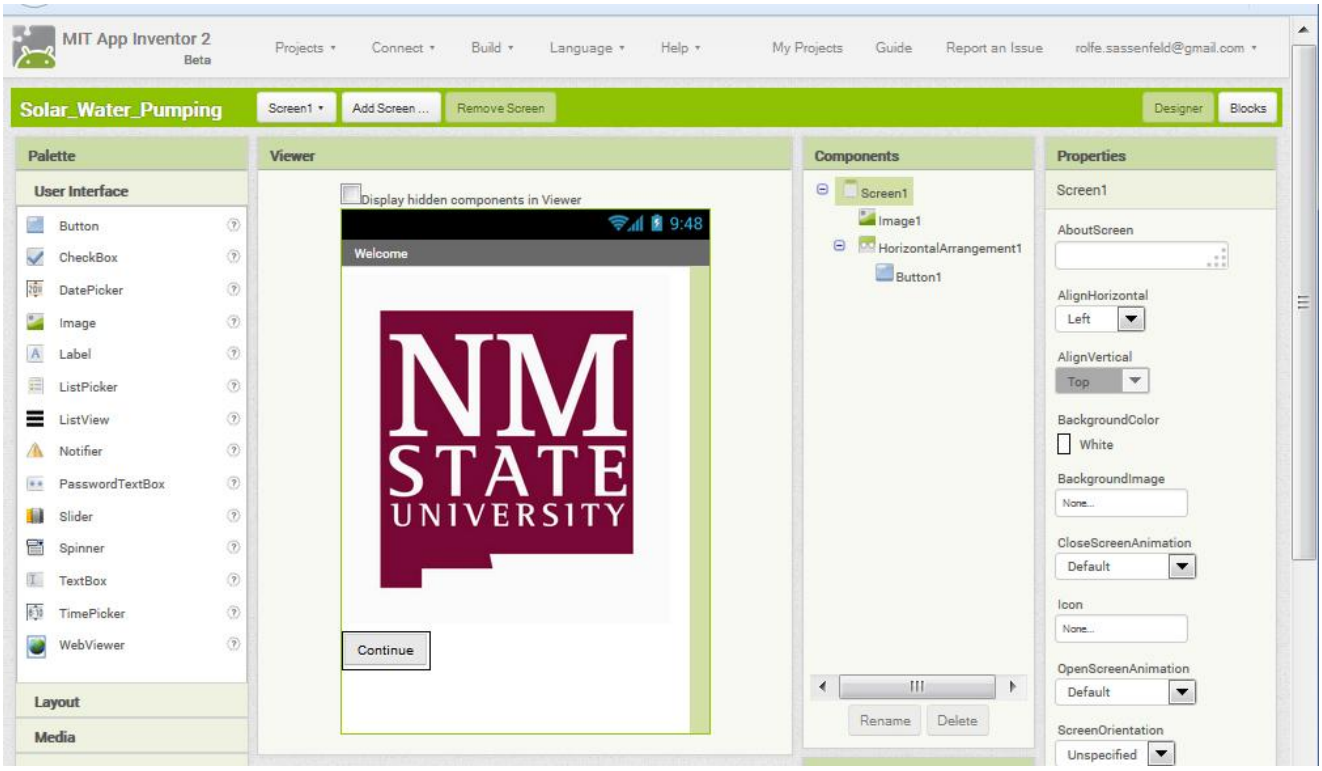

Figure 5. Example of a the AppInvewntor designer screen layout.

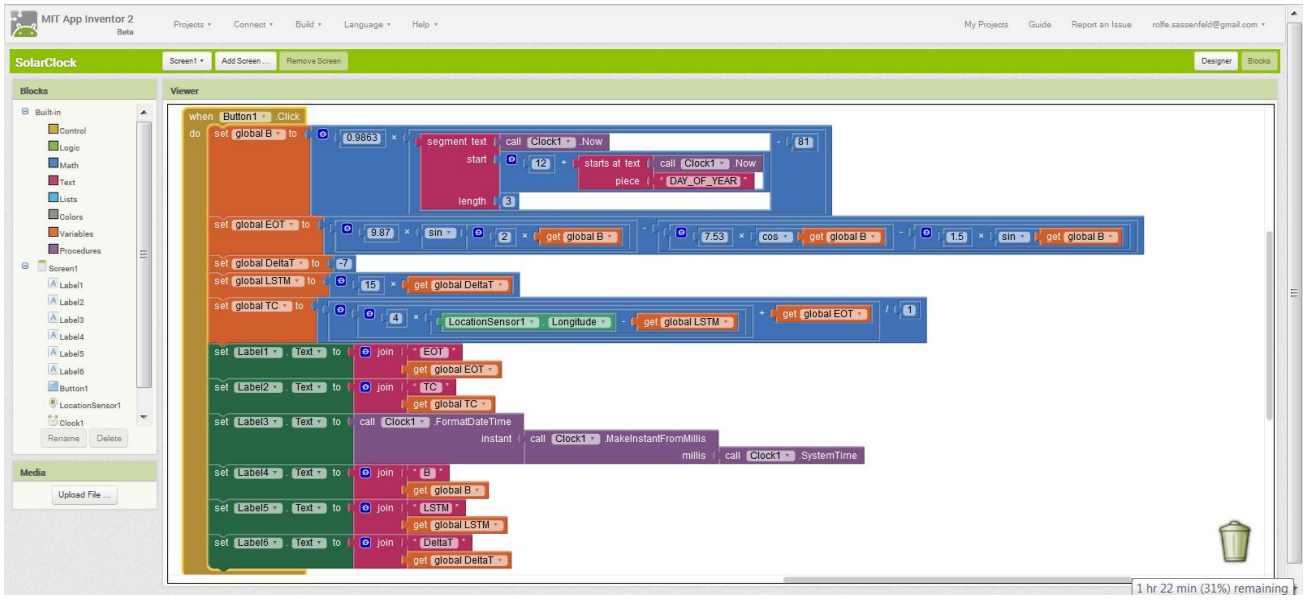

Figure 6. Complex event hadnling using block programming.