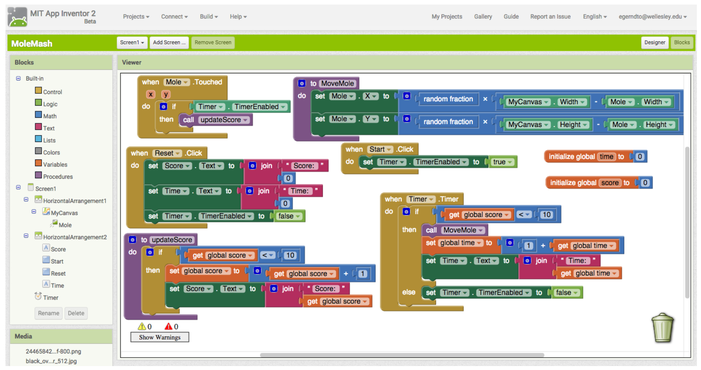
Improving App Inventor's Debugging Features

Emery Gerndt Otopalik '16

Advisor: Franklyn Turbak, Wellesley College Computer Science Department

App Inventor Background

MIT App Inventor is a blocks-based programming language for creating mobile Android applications. App Inventor democratizes mobile app development by providing an intuitive visual interface that anyone can learn. So far, nearly 3 million App Inventor users have become creators, building around 7 million apps. An App Inventor app has two parts, components and block-based code. Components are elements like buttons, labels, sensors, or cameras. Blocks are plug-and-socket puzzle pieces that can be clicked together to specify the behavior of the app.



Sample program in the Blocks Editor

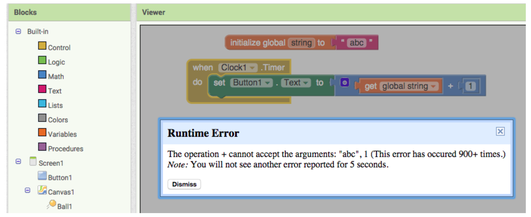
App Inventor also has a live-development feature that allows the user to test an app while building it in the blocks editor on a web browser. The blocks editor converts the blocks into code that can be executed on an Android device. When the device receives the code, it runs it and sends information about the execution back to the editor. This information needs to be formatted in a manner meaningful to the programmer, which is where my work comes in.

Project Overview

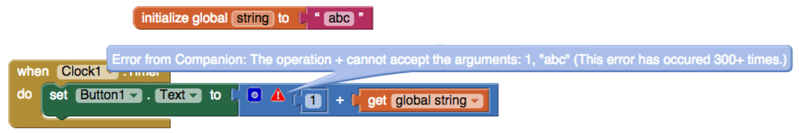
The visual design of App Inventor does help users assemble applications, but it does not help users understand the runtime behavior of their program or troubleshoot programs that do not work. By expanding on work done by Johanna Okerlund '14, I have implemented several features that will help users in these two areas. I have enhanced her (1) errors on blocks and (2) *Watch* features, as well as improved the (3) displayed notation for App Inventor values. All of these features will assist the user in understanding how programs work and in identifying, locating, and fixing errors.

Errors on Blocks

Previously, runtime error messages generated by the app on a device appeared in a pop-up window in the blocks editor that provided little information about the offending block(s).



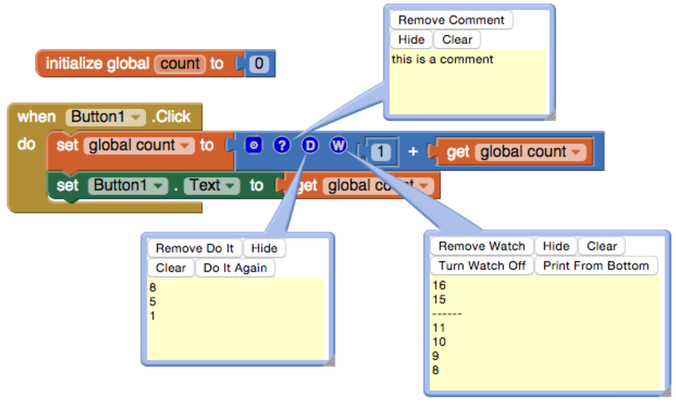
My improved errors-on-blocks feature puts an error message directly on the block responsible for the runtime error. Knowing the source of the error saves the programmer valuable time. Once the error is fixed, the message is removed.



Users can see a running total of how many times an error has occurred. With Johanna's implementation, errors that fire rapidly resulted in the device crashing due to the additional block information being sent to the editor. This major issue was resolved.

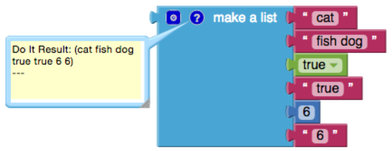
Watch and Icons

App Inventor's existing *Do It* feature shows the value of a block in a bubble connected to the block. Johanna Okerlund added *Watch* feature that prints in a bubble all the values of a block during program execution. Before, *Do It* and *Watch,* as well as *Comments* all used the same bubble. I improved this interface by giving each function its own bubble and adding buttons specific to each bubble.

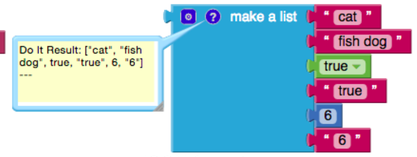


JSON for App Inventor Values

Before, values were displayed using a parenthesized notation that originated from the underlying Scheme implementation of App Inventor. Lists were particularly confusing; it was impossible to tell the lengths or types of elements in the list. I changed the value notation to standard JSON format, which wraps quotes around each string and separates each list element by a comma.

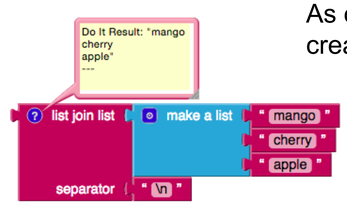


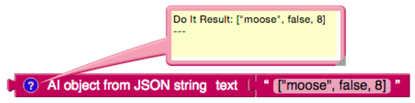
Scheme-based format



JSON format

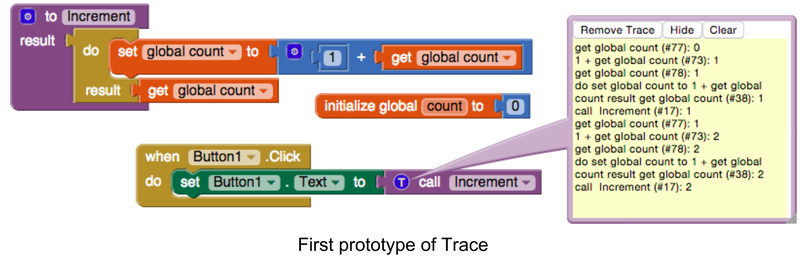
I also added two new blocks that relate to JSON and lists. The first (left) block allows a user to parse a list into a string with a separator between elements. The second (right) allows a user to convert a JSON string representation of an object into an App Inventor object. As of now, users are only allowed to create lists from JSON arrays.





Future Work

My priority is to make sure all of these features are ready for release to users this coming fall. This will require feedback from other App Inventor developers as well as more testing. I am also working on a *Trace* feature that displays the values of executed blocks in one location.



First prototype of Trace

My next big project will be implementing a deterministic replay debugger that visualizes the step-by-step execution of programs.

Funded by National Science Foundation Grant DUE-1226216. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.