Improving App Inventor Usability via Conversion between Blocks and Text

Karishma Chadha[a], Franklyn Turbak[b,1]

[a] *MIT Lincoln Lab, Lexington, MA 02421*
[b] *Computer Science Department, Wellesley College, Wellesley MA, USA 02481*

Abstract

We have developed TAIL, a textual programming language isomorphic to the blocks language of MIT App Inventor (AI), and have extended AI with code blocks, a novel mechanism that enables bidirectional conversions between blocks and text fragments. TAIL improves AI's usability by facilitating the reading, writing, and sharing of programs, and may also ease the transition from blocks to text programming

In blocks languages, programs are constructed by connecting visual fragments (blocks) shaped like jigsaw puzzle pieces. MIT App Inventor (AI) [1] democratizes programming for Android mobile apps through its easy-to-use blocks language. Several features of AI reduce syntactic and cognitive frustrations experienced by novices when learning textual programming: blocks are chosen from menus of related blocks; the shapes of plugs and sockets suggest how the blocks fit together; and labels on sockets document their purpose.

While simple AI blocks programs are easy to read and write, more complex ones can become overwhelming. Creating and navigating nontrivial blocks programs is tedious, and AI's inability to copy blocks between projects inhibits reusing code between projects and between programmers.

To address these issues, we created TAIL, a Textual App Inventor Language isomorphic to AI's blocks language, and provided a means for converting between them. We extended AI with *code blocks* in which TAIL code for AI expressions, statements, and top-level declarations can be entered. These code blocks are interconvertible with the AI blocks they represent (Fig. 1).

Ours is the first bidirectional isomorphic conversion system between blocks and text languages. This distinguishes it from systems that convert blocks to text, but not vice versa (e.g., [2, 3]), and from PicoBlocks [4], where blocks can be defined in a text language more expressive than its blocks language, but blocks cannot be converted to text. Bidirectional conversions (1) increase AI's usability by providing an efficient means for reading, writing, and reusing/sharing programs, and (2) may ease the transition from blocks to text programming.

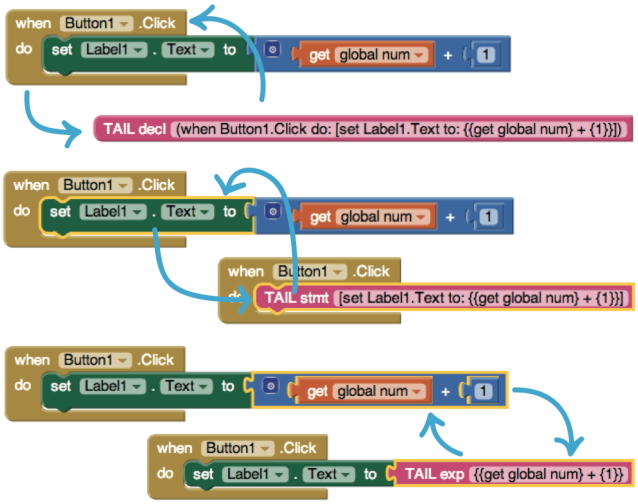

Figure 1: Sample TAIL ⇔ blocks conversions.

This is work in progress. The TAIL language, code blocks, the implementation of the bidirectional conversion between TAIL and AI blocks, and related work are described in detail in [5].

References

[1] MIT App Inventor home page, `http://appinventor.mit.edu`, accessed Sep. 20, 2014.
[2] N. Fraser, Blockly code demo, `https://blockly-demo.appspot.com/static/apps/code/index.html`, accessed Sep. 20, 2014.
[3] P. Guo, Proposal to render Android App Inventor visual code blocks as pseudo-Python code, `http://people.csail.mit.edu/pgbovine/android_to_python`, accessed Sep. 20, 2014.
[4] Playful Invention Company, PicoCricket Reference Guide, v. 1.2a, `http://www.picocricket.com/pdfs/Reference_Guide_V1_2a.pdf`, accessed Sep. 20, 2014.
[5] K. Chadha, Improving the usability of App Inventor through conversion between blocks and text, Undergraduate thesis, Wellesley College, May, 2014.

Email addresses: `karishma.chadha@ll.mit.edu` (Karishma Chadha), `fturbak@wellesley.edu` (Franklyn Turbak)

[1] This work was supported by the National Science Foundation under grant DUE-1226216 and by sabbatical funding from Wellesley College