Improving App Inventor Usability via Conversion between Blocks and Text

Karishma Chadha
MIT Lincoln Lab, Lexington, MA 02421, USA
Email: karishma.chadha@ll.mit.edu

Franklyn Turbak
Wellesley College, Wellesley, MA 02481, USA
Email: fturbak@wellesley.edu

Abstract

We have developed TAIL, a textual programming language isomorphic to the blocks language of MIT App Inventor 2 (AI2), and have extended AI2 with code blocks, a novel mechanism that enables bidirectional isomorphic conversions between blocks and text program fragments. TAIL improves AI2’s usability by facilitating the reading, writing, and sharing of programs, and may also ease the transition from blocks programming to text programming.

1. Project Overview

In blocks programming languages, programs are constructed by connecting visual fragments (blocks) shaped like jigsaw puzzle pieces. MIT App Inventor 2 (AI2) [7], a popular online environment for Android app development, democratizes mobile programming through its easy-to-use blocks language. For example, the AI2 blocks shown in Figure 1 are a No Texting While Driving app that automatically replies to an incoming text message with a canned response and then speaks the phone number and content of the message. The shapes of plugs and sockets on the blocks distinguish expressions and statements and suggest how the blocks fit together, reducing syntactic frustrations experienced by novices when learning textual programming. Sockets have labels documenting their purpose, helping programmers remember the number and order of operands. Selecting a block from a menu of related blocks (e.g. math blocks, control blocks) can be easier than remembering its name. Other blocks languages include Scratch [6, 8], Blockly [4], StarLogo Nova [12], and PicoBlocks [11].

While simple AI2 blocks programs are easy to read and write, more complex ones can become overwhelming. Creating and navigating nontrivial blocks programs is tedious, and AI2’s inability to copy blocks between projects inhibits reusing code between projects and between programmers.

To address these issues, we have created a new textual language, TAIL (the Textual App Inventor Language), that is isomorphic to AI2’s blocks language and have provided a means for converting between them. TAIL syntax is designed to provide users with a systematic way to translate the visual information on the blocks into text. We have extended AI2 with code blocks in which users can type TAIL code representing AI2 expression, statement, and top-level declaration blocks. These code blocks have the same meaning as the larger block assemblies they represent. Programmers can also convert back and forth between these code blocks and the original AI2 blocks (Fig. 2). Language isomorphism guarantees that round-trip conversions (from text to blocks and back, or blocks to text and back) begin and end with the identical program.

Ours is the first blocks language to support isomorphic conversions between blocks and text languages in both directions. This distinguishes it from systems that can convert blocks to text programs, but not vice versa (e.g. [1, 3, 5, 9] and from PicoBlocks, which can define blocks in a text language more expressive than its blocks language but cannot convert blocks to text [11]. Bidirectional conversions
(1) increase AI2’s usability by providing an efficient means for reading, writing, and reusing/sharing programs, and (2) may ease users’ transitions from blocks programming to text programming.

The TAIL language is implemented using an ANTLR parser generator [10] to generate a JavaScript lexer and parser for TAIL. Actions in the grammar are used to translate the TAIL parse tree into AI2’s XML tree representation for blocks. These and other details are explained in [2].

This is work in progress. Currently, TAIL handles only a subset of AI2 blocks. We are extending TAIL to handle the full AI2 language and to express whole programs in addition to program fragments. We also need to test the usability of TAIL and code blocks in user studies so that we can refine them before including them in a release of AI2.

2. Why is this Research of Interest?

This research is of interest to the VLC/DMS community because App Inventor is becoming an increasingly popular visual language for creating mobile apps and teaching programming. It is a step towards addressing common criticisms of App Inventor: that large programs are difficult to read and write; that program fragments cannot be reused between programs; and that there is no clear transition between blocks programming and text-based programming.

3. Nature of the Demo & Poster

Our presentation will have two parts: (1) a poster that summarizes the TAIL language and the mechanisms for converting between TAIL textual program fragments and App Inventor 2 blocks; and (2) an interactive demo of the conversion between TAIL and blocks in a version of App Inventor 2 running on a laptop computer.

Acknowledgments

This work was supported by the National Science Foundation under grant DUE-1226216, by sabbatical funding from Wellesley College, and by the Wellesley College Science Center Summer Research program.

References