Amphibian: Dual-Modality Representation in Integrated Development Environments

Jeremiah Blanchard
Department of CISE
University of Florida
Gainesville, FL, USA
jjb@eng.ufl.edu

Chistina Gardner-McCune
Department of CISE
University of Florida
Gainesville, FL, USA
gmccune@ufl.edu

Lisa Anthony
Department of CISE
University of Florida
Gainesville, FL, USA
lanthony@cise.ufl.edu

***Abstract*— A significant challenge to using dual-modality environments in instruction is that existing dual-modality representations are built into sandbox environments with functionality tailored to a specific purpose (such as Pencil Code's turtle graphics). However, students in introductory programming classes usually use an Integrated Development Environment (IDE), a software application that provides a suite of tools for programming support. The use of IDEs is common in industry, and thus they bring additional authenticity and scaffolding to the learning experience. We developed an IDE plugin, Amphibian, from Pencil Code's Droplet Editor to enable switching between blocks and text within IntelliJ IDEA, a common IDE for the Java language. This plugin allows teachers of Java courses, including those of AP CS and in colleges, to incorporate dual-modality representation into their curriculum. It also enables more rigorous research into use of dual-modality environments in classrooms by facilitating use within existing curricula, allowing researchers to remove potentially confounding variables, such as use of different languages, software systems, and development environments.**

Keywords—Computer Science Education, blocks-based programming environments, programming languages, novice programmers, dual-modality representation, integrated development environments, IDE, Amphibian.

I. Introduction

A significant challenge to using dual-modality representation in instruction is that the dual-modality tools have been built into sandbox environments with functionality tailored to a specific purpose. For example, Tiled Grace and Pencil Code are two website-based environments that allow students to program in the browser without any additional tools, but programs are limited to their custom turtle-graphics sandbox features [1], [2]; users cannot use other standard or third-party libraries and features. However, students in introductory programming classes at the college level usually use an Integrated Development Environment (IDE) which provides a suite of tools for programming support, including integration of standard language libraries. The use of IDEs is common in industry, and thus they bring additional authenticity to the learning experience. As such, we could facilitate instruction and research via dual-modality representations in existing college-level curricula by integrating dual-modality tools within these general-purpose development environments.

At the time we began our work, there were no dual-modality tools for standalone IDE-based development outside of tailored sandbox environments, so we developed a plugin for IntelliJ IDEA based on Pencil Code's online open-source Droplet Editor [3], [4]. Matsuzawa et al. previously developed a blocks-text tool for a subset of the Java language, but this was also limited to a turtle-graphics environment [5]. The plugin that we developed, which we dubbed Amphibian [6], enables instructors to more easily incorporate dual-modality instruction into courses and enables more rigorous investigation of dual-modality representations in classrooms by allowing researchers to reduce other potentially confounding variables, such as use of different languages, software systems, and development environments.

The Droplet Editor's extensibility allowed us to integrate the language of our choice into Amphibian. We noted that many introductory computer science programs at the high school and college levels, including those at our institution, use Java as the target language. To facilitate practical study of CS1 student performance in a "real-world" environment, we focused development on a Java variant. Amphibian allows users to switch back and forth between text and blocks modes, thereby enabling teachers of Java courses, including those of AP CS and many introductory college courses, to build blocks/text transitions into curricula.

II. Using The Amphibian Plugin

Amphibian uses IntelliJ's plugin API and can be installed in the same manner as other plugins. Once installed, Amphibian adds two tabs to the bottom of the editor pane of any Java file (**Figure 1a**). The tabs allow users to switch between the text of a program (Text Mode), which is the default mode upon startup, and its blocks representation (Blocks Mode), and back again.

In Text Mode, the editor retains all features of the IDE's text editor, including syntax highlighting, prediction, error identification, recommendations, and code region identification.

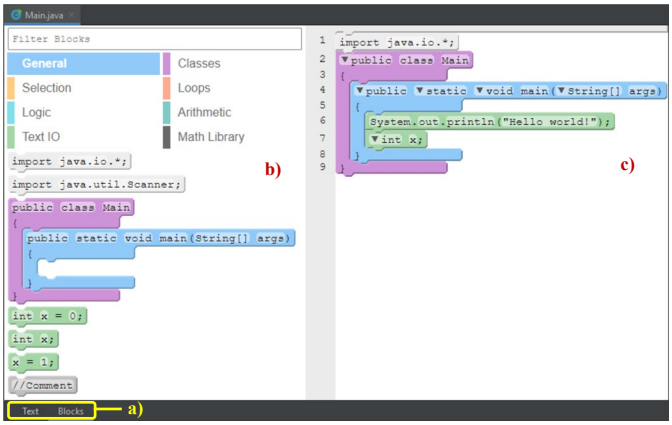

Figure 1. Amphibian Blocks Mode editor showing a) tabs for switching between modes, b) blocks representation of the current program, and c) block toolbox from which users can drag and drop constructs.

978-1-7281-0810-0/19/$31.00 ©2019 IEEE

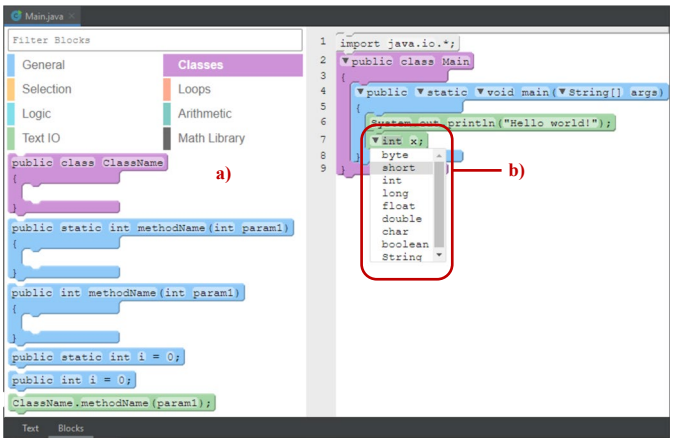

Figure 2. Amphibian Blocks Mode editor showing a) Java object-oriented constructs and b) drop-down menus used for types and modifiers.

When the "Blocks" tab is selected, the editor switches to Blocks Mode, which uses the Droplet Editor to present a toolbox from which blocks can be dragged to add them to the program (**Figure 1b**) as well as display and enable editing of blocks-based constructs (**Figure 1c**). When in Blocks Mode, the program can be modified by adding new blocks to the program, with correct constructions signified by the puzzle piece style snap-together construction often used in blocks-based environments. Text in light-colored areas may be edited directly; in the case of variable value assignment, users may also drag-and-drop blocks representing variables / objects. At any time, a user can change modes using the same tabs.

To facilitate Java programming specifically, we added object-oriented blocks, including classes and methods (**Figure 2a**), while access modifiers such as "public" and "private" can be selected from dropdown components on the blocks themselves. Similarly, built-in variable types for parameters of variables can be selected from a dropdown menu on the blocks (**Figure 2b**), and users can enter text for custom and imported types. Whenever a block is added to the program via the drag-and-drop interface, the embedded Droplet Editor variant adds the construct to the program's text and its blocks-based representation in real-time.

It is important to note that, as the plugin only changes the interface for editing the program, all IDE features remain available. Users can follow the typical workflow to build and run programs, including developing and running unit tests. Any Java project can be used with the plugin, including typical text-based and graphical applications, Android apps, and libraries.

III. Architecture

Amphibian was developed in two distinct phases. In one, we incorporated the Java language into Droplet, and in another, we developed the plugin into which we embedded Droplet.

A. Droplet Editor

To enable Droplet to process Java language constructs, we integrated a customized Java language parser. To do so we constructed a custom variant of the Java 9 grammar specification and used ANTLR [7] to generate a parser program. Once the parser was in place, we developed a Droplet "palette" – a set of blocks-text mappings – for Java language constructs, including control structures, common statements, and object-oriented constructs such as classes and methods.

B. IntelliJ Plugin

The plugin connects to two major IntelliJ systems: the User Interface (UI) and the Document Manager (**Figure 3**). Whenever a Java file is opened, Amphibian adds the "Blocks" and "Text" tabs to the standard text editor. At the same time, in the background the blocks editor is loaded. This is accomplished by embedding a browser component via JxBrowser [8], which is preloaded with the Droplet Editor variant and custom JavaScript files, that can receive notifications from the plugin. When the user is in Text Mode and the "Blocks" tab is selected, an event is sent to the Droplet Editor which includes the current document text state. The text is loaded and processed, after which the embedded browser is displayed in the UI. The Java parser can interpret incomplete programs as blocks even when some constructs are missing. However, if the text syntax cannot be parsed due to irrecoverable errors, such as missing brackets, a modal dialog is shown to the user indicating the syntax error and directing the user to fix it in text mode. Otherwise, the browser editor window is shown, and user can edit the program using the blocks interface.

Any time the toolbox palette changes or a block is dragged or dropped, the event is sent to the log. This log entry by default is displayed in the console, but the plugin can be configured to forward the message to a remote server. In addition, whenever the program is changed, the updated text is sent to the IntelliJ Document Manager. This ensures that the program text is synchronized between Blocks Mode and Text Mode (the standard IDE text editor). In addition, this means that there is always a text representation of the blocks; incomplete programs will not prevent conversion from blocks to text. When the text tab is selected from within Blocks Mode, the current text state is sent one last time to the IntelliJ Document Manager and the display is changed back to the default text editor for the IDE.

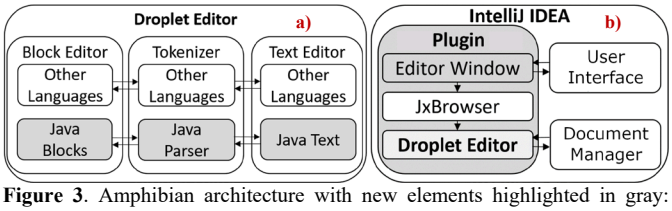

Figure 3. Amphibian architecture with new elements highlighted in gray: a) Modifications to the Droplet Editor and b) Architecture of the IntelliJ Plugin.

IV. Future Work

Future work on the Amphibian plugin will include adding support for more programming languages, which is easily enabled by the Droplet Editor's extensible architecture. In particular, some institutions and curricula have moved to Python as a first text language [9], [10], and in our prior work we added a Python variant to Pencil Code [11]. As such, we intend to integrate the Python language into the plugin and potentially other languages as well. We have also completed prototype work on a Visual Studio variant of Amphibian, which we intend to release after a production variant is completed. Finally, we are using Amphibian to study the use of dual-modality representations in Computer Science Education, especially its effect on learner perceptions and performance among those learning to program for the first time.

Acknowledgements

We extend our thanks to Jackson Yelenik, Benjamin King, and Trevor Lory who developed parts of the plugin architecture during their capstone project as undergraduate students.

References

[1] M. Homer and J. Noble, "Lessons in combining block-based and textual programming," *J. Vis. Lang. Sentient Syst.*, vol. 3, no. 1, pp. 22–39, 2017.

[2] D. A. Bau, D. A. Bau, C. S. Pickens, and M. Dawson, "Pencil Code: Block Code for a Text World," in *Proceedings of the 14th International Conference on Interaction Design and Children*, 2015, pp. 445–448.

[3] "IntelliJ IDEA." [Online]. Available: https://www.jetbrains.com/idea/.

[4] D. Bau, "Droplet, a blocks-based editor for text code," *J. Comput. Sci. Coll.*, vol. 30, no. 6, pp. 138–144, 2015.

[5] Y. Matsuzawa, T. Ohata, M. Sugiura, and S. Sakai, "Language Migration in non-CS Introductory Programming through Mutual Language Translation Environment Basic Function and Interface," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 185–190.

[6] "Amphibian Plugin," 2019. [Online]. Available: https://github.com/cacticouncil/amphibian.

[7] T. J. Parr and R. W. Quong, "ANTLR: A predicated-LL(k) parser generator," *Softw. Pract. Exp.*, vol. 25, no. 7, pp. 789–810, 1995.

[8] "JxBrowser." [Online]. Available: https://www.teamdev.com/jxbrowser.

[9] R. J. Enbody, W. F. Punch, and M. McCullen, "Python CS1 as preparation for C++ CS2," *ACM SIGCSE Bull.*, vol. 41, no. 1, pp. 116–120, 2009.

[10] D. Garcia, "CS10: The Beauty and Joy of Computing," 2019. [Online]. Available: https://cs10.org/fa19/.

[11] J. Blanchard, C. Gardner-McCune, and L. Anthony, "Effects of Code Representation on Student Perceptions and Attitudes Toward Programming," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2019, pp. 127–131.