Blocks, Text, and the Space Between

The Role of Representations in Novice Programming Environments

David Weintrop
Learning Sciences, Northwestern University
dweintrop@u.northwestern.edu

I. Introduction & Motivation

A long-standing question faced by educators when introducing learners to programming is deciding where to start on day one: What programming language to choose? In what environment? An increasingly popular approach to the design of introductory programming tools is the use of graphical, blocks-based programming environments that leverage a primitives-as-puzzle-pieces metaphor. In such environments, learners can assemble functioning programs using only a mouse by snapping together instructions and receive visual feedback on how and where commands can be used and if a given construction is valid. The use of this programming modality has become a common feature of introductory computer science curricula and programming interventions targeted at young learners. Notably, national curricular efforts including Exploring Computer Science, the CS Principles project, and Code.org's course materials utilize blocks-based tools to introduce students to programming.

Despite its growing popularity, open questions remain surrounding the effectiveness of blocks-based programming for helping students learn basic programming concepts and the overall suitability of the approach for preparing students for future computer science learning opportunities. Further, it is unclear what the strengths and weaknesses of block-based programming tools are compared to isomorphic text-based alternatives. The goal of this dissertation is to shed light on these two open questions and then use that knowledge to design a new tool for learning to program that draws on the strengths of both blocks-based and text-based modalities. To answer these questions we are conducting a study comparing blocks-based, text-based, and hybrid block/text programming tools along side each other in high school introductory programming classes. By doing so, we seek to provide evidence to better inform educators and administrators who are tasked with making consequential decisions around how learners are introduced to computer science and, more generally, to contribute to the larger understanding of the relationship between programming representations and the understandings and practices they promote.

II. Research Questions

The proposed study seeks to answer three sets of interrelated research questions. The first set pertains to the effects of programming modalities on students' understandings of programming concepts and the programming practices they engender. The second set of questions looks at the effectiveness of introductory programming tools for preparing students for future computer science learning opportunities. While blocks-based programming environments have been found to be successful at engaging students in programming activities and providing early success with little or no formal instruction, educators have had difficulty transitioning learners from these graphical environments to conventional text-based programming environments. The final set of questions look at the design space between blocks-based and text-based programming approaches, asking: Can we design hybrid introductory programming environments that blend features of blocks-based and text-based programming that effectively introduce novices to programming and computer science more broadly? Our goal for this final question is to develop an environment we can confidently advocate for classroom use or at least advance as a set of design principles for designing effective and engaging hybrid programming tools.

III. Research Design

To answer our questions, we have designed a mixed-method, quasi-experimental study that will take place in high school computer science classrooms. The study will occur over the first 15 weeks of the school year in three sections of the same Introduction to Programming class. For the first five weeks of the course, each of the three classes will use a different environment, either a blocks-based environment (Fig. 1A), a text-based environment (Fig. 1B), or a hybrid blocks/text programming environment (Fig. 1C), built on the Pencil Code environment [1]. All three classes will work through the same curriculum in their respective environments. The five-week introduction will cover fundamental programming concepts including variables, procedures, looping, and conditional logic. Starting in week six, students will transition to Java, the language they will be using for the remainder of the school year. We will follow students for the first 10 weeks of Java instruction.

(A) Blocks-based (B) Text-based (C) Hybrid

Fig. 1. The three environments that will be used in the research study.

A. Data Collection

As we are pursuing a mixed-methods approach, a variety of data will be collected for the study. Attitudinal surveys and

978-1-4673-7457-6/15/$31.00 ©2015 IEEE

specially designed content assessments will be administered three times during the study: at the outset (beginning of week 1), at the midpoint when students transition to Java (end of week 5), and at the conclusion of the study (end of week 15).

Along with our quantitative data, we will also conduct semi-structured clinical interviews throughout the study. During these interviews, students will be asked to reason through problems that employ fundamental programming constructs, read existing programs written in various modalities, as well as author new programs in both blocks and text. These interviews are designed to elucidate student thinking about programming concepts and shed light on if and how modality is shaping emerging understanding. We will also conduct teacher interviews throughout the study and do observations of all three classrooms. Finally, we will record all of the student-authored programs over the course of the study. This includes programs written across the three introductory tools as well as Java programs authored in the subsequent weeks. The programs will be used as a data corpus for analyses using educational data mining techniques.

IV. Pilot Study

This fall we conducted a pilot study in a selective enrollment, urban public high school. The study lasted 10 weeks (a five week introduction and five weeks of Java) and included 90 students. The three classes used different modified versions of the Snap! programming environment [2]. The first class served as a control and used an unmodified version of Snap!, so never had any text-based programming interactions during the first five weeks. The second class used a version of Snap! with the ability to right-click on any block or script and open up a window showing a JavaScript implementation of the selected command(s). The third class used a version of Snap! that allowed them to read their programs in text, and added the ability to define the behavior of new custom blocks in JavaScript. Over the course of the ten-week pilot study we gathered 88 sets of pre/mid/post attitudinal assessments and over 8,500 responses to our content assessments. We also conducted 32 interviews (9 pre, 10 mid, 8 post, 5 teacher) and collected over 73,000 Snap! programs and over 10,000 Java programs. Data analysis is ongoing, below we report on some emerging findings from this study.

V. Emerging Findings

To date we have conducted a number of analyses of our pilot data. Our first investigation was into the question of how students perceive the differences between blocks-based and text-based programming. This analysis drew on data collected in the mid and post attitudinal surveys and was supplemented by the interviews we conducted. We found that a majority of students (92%) viewed blocks-based programming as easier than text-based programming. Students cited a number of features of blocks-based programming for this including the ease of the drag-and-drop composition, the lack of needing to memorize commands, and fact that in the tools we used, blocks were closer to natural language than text-based programs, making them easier to read. Students also identified drawbacks of the blocks-based programming approach, including issues of authenticity and the difficulty of building large complex programs with block-based tools [3].

We have also analyzed student responses to our Commutative Assessment [4]. In particular we investigated the relationship between modality and programming concepts. Students were asked to answer questions about four basic programming concepts: variables, iterative logic, conditional logic, and procedures, along with program comprehension questions in both blocks-based and textual programs. Our analysis revealed that students performed significantly better on blocks-based questions when the questions pertained to conditional logic, iterative logic, and procedures, and performed better, although not statistically significantly so, on questions related to variables in the graphical condition. Interestingly, there was no difference in performance on comprehension questions between the two modalities. This suggests that while the graphical representation supports students in understanding what a construct does (i.e. what the output from using it is), that support does not better facilitate learners in understanding how to use that construct or how it fits into a larger algorithm. A closer analysis revealed a more nuanced relationship between modality and concept than this high-level overview suggest [4].

VI. Conclusion and Future Work

The pilot study gave us the opportunity to test out a number of components of the study design, including our attitudinal and content assessments and our curricular materials, interview protocols, and data collection strategies. While the pilot study yielded numerous useful insights, the choice of environment (Snap!) and the lack of a full text-based condition limited our ability to answer our stated research questions. In the next iteration of the study we will be using a customized version of Pencil Code (Fig. 1), which will give us isomorphic blocks, text, and hybrid interfaces. Our expectation is that the second iteration of the study will provide that data necessary to answer the questions at the heart of this dissertation; namely the relationship between programming modality and understanding in introductory high school programing classrooms, and insight into the design of introductory programming tools that can provide guidance on the creation of the next generation of computer science learning environments. In doing so, we hope to contribute to the development of tools and curricula that will prepare today's students for the computational futures that await them.

References

[1] D. Bau, D. A. Bau, M. Dawson, and C. S. Pickens, "Pencil Code: Block Code for a Text World," in *Proc. of the 14th IDC Conf.*, New York, NY, USA, 2015, pp. 445–448.

[2] B. Harvey and J. Mönig, "Bringing 'no ceiling' to Scratch: Can one language serve kids and computer scientists?," in *Proc. of Constructionism 2010*, Paris, France, 2010, pp. 1–10.

[3] D. Weintrop and U. Wilensky, "To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming," in *Proc. of the 14th IDC Conf.*, New York, NY, USA, 2015, pp. 199–208.

[4] D. Weintrop and U. Wilensky, "Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs," in *Proc. of the 12th ICER Conf.*, Omaha, NE, 2015.