# Comparing Block-based, Text-based, and Hybrid Blocks/Text Programming Environments in Introductory Computer Science Classes

David Weintrop
Learning Sciences, School of Education and Social Policy
Northwestern University

A long-standing design challenge for computer science educators is deciding where to start on day one: What programming language to choose? In what environment? What activity should learners engage in? 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 popular feature of introductory computer science curricula and programming interventions targeted at young learners. Despite its growing popularity, open questions remain surrounding the effectiveness of blocks-based programming for helping students learn basic programming concepts and the overall effectiveness of the approach for preparing learners for future computer science learning opportunities. The goal of this dissertation is to shed light on these two open questions.

The first of these two questions pertains to the effect of the blocks-based modality on students' resulting understandings of programming concepts and the programming practices they develop. A growing body of literature is emerging around the effects, both positive and negative, of using blocks-based programming environments with programming novices in formal education environments (e.g. Cooper et al., 2000; Lewis, 2010; Meerbaum-Salant et al., 2010, 2011). The first goal of this dissertation is to contribute to this literature by conducting an in-depth evaluation of emerging learner understandings built on cognitive interviews and detailed analysis of the learner produced artifacts. Most literature to date has been based on questionnaires and pre/post test written assessments, which provide data on outcomes, but little in the way of mechanism and insight into the affordances of the representations used. The use of interviews, observations, and computational methods in conjunction with conventional pre/post measures, will be used to better understand the effects, both positive and negative, of using blocks-based programming environments in formal educational contexts. The emphasis will be understanding the role of the blocks-based representation on students emerging understandings of core programming concepts, the programming practices that

develop in students by using this representation, and students' overall perception of the practice of programming.

The second question looks at the suitability of blocks-based programming 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 (e.g. Maloney et al., 2008), educators have had difficulty transitioning learners from these graphical environments to conventional text-based programming environments. Studies have found little transfer in knowledge between graphical environments intended to introduce learners to programming and text-based environments that serve as the core modality for further computer science studies (Chetty & Barlow-Jones, 2012; Powers, Ecott, & Hirshfield, 2007). Students often fail to see the relationship between the two programming modalities, which calls into question the effectiveness of blocks-based graphical environments in their capacity to serve as an effective introduction to computer science (Cliburn, 2008). A small number of studies have focused on this problem, but have focused on undergraduate audiences and have relied on add-ons to programming environment, as opposed to being a part of the environment itself (Dann et al., 2012). This shows that bridge blocks and text is possible, but requires a dedicated effort built into the overall approach. The most comprehensive study of the question of transfer between graphical and text-based programming found transfer effects (Hundhausen, Farley, & Brown, 2009), but used a direct-manipulation programming environment with little similarity to the visual presentation or types of activities use with the blocks-based tools that are the focus of this work. Further, little of this work has focused on younger learners or studied the most recent – and popular - generation of blocks-based programming tools. I hypothesize that while more seasoned programmers who have experience with text-based programming recognize the relationship between graphical and text-based programming representations, this link is less apparent to learners – the audience the environments are designed for. If this is true, it has consequences as to the appropriateness of using graphical tools for introductory programming courses.

This study will consist of a 3-condition comparison between blocks-based, text-based, and hybrid text/blocks-based introductory programming tools in a formal computer science education context. The study will proceed in three phases. The first phase will be the design of the programming tools and development of curricular materials to be used in each of the three conditions. The second phase of this dissertation will follow three classrooms, each using one of the three designed programming tools, and seeks to identify the programming concepts learned, the practices developed, and students' attitudes towards and perceptions of programming, looking for differences across the three modalities. The third phase will follow students in the three classrooms as they transition from the introductory programming tools to conventional programming languages and focus on if and how students draw on their recently acquired programming

knowledge and practices when first transitioning to the conventional text-based programming tools used for future computer science educational courses.

In this dissertation I will answer the following three research questions:

1. What are the effects of blocks-based graphical programming environments on learners' understandings of programming concepts? What programming practices do learners develop when working in blocks-based programming environments? And how do these environments affect students' perceptions of and attitudes towards programming? How do they differ between text-based, blocks-based, and hybrid programming tools?
2. How do understandings and practices developed while working with blocks-based and hybrid blocks/text tools support or hinder the transition to text-based programming languages? Which concepts transfer to text-based environments and which are lost? What practices developed in block-based programming environments are preserved and which are abandoned? How do learners' understanding of and attitudes towards programming change as learners shift modalities?
3. How can we design text-based programming environments that incorporate the features of blocks-based programming environments that make them effective tools for introducing learners to programming?

The result of this dissertation will be:

- A clearer understanding of the relationship between programming modality and resulting understandings of programming concepts, the practices that develop, and the attitudes toward and perceptions of programming they promote in learners.
- Documenting the challenges learners face in transitioning from introductory programming environments to the tools used in later computer science educational contexts, including what concepts and practices successfully transfer and which do not.
- Evidence-based recommendations on features to look for and features to avoid when choosing introductory programming environments, languages, and tools.
- A demonstration of a hybrid text/block-based programming environment and whether or not such approaches offer advantages over either text or blocks based tools.

## References


Chetty, J., & Barlow-Jones, G. (2012). Bridging the Gap: the Role of Mediated Transfer for Computer Programming. *International Proceedings of Computer Science & Information Technology*, *43*.

Cliburn, D. C. (2008). Student opinions of Alice in CS1. In *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual* (p. T3B–1). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4720254

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, *15*(5), 107–116.

Dann, W., Cosgrove, D., Slater, D., Culyba, D., & Cooper, S. (2012). Mediated transfer: Alice 3 to Java. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 141–146). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2157180

Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: A "cognitive dimensions" framework. *Journal of Visual Languages and Computing*, *7*(2), 131–174.

Hundhausen, C. D., Farley, S. F., & Brown, J. L. (2009). Can direct manipulation lower the barriers to computer programming and promote transfer of training? *ACM Transactions on Computer-Human Interaction*, *16*(3), 1–40. doi:10.1145/1592440.1592442

Lewis, C. M. (2010). How programming environment shapes perception, learning and goals: Logo vs. Scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 346–350). New York, NY.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *ACM SIGCSE Bulletin*, *40*(1), 367–371.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in Scratch. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp. 168–172). Darmstadt, Germany: ACM.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. M. (2010). Learning computer science concepts with scratch. In *Proceedings of the Sixth international workshop on Computing education research* (pp. 69–76).

Powers, K., Ecott, S., & Hirshfield, L. M. (2007). Through the looking glass: teaching CS0 with Alice. *ACM SIGCSE Bulletin*, *39*(1), 213–217.