Media Designs with Scratch: What Urban Youth Can Learn about Programming in a Computer Clubhouse

John Maloney, Mitchel Resnick, Natalie Rusk, MIT Media Laboratory,
77 Massachusetts Ave. E15-020, Cambridge, MA 02139, USA
Email: jmaloney@media.mit.edu, mres@media.mit.edu, nrusk@media.mit.edu
Kylie A. Peppler, Indiana University, 201 N Rose Ave, Wright 4024,
Bloomington, IN 47405, USA, kpeppler@indiana.edu
Yasmin B. Kafai, UCLA Graduate School of Education and Information Studies, 2128 Moore Hall,
Box 951521, Los Angeles, California 90095-6293, USA, kafai@gseis.ucla.edu

Abstract: We report on the programming learning experiences of urban youth ages 8-18 at a Computer Clubhouse located in South Central Los Angeles. Our analyses of the 536 Scratch projects, collected during a two-year period, documents the learning of key programming concepts in the absence of instructional interventions or experienced mentors. We discuss the motivations of urban youth who choose to program in Scratch and the implications for introducing programming at after school settings in underserved communities.

Numerous approaches to broadening participation in computing have been discussed in K-12 and college education, such as mentoring, revised curricula, tool development outreach programs, and programming courses for non-majors (Margolis & Fisher, 2003). A surprisingly neglected area of research is the learning of programming in community technology centers. In these venues, the learning of programming is more casual and takes place at the discretion of the learner rather than part of a formal curriculum. Such out-of-school activities also present opportunities for youth to succeed who may not flourish in traditional school environments. As a case in point, we focus on the use of Scratch (see Figure 1 or www.scratch.mit.edu), a block-based programming language designed to facilitate media manipulation for novice programmers (Resnick, Kafai, & Maeda, 2003), at a Computer Clubhouse—an urban community technology center (Resnick, Rusk, & Cooke, 1998). Scratch is not the first programming environment aimed at novice programmers (for an extensive overview, see Kelleher & Pausch, 2005). It follows the Logo tradition (Papert, 1980) but emphasizes media manipulation and supports youths' interests, such as creating animated stories, games, and interactive presentations. The Scratch vocabulary of roughly 90 commands includes commands for relative motion like the Logo turtle, image transformations, cell animation, recorded-sound playback, musical note and drum sounds, and a programmable pen, in addition to standard control structures, global and local variables, and simple Boolean operations.

During a period of two years, we collected youths' Scratch projects, which included animations, digital art, and games, on a weekly basis in order to track which programming concepts were taking root in the Clubhouse culture over time. As information sources for this study, we exported project summary files, which contained text-based information such as the date, file name, and author of the project as well as information about the number and types of commands that were used and the total number of stacks, sounds, and costumes used in the project. During the study, mentors were regularly at the site. The mentors had little or no experience programming and were new to Scratch (Kafai, Desai, Peppler, Chiu, & Moya, 2008).

Findings

We collected 536 Scratch projects, which constituted 34% of all the projects created at the Computer Clubhouse during the course of this study. Scratch was more heavily used than any other media-creation tool, including Microsoft Word, which was the next most widely adopted software (n = 461 files). These findings demonstrate that Scratch became a successful part of the local culture. It's also one of the few programming initiatives that successfully engaged equal numbers of boys and girls – all of them youth of color. Of the 536 Scratch projects, 111 of them contained no scripts at all. These "pre-scripting" projects illustrate the use of Scratch simply as a media manipulation and composition tool. Of the remaining 425 projects, all of them make use of sequential execution (i.e., a stack with more than one block) and most (374 projects, 88%) also show the use of threads (i.e. multiple scripts running in parallel). These are core programming concepts that confront every Scratch user when they begin scripting. In addition, we examined a number of other programming concepts: user interaction, such as use of keyboard or mouse, was used in 228 projects (53.6%), loops in 220 (51.8%), conditional statements in 111 (26.1%), communication and synchronization in 105 (24.7%), boolean logic in 46 (10.8%), variables in 41 (9.6%), and random numbers (4.7%). Unlike sequential execution, the aforementioned concepts are not needed in every project and were therefore used less frequently.

We also examined programming trends over time. When we compared the percentage of projects containing the various programming concepts over time, we found that five out of the seven concepts that we targeted for our analyses demonstrated significant gains ($p < .05$) during the second school year. Among these

were the less obvious concepts of variables, Boolean logic, and random numbers. Chi-Square tests were used to analyze differences in the percentages of projects containing targeted programming concepts from Year 1 to Year 2 (see Figure 2). Overall, four of the seven programming concepts (Loops, Boolean Logic, Variables, and Random Numbers) demonstrated significant gains in the number of projects utilizing the targeted concepts ($p < .001$). One of the remaining concepts (Conditional Statements) had marginal gains ($p = .051$) and one concept (Communication/Synchronization) demonstrated a reduction in the projects utilizing this concept.

Figure 1. Screenshot of the *Scratch* User Interface

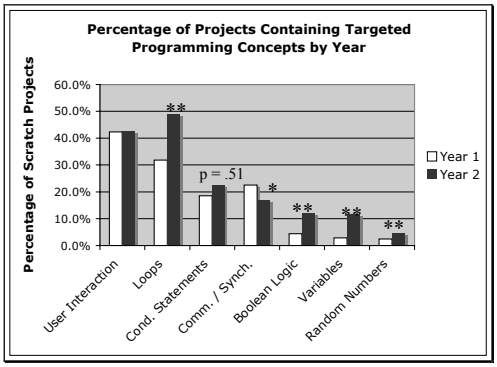

Figure 2. Graph demonstrating the change in the percentage of projects that used various programming concepts over time
**$p < .001$ *$p < .05$

Discussion

Our findings illustrate youths' sustained participation in programming activities. Clubhouse youth utilized commands demonstrating the concepts of user interaction, loops, conditionals, variables, Boolean logic, random numbers, and communication & synchronization. These findings are remarkable given the lack of formal instruction and the mentors' lack of prior programming experience. A more pressing question is: why did Clubhouse youth choose to get involved in Scratch programming given that they had many other software options? The best answer might have been provided by Kelleher and Pausch (2005) who noted how systems can make programming more accessible for novices "by simplifying the mechanics of programming, by providing support for learners, and by providing students with motivation to learn to program" (p. 131). We think that Scratch addresses all three of these areas. The design of the Scratch blocks simplifies the mechanics of programming by eliminating syntax errors. The social infrastructure of the Computer Clubhouse is also important in providing support for novice programmers. Finally, the multimedia aspect of Scratch facilitated urban youth's participation in programming. The project archive provided evidence that youth interest in technology starts with digital media and serves as a promising pathway into programming. The broad spectrum of media designs – from video games to music videos and greeting cards – is a true indicator of youth's interest in not only consuming digital media but in becoming creators themselves, a role often denied to urban youth.

References

Margolis, J. & Fisher, A. (2003). *Unlocking the Clubhouse: Women in Computing*. Cambridge, MA: MIT Press.

Kafai, Y. B., Desai, S., Peppler, K., Chiu, G. & Moya, J. (2008). Mentoring Partnerships in a Community Technology Center: A Constructionist Approach for Fostering Equitable Service Learning. *Mentoring & Tutoring*, *16*(2), 191-204.

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 88-137. Papert, S (1980). *Mindstorms.* New York: Basic Books.

Resnick, M., Kafai, Y. B., & Maeda, J. (2003). ITR: A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers. Proposal [funded] to the National Science Foundation, Washington, DC.

Resnick, M., Rusk, N., & Cooke, S. (1998). Computer Clubhouse: Technological fluency in the inner city. In D. Schon, B. Sanyal and W. Mitchell (Eds.), *High technology and low-income communities*. Cambridge, MA: MIT Press.

Papert, S. (1980). *Mindstorms.* New York: Basic Books.

Acknowledgments

This work was supported by a grant from the National Science Foundation (NSF-0325828) to Mitchel Resnick and Yasmin Kafai and by a dissertation fellowship from the Spencer Foundation to Kylie Peppler. The views expressed are those of the authors and do not represent the views of the supporting funding agencies or universities. We wish to thank Zrinka Bilusic for her preparation and initial analysis of the Scratch archive.