How Programming Environment Shapes Perception, Learning and Goals: Logo vs. Scratch

Colleen M. Lewis
Graduate School of Education
University of California, Berkeley
Berkeley, CA 94720
1- 510-388-7215
colleenl@berkeley.edu

ABSTRACT
This study compares the attitudinal and learning outcomes of sixth grade students programming in either Logo or Scratch. Given proposed affordances of the visual programming language, Scratch, I hypothesized that those students learning Scratch would demonstrate greater competence in interpreting loops and conditional statements and would have more positive attitudes towards programming. However, differences in performance between the two groups appeared only in the greater ability of the students that learned Scratch to interpret conditional statements. Contrary to our hypothesis, we found that students that learned Logo had on average higher confidence in their ability to program and students were no more likely to plan to continue to program after the course or view the learning of topics as difficult if they learned Logo or Scratch.

Categories and Subject Descriptors
K.3 [Computers and Education]: General

General Terms
Human factors

Keywords
Scratch, Logo, K-12, Programming

1. INTRODUCTION
Programming has been seen as an opportunity for students to develop the intellectual resources to tackle challenging problems [4]. However, many of the studies failed to show expected benefits and the Logo community did not flourish as hoped [1][3]. Many of the challenges faced by students learning to program in Logo can be understood through the limitations of the programming environment. For example, before testing a program in Logo, each line of code must adhere to syntactic constraints. This fragile environment may cause students to focus their attention on the syntax of the code and miss the opportunity to focus on the semantic meaning.

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

Building upon the early work on Logo, a new programming environment Scratch [2] offers much of the same functionality as Logo. However in Scratch the code blocks only lock into place in syntactically valid ways, therefore "bugs" are always semantic errors and never the result of a typing error or a misremembered detail of language syntax. Scratch is one of the recently developed visual programming languages that are thought to make complex elements of flow of control, such as loops and conditionals, more natural [5]. This study evaluates the pedagogical value of Scratch in comparison to a well researched tool, the Logo programming environment. We hypothesized that in comparison with students using Logo, students using Scratch would:

- Be more likely to report that programming in general and the learning of individual constructs was easy.
- Feel more confident about their competence writing computer programs and be more likely to report that they plan to continue to pursue computer programming.
- Be better able to trace the flow of control of loops and conditionals.

2. PREVIOUS RESEARCH

New languages, such as Scratch, typically build from previous languages to offer new functionality and claimed pedagogical advantages. This study builds upon a line of research that has begun to explicitly test the pedagogical claims of new programming tools. In a study designed to see if students learning Java will spontaneously develop competency with flow of control constructs in a visual programming language, Alice, the researchers, Parsons and Haden, found that "students struggled to make the connection between work in Alice and 'real programming'."[5]. In an effort to provide fewer syntactic constraints, visual programming languages may be perceived as "simple" and not related to real programming. However, research comparing learning in a more and a less syntactically strict language, Java and Python respectively, attribute the greater success of students in Python to be a result of reduced syntactic complexity [3].

3. METHODOLOGY

3.1 Setting
Participants were students enrolled in the course "Making Music, Movies, and Games with Computers", a summer enrichment program offered through the Academic Talent Development

Program (ATDP) at the University of California, Berkeley. The course met for a total of 36 hours over 12 days. All course participants had completed fifth grade and were between 10 and 12 years old. Students applied to participate in this enrichment program that is designed for academically advanced students.

There were 2 offerings of the course, with the exact same course description. Students indicated in their application a preference for the morning or the afternoon offering and were placed in their preferred offering on a first-come, first-served basis. Each offering of the course was taught by the same instructors, the author with the assistance of two teacher aids.

3.2 Treatment Groups

Each offering of the course became a treatment group in this study and the two treatments differed based upon which language they learned first. Treatment 1 (*Scratch-First*) learned to program in Scratch[1] for 6 days before beginning instruction in Logo. Treatment 2 (*Logo-First*) learned to program in Logo[2] for 6 days before beginning instruction in Scratch. This study focuses on these first 6 days and addresses the students' perceptions, learning and goals before beginning their second programming language.

3.3 Curriculum

Students in each class worked in pairs to complete worksheets designed by the researcher to introduce programming constructs. The first 6 days of instruction focused on the core programming constructs such as conditionals, variables and loops that are shared by both Logo and Scratch. On each of the first 6 days, students worked on worksheets designed to have students in each class accomplish equivalent goals. For example, on the third day of instruction students in each treatment were asked to draw a brick wall. Figure 1 shows the images that were provided on the respective worksheets to demonstrate the goal. Although the images and content of the worksheets differ, the goals on each worksheet were the same.

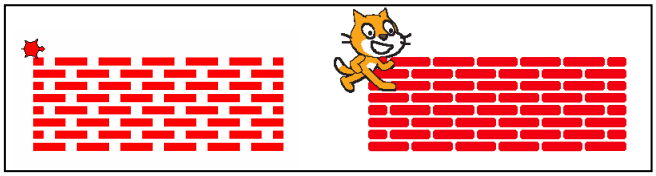

Figure 1: Demonstration of the goal of drawing a brick wall for Logo (left) and Scratch (right)

A recent multi-national, multi-institution study [6] suggests that it is possible to translate exams between languages while preserving the level of difficulty. A similar methodology is used in this study to create assessments and in-class programming tasks of comparable difficulty.

3.4 Participants

In Treatment 1 (*Scratch-First*) there were 16 males and 10 females for a total of 26 students. In Treatment 2 (*Logo-First*) there were 17 males and 7 females for a total of 24 students.

[1] Students used version 1.4 of Scratch (http://scratch.mit.edu).

[2] The version of Logo used was developed by Guy Haas (http://www.bfoit.org/itp/install.html).

3.5 Data Sources

Written assessments were given on the second and fifth days of class to assess student understanding of concepts and techniques covered the previous class day. Written surveys were given at the end of the sixth day of instruction and beginning of the seventh to gather information regarding the students' evaluation of their experience during class as well as their interests and goals.

4. DATA ANALYSIS

4.1 Similarity of Treatment Groups

When the students applied to the course, they were not aware of any difference between the two versions of the course. Based upon this sampling method, we did not suspect that the two groups would vary greatly. As a precaution, on the first day we administered a survey of prior computer experience. The differences responses of the participants in Treatment 1 and Treatment 2 were not statistically significant at the 5% level ($z=-1.007$; $p=0.3140$) using the Mann-Whitney-Wilcoxon test. Although this does not demonstrate that the two populations are equivalent, we postulate similarity based upon the sampling method and the similarity in distribution of responses regarding prior computer experience. To capture the similarity of the two groups, histograms of their responses to four representative questions are provided below in Figure 2. Participants in Treatment 1 (*Scratch-First*) and Treatment 2 (*Logo-First*) had similar distributions of responses to the following 4-level Likert questions.

- I am good at using the computer
- I find computers frustrating to use
- I am comfortable using a mouse
- I am comfortable typing on a computer

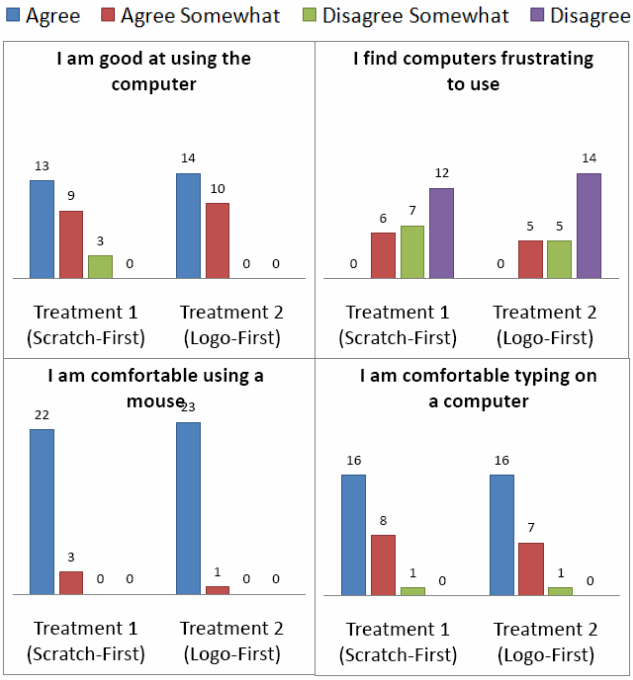

Figure 2: Student responses to 4-level Likert questions regarding prior computer experience

4.2 Student Report of Difficulty

To investigate the hypothesis that students in Treatment 1 (*Scratch-First*) would be more likely to judge programming and the learning of individual constructs to be easy, all students responded to questions regarding their experience of the difficulty of learning various programming commands and constructs. The questions were given on the 7th day of instruction, before introducing students to a second language. There were 8 questions covering the following constructs: `repeat`, variable creation, `if`, and, `forward/move`, set x and y coordinate, set pen color, and set heading. The students responded to a 4-level Likert question in the form of "It was hard to learn how to use [Programming Construct]". A response of "Agree" was coded as a 4 to indicate that the programming command was hard to learn while "Agree somewhat", "Disagree somewhat" and "Disagree" were coded 3, 2 and 1 respectively. The total of the coded scores for the 8 questions were added together to create a rating of perceived difficulty for each student.

Figure 3 below shows the students' rating of difficulty by treatment group. Students from Treatment 1 (*Scratch-First*) had a range of scores between 32 and 46 with a mean rating of 37.62 out of 64 and a standard deviation of 3.43. Students from Treatment 2 (*Logo-First*) had a similar range from 32 to 45 but a slightly higher mean of 38.35 and a standard deviation of 3.70. The difference between the means of the two groups was less than 1 out of 64 and was not significant at the 5% level (z=0.716, p=0.4743).

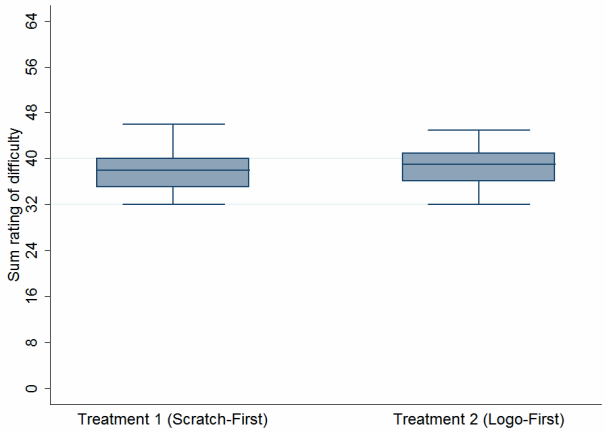

Figure 3: Student cumulative rating of difficulty of programming commands

The unexpected result of no difference between perceived difficulties motivated further investigation. Through an analysis of individual questions, we found that while most students in each treatment group disagreed with the statements "It was hard to learn how to use variables" and "It was hard to learn how to use `repeat`" the groups diverged when asked about learning conditionals. Two questions requested information about the process of learning conditionals: "It was hard to learn how to use `IF`" and "It was hard to learn how to use `AND`". There was a statistically significant response, using a Mann-Whitney-Wilcoxon test, between the two groups when responding to "It was hard to learn how to use `AND`" (z=2.423; p=0.0154). These results match the instructors' perception that students using Logo had a much more difficult time learning `AND`. Figure 4 shows the distribution of responses to four of the Likert questions regarding difficulty of learning. The histograms show the similar responses regarding variables and `repeat` to contrast the difference in distribution regarding the difficulty of conditionals.

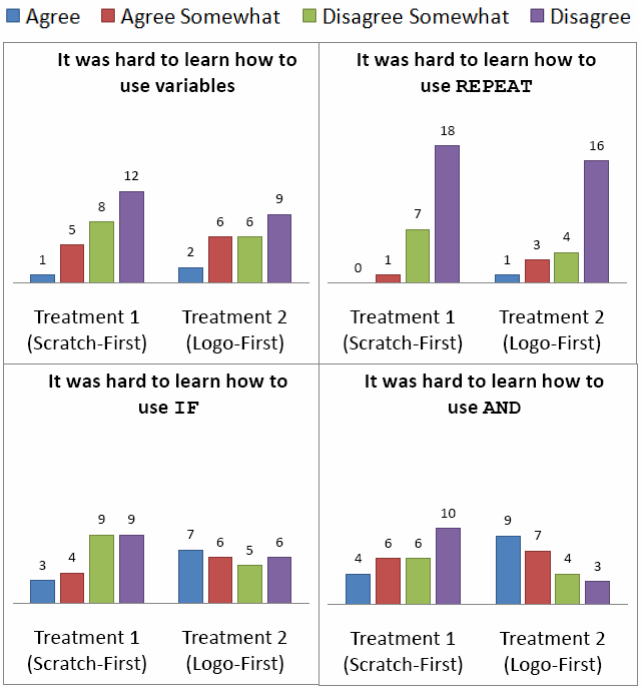

Figure 4: Student responses to 4-level Likert questions regarding the difficulty of learning

4.3 Confidence of Treatment Groups

To investigate the hypothesis that students in Treatment 1 (*Scratch-First*) will feel more confident about their competence writing computer programs and be more likely to report that they plan to continue to pursue computer programming the students were asked to respond to the following questions using a 4-level Likert response:

- Writing computer programs is easy.
- I am good at writing computer programs
- I plan to continue programming after the class is over.
- I want to take another computer programming course.

The results, shown in Figure 5, were analyzed using a Mann-Whitney-Wilcoxon test. Despite the expectation that Scratch makes programming easier, there was no statistical significance between the treatment groups on their response to the question: "Writing computer programs is easy" (z=-1.560; p=0.1189). However in response to the question "I am good at writing computer programs" it was Treatment 2 (*Logo-First*) that answered more positively and the differences between the treatment groups was statistically significant at the 5% level (z=-2.016, p=0.0438). This provides evidence that could prove the counter-hypothesis, that students in Logo will judge their competence at programming higher. It is possible that like the results from [5], the students learning Scratch did not recognize Scratch as "writing computer programs". However, other

references to "writing computer programs" in the course presumably serve to cue the student that they are engaged in "writing computer programs". For example, in the same survey, the students responded to the questions (emphasis added) "Writing *computer programs* is easy", "It is possible to know what a *computer program* will do before you run it", "When I run a *computer program* twice in a row it will do the same thing", and "What is the most frustrating thing about *writing computer programs*?"

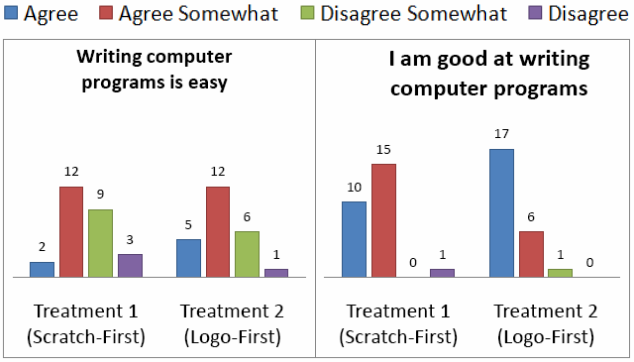

Figure 5: Student responses to 4-level Likert questions regarding competence programming

An alternate explanation is that even after 18 hours of instruction students in Treatment 1 (*Scratch-First)* had not used every command appearing in the Scratch menus, which might have eroded their confidence in their abilities. In contrast students using Logo may have been under the impression that they had learned everything that Logo had to offer. Future research will seek to isolate the effects of these different factors.

As shown in Figure 6, regardless of initial programming language, the majority of students agreed or agreed somewhat with the statements:

- I plan to continue to program after the class is over.
- I want to take another computer programming course.

This suggests that although we hypothesized that students exposed to Scratch would be more likely to plan to continue to program after the course, students indicate a similar level of intention to continue programming or take additional computer programming courses regardless of what languages they have seen.

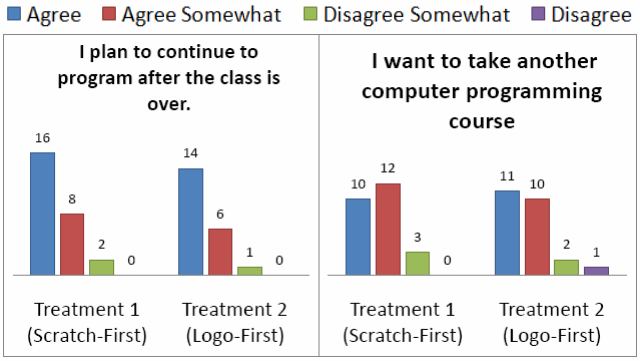

Figure 6: Student responses to 4-level Likert questions regarding their future plans for computer programming

4.4 Student Assessments

4.4.1.1 Student Interpretation of Loops

At the beginning of the 2nd day, after 3 hours of instruction, students were given an assessment where they were asked to describe the results of `repeat` statements. A representative question is shown in Figure 7. This example shows a nested `repeat` statement translated into both Logo and Scratch. Students in each treatment group saw the corresponding `repeat` statement and were asked to write answers to the following questions:

- How many beats/seconds will this example take to run?
- How many times will the example play the note 60?

```
repeat 5 [playnote 67 1 repeat 10 [playnote 60 1]]
```

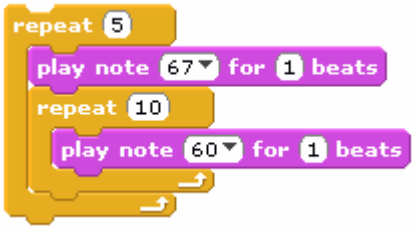

Figure 7: Example question from assessment of ability to interpret loops

We expected that the Scratch question would be easier for the students to answer because of the additional context regarding the meaning of the commands. For example, Figure 8 shows a comparison of the Logo and Scratch syntax to play a note. Whereas the Scratch command can be read like a sentence "play note 67 for 1 beats", in the Logo example there are no cues that differentiate the note to be played from the duration. Despite this apparent complexity, no students answered with a number over 55 on any questions, which might have indicated a student confusing the note number as the duration for the note. As opposed to confusion regarding the `play note` command, it appears that student difficulty was associated with the behavior of the `repeat`.

```
playnote 67 1
```

Figure 8: The `play note` command in Scratch and Logo

The performance difference between the two groups was not statistically significant (z=-0.945; p=0.3448). Students from Treatment 1 (*Scratch-First*) had a mean of 3.0 and a standard deviation of 2.6. Students from Treatment 2 (*Logo-First*) had a mean score of 2.4 and a standard deviation of 2.7.This assessment does not confirm our hypothesis that students using Scratch would outperform students using Logo on their ability to interpret repeat statements.

4.4.1.2 Student Interpretation of Conditionals

At the beginning of the 5th day, after 12 hours of instruction, students were given an assessment of their ability to interpret the

result of conditional statements. On the previous day, both groups of students had made a simplified paint program that involved setting the pen color depending upon the location of the mouse. The assessment asked them to consider similar examples of code and predict the final pen color or in some cases say what notes would be played. The most complicated example included interpreting the result of the code shown in Figure 9 for a series of x and y locations for the mouse.

```
setpencolor 1

if greater? mousey    0
     [setpencolor 2]

if less?    mousey    0
     [setpencolor 3]

if greater? mousex  200
     [setpencolor 4]

if greater? mousex  100
     [setpencolor 5]

if less?    mousex -100
     [setpencolor 6]
```

Figure 9: Question from assessment of ability to interpret conditionals expressions

Figure 10 below shows a box plot of the cumulative scores on the assessment of student ability to interpret conditional expressions. The mean score for students in Treatment 1 (*Scratch-First*) was 8.16 out of 13, while the mean score for Treatment 2 (*Logo-First*) was only 5.68. A Mann-Whitney-Wilcoxon test was used to compare the performance of each group and the higher performance of students in Treatment 1 (*Scratch-First*) was statistically significant at the 1% level (z=-2.528, p=0.0115).

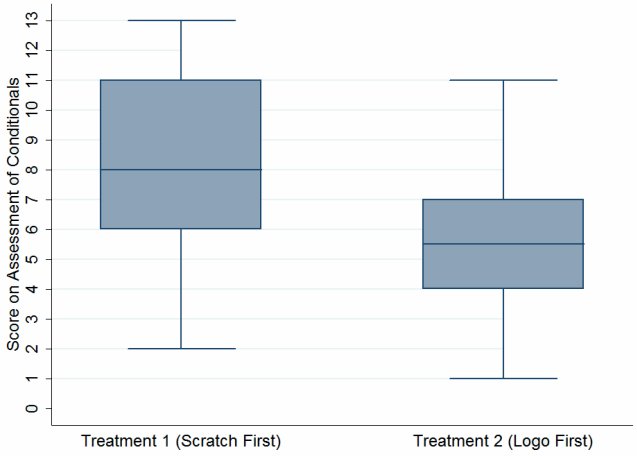

Figure 10: Box plot of student performance on assessment of conditionals

5. CONCLUSION AND FUTURE WORK

While the Logo environment appeared capable of supporting student development of confidence, interest in computer programming, and understanding of the loop construct, the Scratch environment provided a relative improvement in learning outcomes for students learning the construct of conditionals.

Scratch offers a number of affordances that would suggest that it should be easier to learn and interpret. Given these affordances, it is surprising that the students learning Logo and Scratch performed similarly when interpreting loops. For example, in Logo students have only a textual representation and during the learning process would presumably be distracted by low-level details such as syntax. However, perhaps this low-level focus allows students to focus on some important low-level details, such as the role of every command in a larger program. It may be that this low-level focus allowed students learning Logo to compensate for the lack of a visual representation.

There has been a significant amount of research done on student experiences in Logo however, only through a comparison with a different programming environment, such as Scratch, can we begin to isolate the features of student experience impacted by the content of programming and those impacted by the programming environment.

6. ACKNOWLEDGMENTS

I want to thank the study participants, Andrea diSessa, Michael Clancy, Randi Engle, Steven Kisely, George Wang and other members of the Patterns, UC-WISE and Video research groups at the University of California, Berkeley. I want to thank the Scratch team at MIT and Guy Haas for his development and support of JLogo. This work was partially supported by a grant from the Spencer Foundation (grant number 200500036) to Andrea A. diSessa, PI.

7. REFERENCES

[1] diSessa, A. (2000). Changing Minds: Computers, Learning, and Literacy. MIT Press:Cambridge, MA.

[2] Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by Choice: Urban Youth Learning Programming with Scratch. *ACM Special Interest Group on Computer Science Education.,* Portland: ACM.

[3] Mannila, L., Peltomaki, M., & Salakoski, T. (2006). What About a Simple Language? Analyzing the Difficulties in Learning to Program. Computer Science Education, 16(3), 211-227.

[4] Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books, Inc.

[5] Parsons, D., & Haden, P. (2007). Programming Osmosis: Knowledge Transfer from Imperative to Visual Programming Environments. In S. Mann & N. Bridgeman (Eds.), Procedings of The Twentieth Annual NACCQ Conference (pp. 209-215). Hamilton, New Zealand.

[6] Whalley, J. (2006). CSEd Research Instrument Design: The Localisation Problem. In S. Mann & N. Bridgeman (Eds.), Proceedings of The Nineteenth Annual NACCQ Conference (pp. 307-312). Wellington, New Zealand.