Computer Aided Instruction as a Vehicle for Problem Solving: Scratch Programming Environment in the Middle Years Classroom

Quincy Brown, William Mongan, Dara Kusic, Elaine Garbarine, Eli Fromm, Adam Fontecchio
College of Engineering
Drexel University
Philadelphia, PA, 19104, USA
qb23,wmm24,emg26,kusic,fromme,af63@drexel.edu

Abstract

Since the 1960's, Computer Assisted Instruction (CAI) has been promoted as the new standard for teaching and learning. Schools were provided with computers and internet connections at an astounding rate in the 1990's, but there was no correlated increase in student performance.[1] Investigation into this problem has revealed that computer technology is simply used to augment traditional 'instructionist' teaching strategies,[1,2] and this type of integration does not parallel the current real-time problem solving domain that is driven by technology.

Therefore, the integration of technology to reinforce science, technology, engineering and mathematics (STEM) education must not simply augment the existing teaching framework, but also present open-ended engineering problems that require a combination of problem-solving intuition and strategies learned in the classroom. The goal of this work, conducted via the NSF GK-12 program at Drexel University, is to evaluate the integration of computer-aided instruction and computer programming strategies learned in the classroom as it impacts the open-ended problem solving skills of grade 5 and 6 students. The NSF GK-12 program at Drexel University aims to use engineering as a contextual vehicle to augment STEM education, as well as to inspire students to pursue engineering disciplines.

In this work, we introduce computer-aided instruction using the Scratch[3] programming environment for children as a context for problem-solving to engage and assess the problem-solving skills of the students who use them. Reitman defines a problem solver as a person perceiving and adapting a goal without an immediate means of reaching the goal.[4] We utilize the child-friendly Scratch programming environment on the hypothesis that an unfamiliar problem domain can be better approached by students who have been taught to deconstruct mathematical concepts and logical sequences into the simple steps to be understood by a computer. This approach incorporates microelectronic technology as both the means and the subject of learning transferable problem solving skills.

Educational research indicates a deficiency in the measurable impact of technology in the classroom.[2] This work measures and evaluates the impact of CAI on student performance by evaluating the students' use of problem-solving skills and heuristics.[5,6] Students in two fifth and two sixth grade classes as well as a control group receive a pre-test and post-test asking them to solve mathematical problems using any approach. After receiving the pre-tests, the non-control students participate in a series of Scratch-based activities designed to improve their problem

solving skills. During each lesson with Scratch, students are taught to use a variety of problem solving skills and strategies. The emphasis is upon learning problem solving strategies and developing the vocabulary and practice of expressing them during use. Students in the treatment group show improvement in their problem solving skills at a rate greater than those in the control group and this improvement constitutes a statistically significant impact of CAI in the classroom.

Key words: K-12, problem solving, mathematics education, technological fluency, STEM education, computer programming, Scratch

I. Introduction

Attracting and retaining minority students in science, technology, engineering and mathematic (STEM) fields continue to challenge our nation's university systems. The causes and consequences of the small numbers of minority students in STEM fields has a self-perpetuating quality – a small number of minority graduates in STEM fields results in a small number of those who return to the community to lead the way for the next generation of graduates. This shortfall is further highlighted by the "digital divide",[7–9] the gap between those who have access to technology, including the information and proficiencies that accompany it, and those who do not. Often, the digital divide falls along the lines that separate income levels, ethnic groups and educational attainment.[7] Bridging the digital divide by providing technology education in the schools has been shown to improve academic performance in K-12 students[10] and increases the number students prepared to embark on STEM careers.

The Pennsylvania Department of Education's academic standards specifically cite "Technology Education" as an objective at all K-12 grade levels.[11] Included in this set of standards are missives such as, "apply basic computer operations and concepts," with minimal guidance as to how to design and implement lesson plans to advance the student's technical proficiency. While this openness enables freedom within curriculum development, it can also mean that technology programs are underdeveloped, understaffed and without the technical support needed to sustain such programs on a daily basis.

To address these challenges, educational Fellows of the National Science Foundation's (NSF) GK12 program have introduced a pilot program of computer-aided instruction (CAI) at two site-specific public schools having predominantly minority students. The NSF Fellows are Drexel University Ph.D. students in several computing-related disciplines who have utilized the child-friendly Scratch programming environment[3] to introduce fifth and sixth grade students at two schools in the Philadelphia School District to basic concepts of computer programming. To increase the impact of the lessons on the STEM preparedness of the students, the Fellows have explicitly addressed educational standards in Mathematics in addition to those standards in Science and Technology. The overall objectives of the Scratch-based CAI initiative at the schools are to

- develop CAI curricula that specifically address objectives in the state mandated educational standards; and to
- strengthen the mathematical reasoning skills of the fifth and sixth grade students to utilize arithmetic operations efficiently and to make quantitative decisions in context; and to

- improve fifth and sixth grade school students' attitudes toward and interest in computing technology, particularly as respects the students' vantage point as creators, rather than simply users, of technology; and to
- increase the students' knowledge of computer hardware and software terminology; and to
- improve the Fellows' experience in developing CAI curricula that address state standards, implementing those lesson plans in a classroom setting, and assessing the quantitative impact of the program.

In this pilot project, for the 2007-2008 school year, students are introduced to many of the building blocks of computer science and mathematics in higher education. For example, one script created by the students requires decomposing a gaming objective into a set of simple instructions (algorithms), introduce data elements for keeping score (variables), and locate various points on the gaming "stage" (Cartesian coordinates). All of these mathematical concepts are currently beyond the scope of their fifth and sixth grade mathematics education, but can be easily introduced a game-type domain that is familiar to them most often in a recreational setting.

In this paper, we describe the development of the Scratch-based initiative, the methodology used to assess the program, and a quantitative evaluation of the impact upon the students' mathematical reasoning and problem-solving abilities. The paper is organized as follows. Section II provides a summary of research related to STEM education and CAI curricula. Section III describes the participants and context of the study. Section IV discusses our methodology and Section V discusses test administration and scoring. Section VI describes how Scratch is used as an instructional tool. Sections VII and VIII present the results of tests and surveys, respectively, and Section IX addresses the Fellows' ability to develop lesson plans. Section X concludes the paper and discusses ongoing research with respect to the program.

II. Related Work

In the midst of continuing low numbers of African-Americans graduating from undergraduate engineering programs,[12] it is critical to prepare students for the rigors of STEM education to support high retention rates. The number of African-American college students graduating with engineering baccalaureate degrees dropped from a high of 27,400 students in 2002 to a 10-year low of 24,300 students in 2005 (National Science Foundation, 2005). The global ranking of U.S. students in mathematics slips from 11th place in fourth grade to 15th place by the eighth grade (National Center for Education Statistics, 2003). Because of these statistics, U.S. universities and the K12 school systems, in order to increase the number of qualified applicants to engineering programs, have a "crucial role to play" in transforming K12 math and science education,[13] and to further introduce students to the career choices available in STEM fields.

The burden of making connections between mathematics and computer science largely falls upon the students,[14] but some curricula have been designed to more clearly expose this dependency.[14,15] Compellingly, test scores in some mathematical disciplines have shown statistically significant increases when students have been exposed to computer-aided learning and simulation software.[16,17] Researchers at the University of South Carolina report a near-uniform 5% increase in 5th grade test scores on double-digit division and multiplication after spending two weeks working with specially-developed mathematics education software.[17]

Making the connection between mathematics and daily life has been shown to correlate with high achievement. Researchers have found that high-achieving students were more likely to make a connection between everyday activities and mathematics, such as computing a discount while shopping, as compared to low-achieving math students.[18] Although the study did not investigate if better performance enables students to make the mathematical connections to everyday life, or if seeing the connection enables better math performance, the correlation holds promise for Scratch-based computer programming lessons. The students can choose backdrops of real-world environments in their programs, or they may simply connect the activity to their own recreational game playing.

The Scratch programming environment is conceptually similar to the Alice programming environment, in which programmatic elements are dragged and dropped as tiles to create animated movies and video games.[19] Alice has been used in introductory-level computer programming courses to expose students programming constructs free from the syntactical challenges posed by standard languages such as C/C++ and Java. The use of Alice in treatment groups at several community colleges was found to show statistically significant improvements in algorithmic thinking,[20] an ordering of logic that supports mathematical reasoning skills.

Learning of mathematics and problem solving tends to be cumulative, and the ability of students to advance in mathematics-dependent subjects relies upon mastering skills during the earlier years of schooling.[21] For this reason it is crucial to develop mathematical reasoning abilities early in the K-12 years.

A pilot effort at Arizona State University East was designed to introduce elementary school students to computer science and to have the students tackle real-world problems this context. The primary objective of the Arizona State project is to make technology accessible to school children, and in particular, to introduce students to "how" computers work rather than simply "what" computers can do. Initial results were positive and the students demonstrated proficiency in answering computer-technology questions, although the assessment was not extended to the mathematical concepts that are applicable to computer science.[22]

The Scratch environment has been promoted in schools and community centers for media-arts initiatives, technological fluency, and exposure to game programming,[23–25] but in published work has never been explicitly used to develop proficiency with mathematical problem-solving and reasoning. Similarly, initiatives using other multimedia development environments[26,27] and general K-12 computer science curricula[28] focus primarily on the technological fluency of the participants, whereas we aim to develop curricula that directly addresses fundamental mathematical problem-solving skills, in addition to state-mandated educational standards, within the context of computer programming.

III. Participants and Context

This study took place at two public schools in a financially under-resourced urban area of Philadelphia. These schools are populated by students of lower-income families and the percentage of students receiving free or reduced-price lunches is 61% and 90%. The students are predominantly African-American, comprising 80% and 97% of the student bodies at the two schools. The control group against which we compare the results of our treatment groups resides

at one of these two schools. The students were not randomly assigned to their classrooms by the authors. They were assigned using the school's normal methods. The author's, Teaching Fellows, were assigned to their partnering teacher based on their mutual teaching interests.

We developed a four-lesson Scratch curriculum for fifth and sixth grade students. The classes range in size from 22 to 33 students each. The students were instructed in math for an hour and a half each day. The Scratch curriculum consists of four 45-minute lessons over the duration of a month with two additional days for pre- and post-testing.

The classroom teachers in this study were eager to have their students use computers for more than their typical activities including Fun Brain[29] and First In Math.[30] The students enjoyed activities that enabled them to work individually to augment their daily math lessons. The classrooms have between four and six computers each, seldom used by the teachers for structured lessons due to their limited number. The computers were primarily available for students to use independently or without guided instruction. While the teachers understand the importance and potential of utilizing technology in mathematics education, they have not the expertise to develop lessons that truly integrate computing technology into existing curricular goals.

The school sites in this study were chosen for three reasons. The first reason is the existing relationship between the individual schools, the classroom teachers, and the NSF Graduate Fellows. The schools are in the second year of a three-year partnership with Drexel University. Two of the four NSF Graduate Fellows are participating for their second year and the other two are first year participants. One of the fellows is a Computer Science Ph.D. student, one is a graduate student in the Department of Computer Science and the School of Education, one is an Electrical Engineering Ph.D. student, and one is an Computer Engineering Ph.D. student. Prior to the beginning of the 2007 academic year none of the fellows had significant experience using the Scratch programming environment.

The second reason for choosing the sites was due to previous research investigating the use of technology in K-12 mathematics education. This research has concluded that low-performing students receive a significant benefit from the CAI curriculum.[31,32] In addition to the impact of the intervention, the gains can also be attributed to the increased level of attention paid to the subject matter by the students, thereby leading to increased performance.[33] Recognizing the similarities between the students in the studies and those in our classrooms, we believed that our students would greatly benefit from a Scratch-based technology curriculum.

The final reason for choosing the sites was based on the Fellows' experiences in developing lesson plans that integrate new educational technology while meeting requirements to specifically address the state of Pennsylvania educational standards. The Fellows desired to provide the students with the opportunity to utilize new and innovative technologies without interfering with instruction time to achieve the state and federally mandated instructional goals.

IV. Methodology

Each of the Fellows taught the Scratch lessons to one classroom. The classroom teachers were present during the lessons although they did not provide instructional support during the lessons. At one of the two schools, the students participated in the lessons in a school computer

You are given $20 to buy supplies for your class. A set of supplies costs $2.15. How many students can you purchase supplies for? How much change would you receive?

Fig. 1. Question from the pre-test

lab using Apple eMac computers. In classrooms at the other school, the students used Windows-based laptop computers brought into their classroom by the teaching Fellow. Classroom notes, pre- and post-test results, student Scratch files, and Bath County Computing Attitude Survey[34] results constitute the data used for this study.

In this study we set out to ask, "Can Scratch be used as an environment to strengthen students' problem solving skills; and, can allowing students to become technology creators improve their computing attitude; and, can the teaching Fellows develop lesson plans using new technology that specifically address state-mandated educational standards?" Three related analyses will provide insight into these questions:

- Assess the students' problem solving abilities, as measured by our pre- and post- test results. The tests were created to elicit specific problem solving behaviors. They consisted of mathematics problems that could be solved using various techniques, with some techniques being more efficient than others. The results are compared before and after the experiment, and compared for significance against those of the control group.
- Assess the students' computing attitudes, as measured by the Bath County Computing Attitudes Survey. The survey was administered to the students prior to beginning any of the lessons. The survey was modified to use language familiar to the students. The surveys were coded so the students were asked an additional question identifying their gender at the end of the survey.
- Fellows' ability to develop lesson plans addressing state standards, as documented by the production of four lesson plans ready for publication in the Teach Engineering[35] database. All of the Scratch-based plans indicate the state standards addressed in each lesson.

V. Pre- and Post-test Administration and Scoring

Three fifth grade and three sixth grade classes took the pre-test and post-test. A total of 135 students took the pre-test and 133 students took the post-test; 113 students took both tests. Our analysis pertains to those students who took both tests. The total includes a control group consisting of 40 students from fifth and sixth grade who were also tested. Both pre- and post-test contained three questions each presenting the students with an opportunity to employ an efficient problem-solving strategy (e.g. multiplication) or a less efficient method (e.g. repeated addition) yielding the same answer. Fig. 1 shows a sample question from the assessment pre-test. All test items are aligned with exams and standards that the students are regularly expected to perform and address at their grade level.

The tests were analyzed for problem solving strategies, as described in Table I, to emphasize the use of an appropriate approach more the arrival at a correct answer. To score the tests, each question is given one or more letter codes from Table I. It is possible to receive more than one code for a question if, for example, the student begins to use an incorrect or inefficient

TABLE I

ENCODING AND EXPLANATION OF THE SCORING FOR PRE-TESTS AND POST-TESTS

Coding	*Explanation*
(A) Used technique appropriately to completion	The most appropriate technique was used to completely answer the question being asked.
(B) Used technique appropriately not to completion	The most appropriate technique was attempted, but a serious mechanical error was made in its execution, or the question was not completely answered.
(C) Used technique inappropriately	A suboptimal technique (for example, using repeated addition instead of multiplication) was used to answer the question, or to attempt to answer the question.
(D) Minor mechanical error not related to technique	Regardless of the technique chosen, a small arithmetic error was made. Enough evidence was shown to indicate that the student understood the technique to use to solve the problem, but made a simple error perhaps due to the time constraint. This code is used in conjunction with other codes; for example, if a student used the optimal technique but made a small adding mistake, this code is used along with "Used technique appropriately to completion."
(E) Off task	No attempt was made to solve the problem, the student's work is unintelligible, or the student's work cannot be interpreted.

methodology for solving a problem (e.g. using repeated addition or subtraction instead of multiplication or division), and then changes to the correct one. It is also possible to receive multiple codes if the student uses an efficient methodology to solve the problem but makes a minor mechanical error (e.g. a small arithmetic mistake). The goal is to receive a code of A, indicating that the correct technique was used to solve the problem to completion.

Each letter score in Table I corresponds to a numeric score. A score of *A* corresponds to 16 points for using an efficient technique to complete the problem, a score of *B* corresponds to 8 points for using an efficient technique but not finishing the problem, a score of *C* is worth 4 points for using less efficient mathematical technique, and scores of *D* and *E* are worth no points for either making a mechanical error or for being off-task. As there were three questions on each test, the total possible score is 48.

VI. Scratch Lesson Description and Administration

The Scratch-based lessons are designed to provide students with practice in applying and evaluating mathematical problem solving techniques in the context of computer programming. The students are then expected to apply some of those mathematical problem solving techniques to in-situ problems that they have practiced during the Scratch lessons. The Scratch lessons introduce students to both efficient and inefficient mathematical problem solving methods used by computer scientists in real-life contexts, and provide the students with a visual means to evaluate their results after applying a particular method.

Fig. 2 shows an example of the visual cue presented to the students to evaluate the correctness of their solution. In the Scratch-based lesson of Fig. 2, students were asked to "debug" a Scratch script, a common term used in computer programming to correct errors in an existing program.

In this lesson, the game character may pick a fixed number of cherries each time he jumps, and we call this number n. The character's basket, however, holds a number of cherries that is not evenly divisible by n. Fig. 2(b) shows what happens if the character jumps too many times and his basket overfills with cherries.

One of the objectives of this exercise is to help the students visualize the mechanics occurring within a multiplication operation. In the pre-test, it was clear that the students often used repeated addition operations rather than a more efficient multiplication operation. Students were taught that a computer programmer will use multiplication to efficiently encode a "looped" operation instructing the game character to jump specified number of times, which the game character then illustrates as an act of repeatedly adding cherries to the basket. In another Scratch activity, this point was further emphasized by drawing a square on the Scratch "stage" first by placing the same instructions four times, once for every side of the square, then by substituting a control structure to loop one instruction four times. Many students made mistakes when they used the repeated instruction-placement method, revealed to them when the square was mis-drawn, to highlight the weaknesses of using an "inefficient" (addition) method versus a more efficient (multiplication) method.

In addition to learning to substitute one multiplication operation for several addition operations, students also practiced rounding numbers in context with the Scratch exercise in Fig. 2. Students demonstrated in the pre-test that they often have trouble rounding numbers in context, instead relying upon ruled-based methods (e.g. rounding up when the next significant digit is greater than or equal to five). For example, in the pre-test, students were given a hypothetical $20 bill to buy supply kits at a cost of $2.15 each for their classmates. The question asked for how many classmates could they buy supplies. Many students clearly demonstrated that they had trouble reasoning to round the number of supply kits they could purchase down to the nearest whole number. In the Scratch activity shown in Fig. 2(b), the temptation for a student would be to round the number of jumps to the next largest whole number, causing the basket to overflow as in Fig. 2(b). When the jumping action is looped a correct number of times, determined by rounding down in this context, the student sees that the basket does not overflow before the program terminates.

VII. Pre-test and Post-test Results

Of the three treatment groups, two groups increased their mean scores by approximately 5 (out of 48) points. The largest treatment group was also the highest-scoring treatment group for the pre-test, but experiences only a small improvement of half a point on average. The control group actually out-scored the treatment groups in the pre-test, but showed no improvement between the pre-test and post-test. As a whole, the treatment groups show an 9% improvement between the pre-test and post-test in choosing an efficient mathematical technique, as compared to the control group's 26% decrease.

An analysis of covariance (ANCOVA)[36] is used to answer the question, "Is there a strong linear relationship between the pre- and post-test scores within the individual classes as well as between the treatment groups and control group?" This analysis is two-fold. First, Levene's Test for Equality of Variances[36] yields an F value that enables us to conclude that the mean

(a) (b)

Fig. 2. (a) Screen shot of the Cherry Picker game before basket overflow; and (b) a screen shot after basket overflow

TABLE II

PRE- AND POST- TEST DATA ANALYSIS

Group	*Number of Students*	*Pre-Test Mean*	*Post-Test Mean*	*Percent Increase*
Treatment Group 1	19	28.9	33.5	16%
Treatment Group 2	40	30.4	30.9	2%
Treatment Group 3	14	17.4	22.5	29%
All Treatment Groups	73	27.5	30.0	9%
Control Group	40	30.5	22.9	-26%

variances are equal and that the post-test scores are equally spread between the treatment and control groups, $F(1, 111) = 0.83, p < 0.01$.

Despite the conclusion from the Levene's test, the Tests of Between Subjects[36] yields an F value concluding no significance and does not allow us to reject the null hypothesis, $F(2, 111) = 24.54, p < 0.05$. We expected this test to yield the opposite conclusion and this enables us to statistically conclude that the post-test increases were caused by the treatment. As a result of this analysis, we put forth several plausible explanations.

First, we recognize that there are factors beyond our control such as exceptional teacher performance, help from school support staff, or home assistance to the students that could contribute to the performance improvements within the treatment group. Second, the Scratch lessons are designed to provide students with practice in applying and evaluating different methods of problem solving; this practice may lead to an increased amount of time spent on-task and a greater attention to correctness. Such contributing factors, independent of the Scratch lesson methodology, could lead to improved performance in the treatment group.

VIII. Survey Results Analysis

Computing attitudes are important to measure because in addition to helping students to improve their application and evaluation of mathematical problem solving techniques, the Fellows

desire for students to experience a vantage point as creators, rather than consumers, of technology. To assess the attitudes students have toward computing technology, we administered the Bath County Computer Attitudes Survey.[34] The survey is designed to measure the attitudes of students in grades four through twelve towards computers. The Construct Validity was validated by Bear et al. using 551 students in grades four through seven.[37] The analysis was performed using data from 70 students from the treatment group, those who fully answered the pre- and post-test questions and completed they survey. We selected this survey because of its applicability to our population's age group.

Using Pearson Correlation Analysis,[36] we found a significant correlation between survey questions that pertain to a student's interest in studying technology. This correlation seems natural, since the students are being taught using a modality and medium with which they indicate a high degree of familiarity, similar to computer games. The majority of students indicate an interest in, familiarity with, and access to computing technology, all suggestive that Scratch as an instructional tool would be an effective way to practice the mathematical reasoning skills useful to problem solving. The practiced application and evaluation of mathematical methods are essential for mastering core curricular content as guided by state educational standards. Therefore, Scratch could be used as a way to improve academic performance on core curricular material.

For the question, "People who like computers are weird," there is a significant correlation between gender and agreement with this statement, $r(70) = -0.254, p < 0.05$. All those who responded positively to this question were male. Gender also correlates to responses to the question, "Learning to program a computer is something I can do without," $r(70) = -0.309, p < 0.01$. Female students, in majority, disagreed with this statement while male students agreed with it.

Examining the questions that pertain to home ownership of computers and the perceived usefulness of and computers, we note that the overwhelming majority of students, 86%, report owning a computer at home and that 70% of students use computers at least one hour per week. Unfortunately, a lower percentage, 50% of students, agree that "computers help people to think." This leads us to speculate that computers, to many students, are for recreational rather than educational purposes.

The survey score histogram is shown in Fig. 3. The maximum survey score for a positive attitude toward technology, 52 points, is obtained by one student. The most frequent survey score $(n = 16)$ is 46 points. The average survey score is 41 points. We interpret these results to mean that the students currently have positive attitudes towards computing, yet there is ample room to expand the range of uses to which students apply their computer time. Cultivating positive attitudes is important because of research correlating post-secondary computer science student retention and positive computing attitudes.[38] Further, the room for improvement is significant to the Fellows because of the continuing objective to have students see themselves as creators, rather than simply users, of technology.

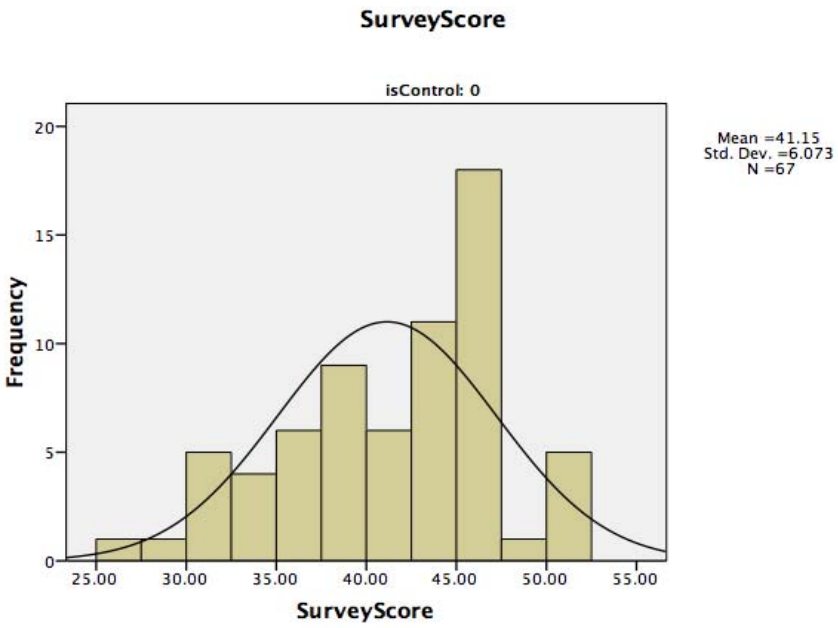

Fig. 3. Computing attitude survey scores for the treatment groups

IX. Fellows' Ability to Develop Lesson Plans

To satisfy the requirements of the GK-12 Fellowship program, teaching Fellows are required to develop lesson plans that conform to a specific template. To ensure the activities we do with students support their required curricular goals we are required to correlate the lessons we develop with the state standards. We were successful in developing full lesson plans that addressed the following state standards:

- **Science & Technology:**
- 3.1.7.A Explain the parts of a simple system and their relationship to each other.
- 3.1.7.B Describe the use of models as an application of scientific or technological concepts.
- 3.2.7.A Explain and apply scientific and technological knowledge.
- 3.2.7.C Identify and use the elements of scientific inquiry to solve problems.
- 3.6.7.B Explain information technologies of encoding, transmitting, receiving, storing, retrieving, and decoding.
- 3.7.A Describe the safe and appropriate use of tools, materials, and techniques to answer questions and solve problems.
- 3.7.C Explain and demonstrate basic computer operations and concepts.
- 3.7.D Apply computer software to solve specific problems.
- **Mathematics:**
- 2.2 Computation and Estimation
- 2.4 Mathematical Reasoning and Connections
- 2.5 Mathematical Problem Solving and Communication
- 2.9 Geometry

X. Conclusion

The purpose of this study was to investigate the use of Scratch in middle school classrooms to teach problem solving skills to students. Based on the results of this study, it can be concluded that the Scratch-based lessons can be beneficial to the students learning problem-solving skills. This is consistent with previous research on the use of technology-based curricula that indicates its efficacy in K-12 education.

The primary finding from this study is that on gain scores on the mathematics test, the students who the Scratch-based lessons, outperformed those who did not use it. The survey results highlight students' willingness to participate in computer-based activities despite not using computers for large amounts of time. The Fellows were successfully able develop a set of standards-based lesson plans that integrate technology without sacrificing the students mandated curricular goals. The lessons and findings presented in this study are the first portion of a year-long study the Fellows will continue implementing in their respective schools. Our future work includes the development and publication of additional lesson plans.

References

1. R. Sawyer, *The Cambridge Handbook of The Learning Sciences*. Cambridge University Press, 2006.
2. L. Cuban, *Oversold and Underused: Computers in the Classroom*. Harvard University Press, 2001.
3. Scratch, *http://scratch.mit.edu*. M.I.T. Media Labs.
4. W. Reitman, *Cognition and Thought*. Wiley: New York, NY, 1965.
5. L. Fan and Y. Zhu, "Representation of problem-solving procedures: A comparative look at china, singapore, and us mathematics textbooks," *Educational Studies in Mathematics*, vol. 66, no. 1, pp. 61–75, Sept. 2007.
6. A. Ellis, *Research on Educational Innovations*. Eye on Education: Larchmont, NY, 2005.
7. "Falling through the net: Defining the digital divide," National Telecommunications and Information Administration, http://www.ntia.doc.gov/ntiahome/fttn99/contents.html, Tech. Rep., Oct. 2000.
8. B. Gannod, "Technology education for kids: Cultivating technology professionals of tomorrow and today," in *Proc. of the ACM Annual Conf.*, Jun. 2003.
9. W. Ofosu, "Equity in the educational environment," in *Proc. of the ACM Annual Conf.*, Jun. 2001.
10. "e-learning: Putting a world-class education at the fingertips of all children," United States Department of Education, http://www.ed.gov/about/offices/list/os/technology/reports/e-learning.pdf, Tech. Rep., Dec. 2000.
11. "Academic standards for science and technology," Pennsylvania Department of Education, http://www.pde.state.pa.us/k12/lib/k12/scitech.pdf, Tech. Rep., Jan. 2002.
12. N. S. Foundation, *Bachelor's degrees awarded by field, citizenship, and race/ethnicity of recipients: 1995-2004*. NSF: http://www.nsf.gov/statistics/nsf07308/pdf/tab4.pdf, 2004.
13. M. Fox, *Colleges and Universities Hold Key to Improving Science and Mathematics for All*. http://www4.nationalacademies.org/onpi/oped.nsf, Jun. 1999.
14. S. Leung and C. Johnson, "A hierarchical optimization framework for autonomic performance management of distributed computing systems," in *Proc. of the ACM SIGITE*, Oct. 2005, pp. 37–42.
15. A. Ralston, "Do we need any mathematics in computer science curricula?" *AMC SIGSCE Bulletin*, pp. 6–9, Jun. 2005.
16. L. Sherrell, J. Robertson, and T. Sellers, "Using software simulations as an aide in teaching combinatorics to high school students," *ACM J. of Computing Small Colleges*, 1993.
17. T. Richardson and J. Lyons, "Developing effective k-5 mathematics educational software," in *Proc. of the ASEE annual conference*, Jun. 2005.
18. R. Stevens, V. Mertl, S. Levias, L. McCarthy, S. Goldman, L. Martin, R. Pea, A. Booker, K. P. Blair, N. Nasir, M. Heimlich, G. Atukpawu, and K. O'Connor, "At home with mathematics: meanings and uses among families," in *Proc. of the ACM Conf. on Learning Sciences*, Jun. 2006, pp. 1088–1093.
19. Alice, *http://www.alice.org*. Carnegie Mellon University.
20. S. Cooper, W. Dann, A. Hutchinson, and B. Moskal, "The alice curriculum: Impact on women in programming courses," in *Proc. of the ASEE Annual Conf.*, Jun. 2006.
21. T. N. A. of Sciences, *Learning and Understanding: Improving Advanced Study of Mathematics and Science in U.S. High Schools*. NAS: http://www.nap.edu/openbook/0309074401, 2002.
22. B. Gannod, "Incorporating higher education computer animation principles into primary education math systems," in *Proc. of the ASEE annual conference*, Jun. 2003, p. 1660.
23. Y. Kafai, K. Peppler, and G. Chin, "High tech programmers in low income communities: Creating a computer culture in a community technology center," in *Proc. of Communities and Technology, C. Steinfeld, B. Pentland, M. Ackermann, and N. Contractor, Eds.*, 2007, pp. 545–562.
24. K. Peppler and Y. Kafai, "Collaboration, computation, and creativity: Media arts practices in urban youth culture," in *Proc. of Computer Supported Collaborative Learning, C. Hmelo-Silver and A. O'Donnell, Eds.*, 2007.
25. K. Peppler and Kafai, "Creative coding: The role of art and programming in the k-12 educational context," in *MIT Media Labs*, 2005.
26. L. Werner, S. Campe, and J. Denner, "Lifelong student engagement in it: Middle school girls + games programming = information technology fluency," in *Proc. of the IASTED*, Oct. 2005, pp. 301–305.
27. S. Campe, L. Werner, and J. Denner, "Information technology fluency for middle school girls," in *Proc. of the IASTED*, Aug. 2005.
28. A. K.-. T. F. C. Committee, *A Model Curriculum for K-12 Computer Science*. ACM: http://csta.acm.org/Curriculum/sub/K-12ModelCurr2ndEd.pdf, Mar. 2003.

29. *Fun Brain, http://www.funbrain.com*. Pearson Education.
30. *First in Math, http://www.firstinmath.com*. Suntex International Inc.
31. H. Ketamo, "mlearning for kindergarten's mathematics teaching," in *Proc. of the IEEE Wkshp. on Wireless and Mobile Tech.*, Aug. 2002, pp. 167–168.
32. N. Shin, C. Norris, and E. Soloway, "Effects of handheld games on students learning in mathematics," in *Proc. of the ACM Conf. on Learning Sciences*, Jun. 2002, pp. 702–708.
33. P. Vahey, D. Tatar, and J. Roschelle, "Leveraging handhelds to increase student learning: Engaging middle school students with the mathematics of change," in *Proc. of the ACM Conf. on Learning Sciences*, Jun. 2004, pp. 553–560.
34. G. B. et. al, "Attitudes toward computers: Validation of a computer attitudes scale," *J. of Educational Computing Research*, vol. 3, no. 2, pp. 207–18, Jun. 1987.
35. *Teach Engineering, http://www.teachengineering.com*. National Science, Technology, Engineering, and Mathematics Education Digital Library.
36. M. Gall, W. Borg, and J. Gall, *Educational Research: An Introduction, 6th Ed.* Allyn and Bacon, 1995.
37. P. Moroz and J. Nash, "Bath county computer attitude scale: A reliability and validity scale," in *Proc. of the ASEE Annual Conf.*, Jun. 1997.
38. T. Crenshaw, E. Chambers, H. Metcalf, and U. Thakkar, "A case study of retention practices at the university of illinois at urbana-champaign," in *Proc. of ACM SIG: Computer Science In Education*, 2008.