2-2018

Bricklayer: Elementary Students Learn Math through Programming and Art

Michelle Friend
University of Nebraska at Omaha, mefriend@unomaha.edu

Bricklayer: Elementary Students Learn Math through Programming and Art

Michelle Friend
University of Nebraska Omaha
Omaha, Nebraska
mefriend@unomaha.edu

Michael Matthews
University of Nebraska Omaha
Omaha, NE
michaelmatthews@unomaha.edu

Victor Winter
University of Nebraska Omaha
Omaha, NE
vwinter@unomaha.edu

Betty Love
University of Nebraska Omaha
Omaha, NE
blove@unomaha.edu

Deanna Moisset
Omaha Public Schools
Omaha, NE
deanna.moisset@ops.org

Ian Goodwin
University of Nebraska Omaha
Omaha, NE
igoodwin@unomaha.edu

ABSTRACT

As computer science becomes more prevalent in the K-12 world, elementary schools are increasingly adopting computing curricula. Computer scientists have recognized the connection between math and computer science, but little work has demonstrated how and whether computer science can support improved learning in math. This paper reports on a project in which elementary students in a gifted program used Bricklayer, a functional programming environment that supports artistic and mathematical expression. A pre- and post-test design demonstrates significant learning gains in coordinate graphing and visual-spatial skills.

KEYWORDS

Elementary School, Functional Programming, Mathematics

ACM Reference Format:
Michelle Friend, Michael Matthews, Victor Winter, Betty Love, Deanna Moisset, and Ian Goodwin. 2018. Bricklayer: Elementary Students Learn Math through Programming and Art. In *Proceedings of The 49th ACM Technical Symposium on Computer Science Education (SIGCSE'18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159515

1 INTRODUCTION

Efforts such as the CSForAll initiative [12] and Code.org have been increasingly successful at promoting the adoption of computer science across schools, particularly in elementary schools. In order to continue justifying instructional time spent on computer science, educational research must demonstrate that the time is effective - that students are indeed making sufficient progress towards instructional goals. Traditionally, language arts and math have been the two cornerstones of elementary curriculum. Computer science most closely aligns with mathematics, yet little research has demonstrated that incorporating computer science improves mathematics outcomes [10].

This paper reports on a project in which elementary school students were provided computer science lessons once per week for ten weeks in a gifted and talented education (GATE) program, and were tested for mathematics outcomes in a pre- and post-test design. The paper first explains the relationship between mathematics and computer science, particularly in elementary school. It then describes the educational intervention, using Bricklayer, a programming-based pedagogical tool that creates artistic artifacts. Finally, the research project is described, including findings related to improvements in students' coordinate graphing knowledge and visual-spatial skills following the intervention. Implications, limitations, and areas for future research are identified.

1.1 Computer science and mathematics

Computer science is in many ways an application of mathematical principles, and computer science educators have long argued the relationship between computing and math [7]. While anecdotal evidence that mathematical skill enhances students' ability to understand computing is reinforced through mathematics prerequisites for high school and college computing courses, there has been little evidence that learning programming actually leads to gains in mathematics performance [8]. This is unsurprising, as transfer between domains requires deep understanding and explicit instructions for how to take ideas from one area (such as programming) and apply them in another (such as mathematics) [1]. There is reason to believe that a strength of computing is the opportunity to apply mathematics skills and topics to a real world setting [9]; for example, programming graphics objects can motivate students' understanding of the coordinate graphing system [3]. Recently, the Bootstrap program has shown evidence that functional programming can successfully support student learning of algebraic functions [10, 11]. However, there remains a dearth of research on the explicit connections between computer science and mathematics, particularly for young children, before the introduction of algebraic reasoning.

Although it has not traditionally been a focus of mathematics curriculum and instruction, spatial reasoning has been gaining attention as an important skill underpinning mathematical and scientific learning. The National Research Council called for explicit

SIGCSE'18, February 21–24, 2018, Baltimore, MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159515

attention to spatial reasoning in not just mathematics but across the entire curriculum, calling it a "major blind spot" in almost all current curricula [2]. A large longitudinal study demonstrated that spatial ability plays a critical role in the development of STEM abilities [13]. In a study of 8-year-old students, Gunderson et al. found a "strong relationship between spatial skills, number line knowledge, and math achievement" [4]. In other words, curriculum that supports student development of spatial skills is likely to support improved academic performance across the curriculum and especially in mathematics [6, 13].

A major topic in upper elementary mathematics instruction is geometry, particularly introducing students to the coordinate plane and extending their early number sense to two- and three-dimensional spaces. For example, the Common Core State Standards for Mathematics includes "Graph points on the coordinate plane to solve real-world and mathematical problems (CC.5.G.A.1 and CC.5.G.A.2)" and "Solve real-world and mathematical problems involving area, surface area, and volume (CC.6.G.A.3)." Further, elementary mathematics curricula are designed to support the underlying mathematical skills and concepts that will help students be successful in algebra, such as generating and analyzing patterns (CC.4.OA.C.5) or writing and interpreting numerical expressions (CC.5.OA.A.1, CC.5.OA.A.2).

2 BRICKLAYER

Bricklayer is an open-source, online educational ecosystem designed to teach coding to people of all ages and coding backgrounds [5, 14]. Designed with a "low-threshold, infinite-ceiling" philosophy, it provides an example-rich and problem-dense domain in which students learn to write programs in the functional programming language SML. When executed, Bricklayer programs can produce LEGO®artifacts, Minecraft artifacts, and even artifacts suitable for 3D printing.

Bricklayer is a collection of interactive web apps and downloadable software, including a Google Blockly-type coding environment called **Bricklayer-lite**, which was used by the students in this project. The pedagogical tool begins with an *unplugged* level that focuses on the exploration and understanding of various mathematical patterns such as tessellations, fractals, and elementary cellular automata - all concepts that even young students can understand through drawing exercises. Bricklayer's five coding levels provide instruction and challenges of increasing complexity, beginning with two-dimensional pixel art and progressing to the creation of three-dimensional patterns such as a Menger sponge and even more complex shapes such as those shown in Figure 1. Similar to other environments like Scratch, Bricklayer-lite has a block-based programming system and displays the output of the code, as shown in Figures 2, 3 and 4, allowing users to see their code and the output in different panes of the same window. Bricklayer itself is a text-based language with the same syntax as Bricklayer-lite, making it easy for students to transition between environments.

2.1 Bricklayer and Art

One notable element of Bricklayer is its focus on art. Although the communication with the computer takes place through coding, and the underlying concepts are mathematical such as placing objects at particular coordinate points in space, the output is artistic in nature. After introducing students to new commands and concepts, the curriculum encourages students to use creativity to design their own artifacts such as those shown in Figure 1. While early tasks are very specific, directing users to place colored bricks at particular points, users are quickly pushed to make pictures or objects based on their own artistic interests. Examples of art created by participants in this study are seen in Figures 2, 3 and 4.

Most introductory programming environments have some 'theme'; in many cases, particularly environments designed for young children, this theme is game-based. For example, both Code.org and Swift Playground lessons provide game-like challenges for users to solve. Although there are challenges built into the Bricklayer pedagogical suite, the goals are not game-like, but artistic, to create pictures and objects. In this way, Bricklayer is designed to appeal both to "STEM-y" students who are traditionally drawn to programming, and also to students with an artistic bent for whom STEM hobbies may not be intrinsically interesting.

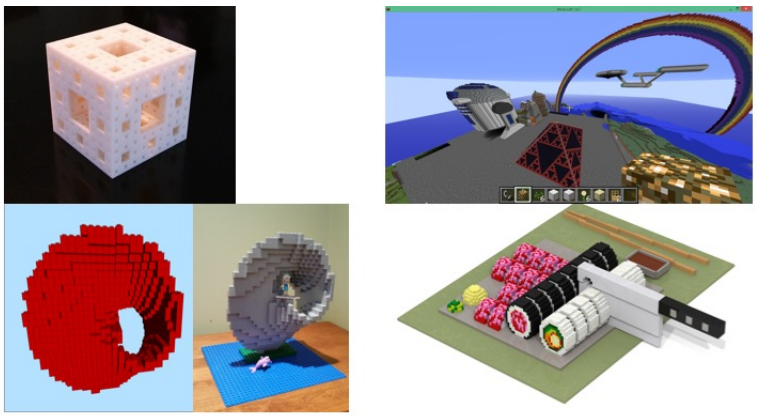


Figure 1: Artifacts created using Bricklayer.

A significant portion of the Bricklayer ecosystem has been developed specifically to help novices, especially primary school children, learn how to code. Bricklayer has been used with students from elementary school through college courses, and is currently being used in over 75 elementary, middle, and high schools. This paper reports on a project in which elementary school students used Bricklayer in a ten-week workshop one hour per week at school.

3 METHOD

This study was designed to investigate the impact of teaching Bricklayer to elementary school students. It particularly investigated whether programming in Bricklayer would promote improved mathematics skills. Thus the research question was whether students would show improvement in coordinate graphing, spatial skills, and functions following a Bricklayer unit in their GATE program.

3.1 Participants

The participants in this study represent a convenience sample. Participants were students who had been identified by the school district for GATE (gifted) services. The GATE program has considerably more leeway in implementing innovative curriculum than standard classrooms which must adhere to district-adopted curriculum. Students in the GATE program are pulled out of their regular classes for approximately one hour per week to receive instruction and resources from a trained facilitator.

Pre- and post-tests were collected from a total of 66 students in grades three through six at nine different schools in the district. Four participants were excluded for lack of consent, resulting in data from 62 participants. Data were anonymized for analysis. Tables 1 - 3 show the demographic breakdown of participants by grade level, school attended, and gender.

Table 1: Participants by grade level

Grade	3	4	5	6
Number of participants	10	14	23	15

Table 2: Number of participants at each school

School	A	B	C	D	E	F	G	H	J
Number of participants	11	8	5	1	11	6	10	3	7

Table 3: Participants by gender

Gender	Female	Male	Declined to state
Number of participants	29	29	4

3.2 Procedure

The six GATE facilitators for the nine schools were trained in Bricklayer over two evenings before teaching the students. Facilitators were trained prior to spring break and implemented the program following spring break and through mid-May; students spent no more than ten weeks using Bricklayer as a part of this program. A Piazza online community was established for the facilitators to be able to ask questions as they encountered them. The facilitators were introduced to a ten-week curriculum designed for after-school programs, and used this curriculum as a guide for their implementations.

Students took the tests on paper during their regular GATE time; the tests were administered by GATE teachers and were collected and maintained by the GATE coordinator for the district, who delivered the tests to the researchers at the conclusion of the school year.

During the Bricklayer lessons, student progressed through Bricklayer levels of increasing complexity. Examples of student work created during the project are shown in Figures 2, 3 and 4. These show a progression in conceptual complexity. Figure 2 represents level one: pixel art, created by placing each block individually. Figure 3 represents level two, and demonstrates a function which creates a simple repeating pattern. Figure 4 is a more complex function that creates a non-repeating pixelated image, from the third level of Bricklayer. Each figure shows both a Bricklayer-lite code snippet on the left, and the outcome as displayed on the screen, on the right side of the figure.

Figure 2: Early art created by a participant in Level 1

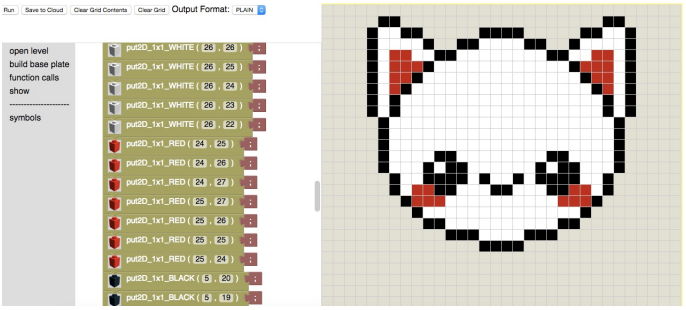
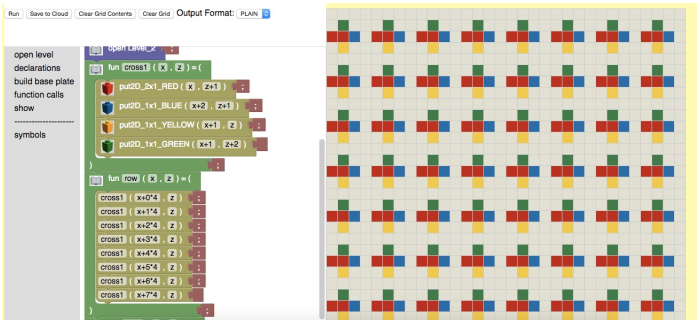


Figure 3: Intermediate art created by a participant in Level 2



3.3 Measures

Pre- and post-test mathematics questions were isomorphic: slightly different versions of the same questions. For example, the mental

Figure 4: More complex programming by a participant in Level 3

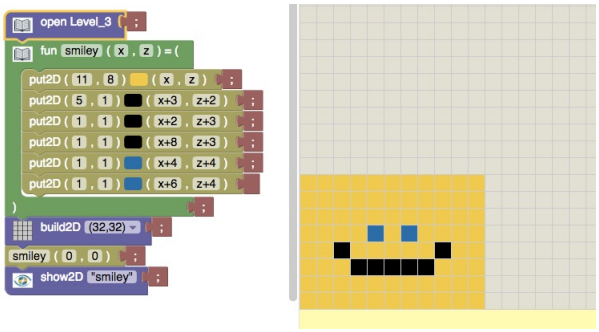
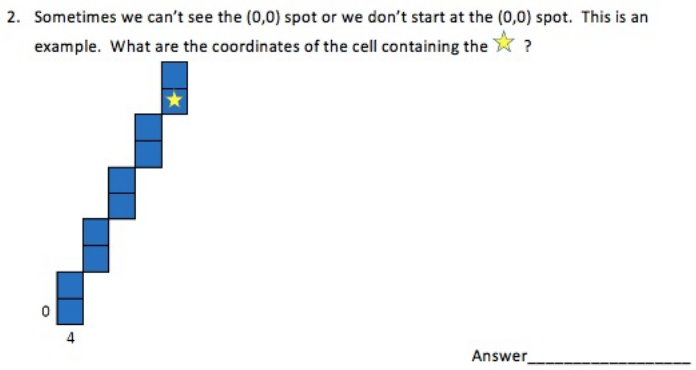


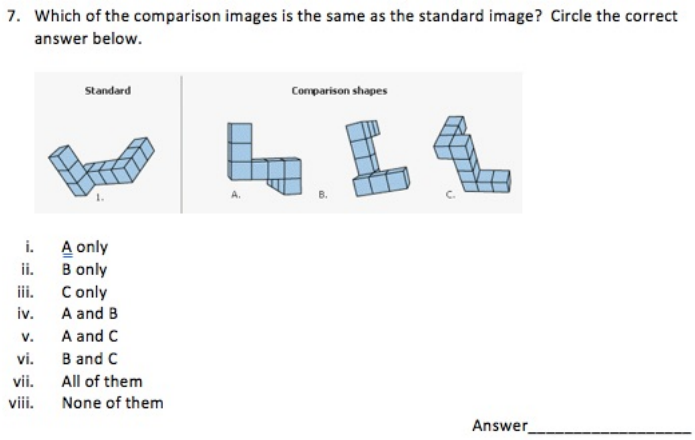
Figure 5: Sample coordinate graphing question



rotation question shown in Figure 6 had a different shape on the pre-test than the post-test. The other test questions were identical between the pre- and post-test. The following measures were analyzed:

- **Demographics** Participants were asked their *grade*, *age*, *gender identification*, and *ethnicity*. The first two pieces of demographic information were at the beginning of the test, and the last two at the end of the test in order to decrease the likelihood of stereotype threat. The grade level reported on the post-test was used for analysis because not all students answered the question on the pre-test. Four students did not report their gender. The GATE coordinator identified the school associated with each set of tests.
- **Attitudes about math & computers** Participants were asked to respond to ten statements using a five-point Likert scale. The statements broke into three constructs: *Computer Interest* such as "Computers are interesting to me", *Computer Confidence* such as "I am good with computers", and *Math Confidence* such as "I feel confident about my ability to do math problems." For each construct, responses were averaged and normed to result in a single score between 1 and 5.
- **Mathematics** A mathematics score represented how many questions were answered correctly out of twelve; unanswered questions were marked wrong. These were further divided into three subsections:
  - Three questions about *coordinate graphing* prompted students to identify the coordinates of a cell containing a particular symbol such as that shown in Figure 5
  - Four questions allowed students to demonstrate *visual-spatial ability* such as the task shown in Figure 6 that requires mentally rotating an image.
  - Five questions inquired into students' understanding of mathematical *functions*. For example, "A mathematical function can sometimes be written in words like this. Let's say I have a function called 'Almost double'. Here is what almost double does. Take any number, double it, then take away 1. What would you get if you used 7 with the 'almost double' function?"

Figure 6: Sample mental rotation question



3.4 Data Analysis

At the end of the school year, the tests were returned to the researchers. One author entered the responses into a spreadsheet. The data was analyzed using R. Because the data was non-normal, Yuen's test for trimmed means was used to compare means.

The analysis addressed the following hypotheses:

(1) Average score on the test of mathematics would increase between pre-test and post-test.
(2) Average score on the coordinate graphing mathematics subsection would increase between pre-test and post-test.
(3) Average score on the visual-spatial mathematics sub-section would increase between pre-test and post-test.
(4) Average score on the functions mathematics sub-section would increase between pre-test and post-test.
(5) There would be no significant difference between boys and girls in terms of improvement on the tests
(6) There would be no significant difference between schools in terms of improvement on the tests
(7) Average score increase on the test of mathematics would be different for different grades. Either a ceiling effect would occur where older students (e.g. grade 6) would already have

mastered the skills and so a smaller change would occur, or the intervention would be less effective with younger students (e.g. grade 3) who had not yet learned coordinate graphing and functions, or both.

(8) Students would increase their computing interest and confidence and their mathematics confidence

4 FINDINGS

4.1 Overall Score

There was a significant increase in average test scores between the pre- and post-test across all questions from 36% correct (se = 2.6) on the pre-test to 47% correct (se = 2.6) on the post-test on average ($t(37) = 5.51$, $p<.0001$), representing a large effect ($r = .67$). This confirmed the first hypothesis.

4.2 Coordinate Graphing

There was a significant increase in scores on the coordinate graphing section of the test. On average, scores went from 26% correct (se = 4.2) on the pre-test to 42% correct (se = 4.0) on the post-test ($t(37) = 3.84$, $p = 0.0005$) with a moderate effect size ($r = .53$). The smaller significance and effect size despite a larger change in percentage correct is due to the much smaller number of questions: only three rather than 12 on the composite score. This confirmed the second hypothesis.

4.3 Visual-Spatial Skill

There was a similarly large increase in scores on the visual-spatial section of the test. On average, scores went from 24% correct (se = 2.9) on the pre-test to 52% correct (se = 3.6) on the post-test ($t(37) = 4.30$, $p = 0.0001$) with a moderate effect size ($r = .53$). Like the coordinate graphing section, this represents that the visual-spatial skill section encompassed only four questions. This confirmed the third hypothesis.

4.4 Functions

On the functions section of the test, there was a small and non-significant increase in scores from 43% correct (se = 3.4) on the pre-test to 45% correct (se = 3.7) on the post-test ($t(37) = 0.82$, $p = 0.41$, $r = 0.13$). Thus, the fourth hypothesis was not confirmed, though the small improvement contributed to the improvement seen on the overall score.

4.5 Demographic differences

An Analysis of Variance (ANOVA) was conducted to compare score differences among students with various demographic differences. No significant difference was found for changes in score between pre- and post-tests for students in different grades ($F = .32$, $p = .57$) or different genders ($F = ..03$, $p = .97$). Although there was a small difference between students in different schools ($F = 2.74$, $p = .01$) the number of students at each school was small, and further research is necessary to draw any conclusions about differences between schools. Thus hypotheses five was confirmed (that there would be no difference between genders) and hypotheses six and seven were not confirmed.

4.6 Attitudes

Because Bricklayer provides an engaging, creative opportunity for students to learn programming and math, the hypothesis was that after engaging in the Bricklayer experience, students would increase their computing interest and confidence and their mathematics confidence. Although no formal qualitative data was collected, there was anecdotal evidence that students were highly engaged and enjoyed using Bricklayer. One of the teachers reported that she gave the students an opportunity to go on to another desirable curricular experience or continue using Bricklayer and that the students unanimously chose to continue using Bricklayer. However, the survey results do not demonstrate a significant increase in attitudes. Computer interest increased from 4.21 (se = .1) to 4.24 (se = .11) ($t = 0.32$, $p = .75$) on a five-point scale, computer confidence increased from 4.17 (se = .09) to 4.24 (se = .08) ($t = 0.79$, $p = .43$), and math confidence increased from 4.17 (se = .09) to 4.24 (se = .08) ($t = 0.79$, $p = .43$). Thus, hypothesis eight was not confirmed.

5 DISCUSSION

At the most basic level, this paper reports upon a successful proof of concept: that elementary students can use Bricklayer-lite to create artistic artifacts of their own design. It further suggests that Bricklayer can support students' mathematics learning.

Participants' improvement in their ability to answer questions about coordinate graphing and visual-spatial skills following the Bricklayer course is extremely promising. Although the ability to identify locations on a coordinate graph is not a traditional computer science topic, it is a fundamental skill required in elementary math standards, and is represented in graphics programming. Students' improvement in their visual-spatial ability is not currently strongly represented in math or computer science standards, but prior research has demonstrated that it is an important skill associated with strong outcomes in STEM.

Despite successfully writing functions in order to run their programs in the higher levels (2 and 3), participants did not demonstrate improvement from pre- to post-test on the functional reasoning sub-section. Further research is warranted to understand why this hypothesis was not confirmed. One possibility is that, because the functions questions were at the end of the test, students ran out of time and could not complete them. Another possibility is that in only ten sessions with Bricklayer they were not able to engage with enough of the higher levels of the curriculum that would substantially strengthen their functional reasoning. No formal record was kept of how far students progressed in the curriculum. The curriculum has 5 levels with exposure to functional reasoning increasing dramatically in levels 3, 4, and especially 5. Feedback from the teachers indicate that many of the 62 participants only made it to levels 2 or 3 material, and thus had limited exposure to functional reasoning. It may be that more exposure to Bricklayer functional reasoning would lead to more general mathematical functional reasoning. One of our in-progress initiatives involves working with Master elementary teachers with experience teaching coding to develop Bricklayer coding projects that correlate well with elementary mathematics content standards and more advanced topics like functional reasoning.

As in most educational research, causality cannot be applied to this ten-week program. It is certainly possible that other experiences, including students' regular classroom math classes and lessons were responsible for the significant increase in performance on the post-test. However, the students were distributed across nine schools and four grades as shown in Tables 1 and 2. The heterogeneity of participants' outside-workshop experience suggests that the impressive changes are likely associated with using Bricklayer, particularly since Bricklayer was explicitly designed to support the skills which were assessed. This is not itself a weakness - the hypotheses were based on prior research and understanding of the likely gains, and were objectively assessed using isomorphic questions. To confirm a causal relationship, experimental conditions must be set up in which control groups will complete a different mathematical workshop, teaching similar skills without the use of functional programming. However, we feel comfortable suggesting that this research is worthwhile, given the strength of the findings presented here.

With the increasing push to cover more and more topics in elementary grades, finding a way to address multiple content strands is one of the great potentials of Bricklayer. A significant limitation is that this work reports upon workshops run with students identified as gifted - that is, more intelligent than the average public school population. This limits the generalizability of the results, and we hope to expand the work to implement Bricklayer across math classes throughout the district rather than limiting it to pull-out classes. It would be of interest to see how exposure to learning coding with Bricklayer will lead to similar changes in a general population of elementary students. Elementary class time is increasingly constrained, particularly with the ever-present threat of standardized tests that students must pass. By demonstrating that Bricklayer has the potential to improve students' performance on these tests, we hope that teachers, principals, and district personnel would be open to incorporating Bricklayer into curricular programming more broadly.

One of the reasons touted for getting programming into mathematics classes is that it will increase student engagement - students will think it is fun and pay more attention. Anecdotal evidence does support this idea, as students indicate more interest in some ideas such as coordinate graphing when they are motivated by a "real world" context to use what is otherwise a dry practice. This was the case with Bricklayer, where teacher reports suggest that students enjoyed using it. However, the survey of student attitudes indicate that there was not a significant change in student interest or confidence about computing and math. This is likely because of a ceiling effect - the students were generally very positive on the post-test. This is not surprising from an academically-strong group, and a group who is young enough to not yet have "turned off" to mathematics or programming. Further investigation of the role of student attitudes on learning and engagement will continue to shed light into when and how such interventions may be most effective.

Recent efforts to incorporate coding and computer science across all schools have focused on the positive outcomes of computer science. This paper reports upon a successful implementation of an environment that not only teaches programming but also demonstrates mathematics learning as well. Being able to learn coding skills while at the same time improve on students' understanding of coordinates and develop students' spatial reasoning at the same time is a win-win.

REFERENCES

[1] J D Bransford and D L Schwartz. 1999. Rethinking Transfer: A Simple Proposal With Multiple Implications. *Review of Research in Education* 24, 1 (1999), 61–100. http://rre.sagepub.com/cgi/doi/10.3102/0091732X024001061

[2] Committee on Support for Thinking Spatially. 2006. *Learning to Think Spatially*. National Academies Press. DOI: https://doi.org/10.17226/11019

[3] Robert Cutler and Michelle Friend Hutton. 2010. Digitizing Data : Computational Thinking for Middle School Students through Computer Graphics. In *Proceedings of the 31st Annual Conference of the European Association for Computer Graphics (Eurographics)*. 1–8.

[4] Elizabeth A. Gunderson, Gerardo Ramirez, Sian L. Beilock, and Susan C. Levine. 2012. The relation between spatial skill and early number knowledge: The role of the linear number line. *Developmental Psychology* 48, 5 (2012), 1229–1241. DOI: https://doi.org/10.1037/a0027433

[5] Betty Love. 2016. Creating an Environment in which Elementary Educators Can Teach Coding. *ACM SIGCHI Conference on Interaction Design and Children 2016* (2016), 643–648. DOI: https://doi.org/10.1145/2930674.2936008

[6] N. S. Newcombe. 2013. Seeing Relationships. Using Spatial Tinking to Teach Science, Mathematics, and Social Studies. *American Educator* (2013), 26–32.

[7] Seymour Papert. 1971. Teaching children to be mathematicians vs teaching about mathematics. *International journal of mathematical education in science and technolog* 3, 3 (1971), 249–262. DOI: https://doi.org/10.1080/0020739700030306

[8] Roy D. Pea and D. Midian Kurland. 1984. On the Cognitive Effects of Learning Computer Programming. *New Ideas in Psychology* 2, 2 (1984), 137–168.

[9] Jeremy M Roschelle, Roy D Pea, Christopher M Hoadley, Douglas N Gordin, and Barbara M Means. 2000. Changing How and What Children Learn in School with Computer-Based Technologies. *The Future of Children* 10, 2 (2000).

[10] Emmanuel Schanzer, Kathi Fisler, Shriram Krishnamurthi, and Matthias Felleisen. 2015. Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15* (2015), 616–621. DOI: https://doi.org/10.1145/2676723.2677238

[11] Emmanuel Tanenbaum Schanzer. 2015. *Algebraic Functions, Computer Programming, and the Challenge of Transfer*. Doctoral dissertation. Harvard Graduate School of Education. DOI: https://doi.org/10.1016/S0140-6736(09)60087-8

[12] Megan Smith. 2016. CS For All. (2016). https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

[13] Jonathan Wai, David Lubinski, and Camilla P. Benbow. 2009. Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology* 101, 4 (2009), 817–835. DOI: https://doi.org/10.1037/a0016127

[14] Victor Winter, Betty Love, and Cindy Corritore. 2016. The Bricklayer Ecosystem - Art, Math, and Code. *Electronic Proceedings in Theoretical Computer Science* 230 (2016), 47–61. DOI: https://doi.org/10.4204/EPTCS.230.4 arXiv:1611.09472

ACKNOWLEDGMENTS

The authors would like to thank the participating GATE teachers and students.