5-2022

A Computer Programming Intervention for Second Grade Math Students

Eric B. Bagley
Utah State University

A COMPUTER PROGRAMMING INTERVENTION FOR SECOND GRADE MATH STUDENTS

by

Eric B. Bagley

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

John M. Edwards, Ph.D. Major Professor	Jessica F. Shumway, Ph.D. Committee Member
Vicki H. Allan, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2021

ABSTRACT

A Computer Programming Intervention for Second Grade Math Students

by

Eric B. Bagley, Master of Science

Utah State University, 2021

Major Professor: John M. Edwards, Ph.D.
Department: Computer Science

The multiplication algorithms that we teach elementary students are designed to be efficient, but the conceptual understanding of these algorithms is usually hidden behind this efficiency. For example, one intuitive meaning of multiplication is that of iterated addition. In this paper, we investigate the ways a visual programming activity leverages the concept of iteration to learn multiplication. We report results of a mixed-methods study on second-grade students' use of visual programming and iteration to set up and solve multiplication story problems. We found evidence that participants gained improved intuition of multiplication as repeated addition and that they made important connections between the programming blocks and multiplication story problems.

(37 pages)

PUBLIC ABSTRACT

A Computer Programming Intervention for Second Grade Math Students

Eric B. Bagley

The multiplication algorithms taught to elementary students are made to help students find answers quickly, but why the algorithm works and how it relates to multiplication is not widely known. For example, one intuitive meaning of multiplication is that of iterated, or, repeated, addition. In this paper, we look at the ways a visual, block-based, programming activity uses the concept of iteration to help second-graders learn multiplication. The results of the study observing second-grade students use visual programming and iteration to set up and solve multiplication story problems. We found that generally students enjoyed these activities and found them helpful during the learning process.

To those who just use a calculator...

ACKNOWLEDGMENTS

A big thank you to advisors, friends, and family who helped every step of the way.

Eric B. Bagley

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

"Sometimes I try and get the answer, but I ask my mom for help because she has a calculator on the computer." Most of the time, this second-grader's response reflects our own – if you need to multiply two numbers, just use a calculator. Why then do we learn how to do multiplication without a calculator? The standard answer is that you never know when you won't have a calculator. This argument, however, is becoming less relevant with the increasing ubiquity and accessibility of computing devices.

Before the Common Core State Standards (CCSS) were introduced in the United States [1], there was an ongoing debate as to whether or not students should learn how to read and write in cursive [2]. When ink bottles and quills were used, writing in cursive helped avoid unnecessary ink splotches and broken quills. Cursive handwriting continued to be used because it required less lifting of the pen, arguably making writing more efficient. One of the reasons cursive is not widely taught in schools today is that most of our written communication is prepared with a computer, rendering the teaching of cursive handwriting nearly obsolete.

Could the same be true for doing multiplication? Should we just use the calculators on our phones? Before answering the question we need to make an important distinction between understanding what multiplication is and being able to efficiently compute products by hand. We suggest that it is more important to be able to set up a multiplication story problem than to be able to perform the computation by hand. Nevertheless, computing products by hand can be a worthwhile approach to learning multiplication. Several algorithms help us multiply by hand. Most of them are efficient but they do not reflect an intuitive sense that *multiplication is simply iterated addition.* Computing a product using simple iteration, or, repetition, was avoided because it is slow and inefficient when done by hand. But now, with advancements in programming languages for children, including visual

programming languages, even a second-grader can set up a computer program to compute the iterations for them.

In this thesis, we suggest that teaching second-grade children to multiply using an iteration-based algorithm with accompanying visualizations may give them a stronger intuition of multiplication as repeated addition and may improve their ability to set up solutions to story problems. As mentioned, the two advancements that enable this approach are 1) the availability of computing devices rendering the need to multiply by hand less important and 2) the maturity of programming languages for children, allowing them to write programs that use iteration.

According to the United States Common Core State Standards for Mathematics (CCSSM), the main focus areas for mathematics in second-grade include extending understanding of base-ten notation and building fluency with addition and subtraction. In third grade, the main focus areas include developing an understanding of multiplication and division and strategies for multiplication and division within 100 [3].

Our work focuses on the transition from addition to multiplication and the use of iteration and visual programming as teaching methods. Our research question is, *In what ways does iteration influence students' learning of multiplication during visual programming activities?* We want to understand whether the concept of iteration leads to a deeper understanding of multiplication. Visual programming can act as an appropriate tool to set up multiplication as problems of repeated addition. We designed and constructed a web-based visual programming language then conducted a think-aloud study with second-graders (IRB #11372) during which we collected site usage data, observed student interactions with the web application, and asked them various questions throughout the process. We found that visual programming appears to be an effective way to help students learn the iterative process of multiplication.

This paper proceeds as follows: we discuss related work (Chapter 2), then methods (Chapter 3), followed by results and discussion (Chapter 4), and finish with conclusions (Chapter 5).

CHAPTER 2

Related Work

Programming has been used to supplement instruction since the late 1960s [4, 5]. In recent years, perhaps with Seymour Papert's book acting as a catalyst [6], learning environments and outcomes have been enhanced by technologies and tools that utilize appropriate levels of computational thinking [5]. This is true for multiple areas of discipline such as English, Language Arts, Mathematics, and Computer Programming [5].

2.1 Iteration

Within mathematics, iteration is used initially to teach the concept of multiplication using ideas such as repeated addition, skip counting, and array counting. Due to its tedious nature, iteration is quickly replaced by methods that favor efficiency over understanding. However, if students do not understand that multiplication is simply iterated addition, it can become difficult for them to set up story problems and recognize where multiplication can be used in real-life situations. There are many different methods to assist teachers in teaching multiplication and other math topics. Some of these methods include virtual and physical manipulatives as well as other learning activities and math games [7, 8]. All of these leverage different strengths to assist in the learning process. In more recent years, visual programming has entered the arena.

2.2 Visual Programming and Other Teaching Tools

Visual programming, or programming languages that provide graphical objects that compose a program, has been used to assist young or novice programmers in learning various concepts of computational thinking and computer science. For example, Scratch, a visual programming environment, targets eight to sixteen-year-olds [9]. Some languages and frameworks have targeted younger audiences of five to seven-year-olds [10, 11]. Recently,

students as young as three years old have been the subjects of visual programming research [12]. In these studies, visual programming was used both to teach the skill of programming and as an exploratory teaching tool for cross-curricular subjects [5, 10, 13, 14].

Other studies used coding toys that allow children to construct code by either manual input on the toy or through physical blocks and other tactile means [15–18]. Among the studies reviewed, there were not any that analyzed guided, incremental visual programming activities to teach and visualize the concept of iteration in multiplication.

CHAPTER 3

Methods

This section describes the design of our mixed-methods study in which we observe second-graders using iteration and visual programming to reinforce an intuitive understanding of multiplication. We also describe the programming language we developed for use in the study.

3.1 Study Design

To answer our research question, we used a mixed-methods approach by collecting both quantitative and qualitative data from 11 student participants. Participants were recruited from two second-grade classes in a university laboratory school in the Intermountain West. The study was divided into three parts – pre-test, learning module, and post-test – all of which were completed during a single 25-45 minute session with each child. Seven of the 11 students were observed and interviewed and the remaining four students completed the activity without being observed. During the observations, students were asked to think aloud. Most correspondence was done virtually over Zoom, including observation and interviews. For all students, various forms of quantitative data were gathered, including the total time it took for the student to complete each activity, the number of times they executed the code, and other interactions with the coding canvas.

3.1.1 Tests

Both pre and post-tests were composed of three types of questions: those relating to the second-grade CCSSM standard 2.OA.4 (use addition to find the total number of objects arranged in rectangular arrays with up to 5 rows and up to 5 columns; write an equation to express the total as a sum of equal addends), those corresponding to the third-grade CCSSM standard 3.OA.1 (interpret the products of whole numbers, such as interpreting 5×7 as

the total number of objects in 5 groups of 7 objects each.), and unrelated third-grade level questions to act as a control. The pre and post-tests were composed of eight questions: five related to multiplication and three did not. Test questions were presented to the students one at a time. Audio for each question was available. For a list of test questions, their associated standard, and other details see appendix A.

3.1.2 Activities

The learning module was a series of guided visual programming activities that focus on the transition from addition to multiplication. Activity difficulty was incremental and each offered appropriate scaffolding to help the student succeed. Audio clips were prepared for activity prompts, hints, and completion. The first activity required that students only run the code, the second activity asked the students to change the Repeat Item Block, the next activity slightly adjusted the prompt, and so on as indicated by Table 3.1. For a list of activity prompts and other details see appendix B.

Activities	Objective
P1	Click the run button
A1 - A3	Modify repeat item block value process.
B1 - B3	Drag repeat item block onto the canvas.
C1a - C1b	Gradual ease into Drag the repeat row block onto the canvas.
C2a - C2c	Gradual ease into modifying the repeat row block
D1 - D3	Comprehensive activities involving all previous learning

Table 3.1: Activities

3.1.3 The Programming Environment

We created a simple programming environment. The programing language was built with Blockly, Google's visual block-based programming framework [19]. Blockly allows developers to create blocks that generate syntactically correct code. When blocks are as-

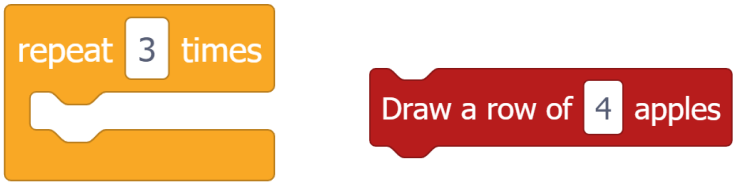

Fig. 3.1: The orange block on the left (Repeat Row) loops over and executes its content the specified number of times. The red block on the right (Repeat Item) draws the item a specified number of times. In this case an apple is drawn four times.

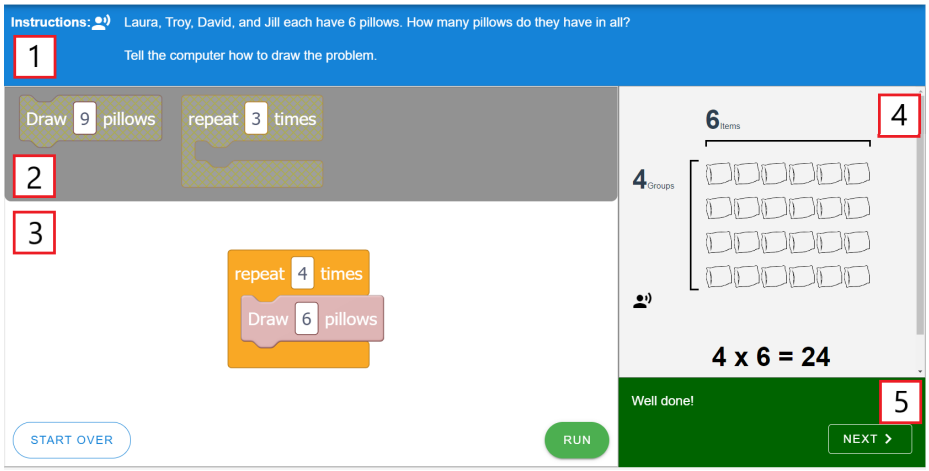

Fig. 3.2: Pieces of the application include 1) the activity prompt, 2) block pallet, 3) block canvas, 4) multiplication matrix, 5) message center. The orange repeat block loops over and executes its content the specified number of times. The purple block (Repeat item) draws the item a specified number of times. In this case four rows of six pillows pillow are drawn. If the students did not correctly address the prompt, a hint message would display below the multiplication matrix.

sembled they can be executed step-by-step. As seen in Figure 3.1, two blocks were available for participants to use in our study. The items drawn by the Repeat Item Block was changed throughout the activities. Different objects included balls, shapes, trees, pillows, and books, to name a few.

As seen in Figure 3.2, key components of the app design include activity prompt, block canvas, block pallet, multiplication matrix, and message center. The prompt introduces the activity and describes what to do, the block pallet contains the possible blocks that can be used, the block canvas is where the blocks are assembled into instructions, the multiplication matrix visualizes code execution by showing the number of groups and items in each group

(see Figure 3.3), and the message center displays either a grey hint message or a green success message. Audio was used to read test questions, activity prompts, provide hints, and indicate success (see Figure 3.4).

Fig. 3.3: Blockly allows for step-by-step execution. This image shows that the apples were drawn one by one.

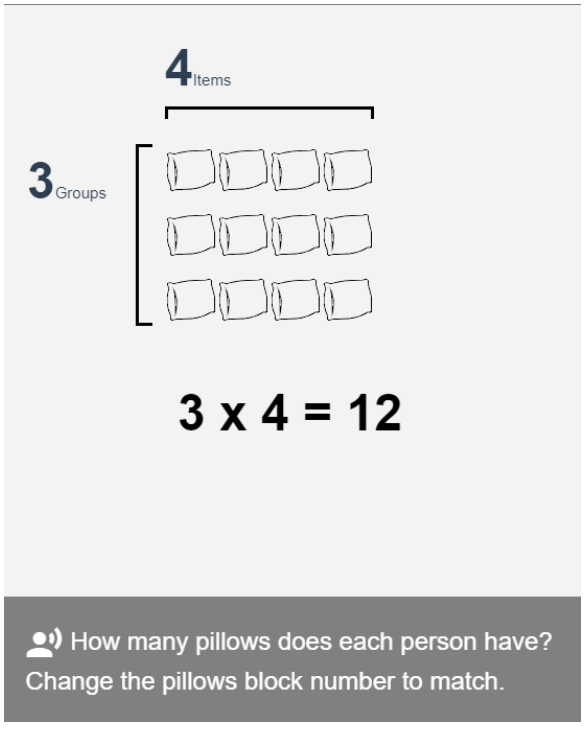

Fig. 3.4: If the students did not fully address the prompt, a hint message would display below the multiplication matrix.

3.2 Data Collection and Analysis

Qualitative data were gathered while the students were observed virtually through Zoom. Participants were asked to think aloud while answering test questions and while working through the activities. They were also prompted throughout the study to explain certain aspects of the interface or clarify something they said. At the end of the session, they were asked if they enjoyed the activity and whether or not it helped them learn something new. If parents were present, they were asked to share their insights on the activities and their child's performance.

During the test questions and each activity, quantitative data were gathered. Test data included timing information and answers. Activity data included timing information, number of attempts made, and other interactions with the coding canvas.

After each session, audio was transcribed and quantitative data analysis was used to determine which areas of the intervention should be analyzed in more depth.

Although every activity was slightly more difficult than the activity preceding it, several activities were noticeably more difficult because 1) they introduced a new programming concept or 2) scaffolding, or some form of additional support, was reduced. We will refer to these as "increment" activities. More information is given on this in the results section of the Thesis.

MAXQDA, a qualitative and mixed methods analysis software, was used to analyze increment activities and test questions for all participants. We developed and assigned the code labels during this analysis. Initially, transcripts and videos were tagged with the related context (e.g., "Activity A-1", or "question zach-stickers" and "pre-test"), then common phrases and themes were noted. After gathering a number of common sayings (e.g., "I do not know", "I got this", or "count the rows"), common programming references (e.g.,"repeat X times", "Scratch") and mathematics topics (e.g., "adding", "times/multiplication"), the participants' general actions were noted (e.g., "listening", "debugging", "building", "overcame struggle"). Various code labels were then combined and placed into different code groups as depicted in Table 3.2.

Class	Code
Problem Solving	Uses an iterative process (skip counting, repeated addition, etc.).
	Uses memorized times tables.
	Uses different structures (programming blocks, physical manipulatives, etc.).
	Requests help from researcher or parent.
Programming Abilities	Understands function of programming blocks.
	Attempts to modify the wrong block with a correct value.
	Student does not understand the functionality of a programming block.
	Student verbally describes the block functionality in an accurate way.
	Student verbally describes the functionalities of a block incorrectly.
	The user interface (UI) is confusing in some way for the student or the student expected the UI to do one thing and something else happened.
Previous Experience	Student references prior experience with multiplication.
	Student references prior programming experience (scratch, code blocks, etc.).
	Student mentions prior experience with a different math game (Prodigy).
Intervention Attitude	Intervention was perceived as helpful.
	Intervention was not perceived as either helpful or unhelpful.
	Intervention was perceived as not helpful.
Math Attitude	Positive perception toward math.
	Neutral perception toward math.
	Negative perception toward math.
Math Abilities	Student is guessing the answer before it is given.
	Student misunderstood the prompt.
Incorrect Answers	Student performed an iterative process incorrectly.
	Student confused multiplication and addition.
	Student formulated the problem incorrectly
	Student used answer field as work space.
	It was unclear why the student answered the question incorrectly,

Table 3.2: Code System

CHAPTER 4

Results and Discussion

The results of our study showed that 1) students were observed verbalizing their use of iteration while solving multiplication problems, 2) students' perspectives on math and programming were generally positive, and 3) students succeeded in manipulating the programming environment effectively.

4.1 Thinking Iteratively

We first explored ways in which students use the concept of iteration to learn multiplication during a visual programming activity. Students did not mention the term "iteration" specifically. However, a common theme across both the tests and the programming activities was the use of iteration. The main iterative processes used to solve multiplication problems observed during the study included some form of skip counting (e.g. counting by fives: 5, 10, 15...) or repeated addition. They were used by all of the students at least once. One student, Emily (names have been changed), relied on iteration more than the other students. The following three quotes are representative of her approach to solving the problems:

> [when computing 3x5] 5, 10, 15...
>
> I know that four plus four is eight. Four plus eight is 12. Four plus 12 would be 16. The answer is 16.
>
> Eight plus eight equals 16. Four plus 16 is 20. So, plus four is 24. So it has to be 24.

Using iteration, Emily got the correct answers on every pre-test and post-test question except one: on one pre-test problem she added an extra iteration on one question. Initial reaction to Emily's approach might be that she is not as advanced as other students because

she is using repeated addition rather than memorized times tables. However, one could argue that Emily has a strong understanding of multiplication – she used a first-principles approach to compute it rather than memorized tables. She understands what multiplication is at its core. Of course, because Emily used iteration she made a simple mistake and got one problem wrong in the pre-test. Visual programming can assist in mitigating human error and computing large numbers of repeated additions.

This demonstrates one advantage of using technology as a tool in the learning process. It can provide exercises and activities that have functional value (such as finding an answer to a multiplication problem) and still demonstrate the underlying behavior of a concept (such as iteration found in multiplication). It does not need to sacrifice understanding for efficiency.

This understanding of multiplication was evident when students used language such as "repeat", "count", "times", and "adding" during the tests and while using the programming blocks.

> *Ian:* I'm *adding* threes together and then I'm getting my [answer].
>
> *Emily:* Each [row] had five desks so we have to *count* 5, 10, 15.
>
> *Emily:* I knew that [a pod] has six peas...so I had to *repeat* two times.

As described in Section 3, the visual programming activities used two looping code blocks to draw objects on the page. See Figure 3.2. The item looping block determined how many items to put in one row. The row looping block determined how many rows to draw. By the end of the activities, all of the students could accurately describe the function of both of these code blocks. Owen said it simply, stating there was a "number" block (the repeat item block) and a "multiplication" block (the repeat row block). Two other students explained it as follows:

> *Matias*: [The repeat row block] says repeat three times. So, if you [change] this [the input of the repeat item block], and you put this [the repeat item block] inside there [the repeat row block] it would repeat it three times.

> *Lance*: If I put the three in here [the repeat row block], then it will...do three times five, which will equal 15.

In summary, the students used iteration to solve multiplication problems by skip counting and repeated addition. They also used the programming blocks to aid them in understanding and solving the problems.

4.2 Student Perspective Toward Iteration

While solving problem C-2c students were asked to draw four rows of trees instead of three (as in the previous problem). Owen paused after listening to the prompt.

> What? Oh, four rows. That means repeat four. Ah, this is not tricking me. That's easy!

Owen found it easy to repeat something four times. Removing the complexities of multiplication algorithms and other teaching methods appears to allow the student to focus on multiplication as repeated addition.

Ian turned many of the activities into a game. First, he would assemble the blocks and update their input. Before he ran his code he said what he thought the answer was. For example, on activity D-2 the prompt reads: "Josh cut the cake at a party. He made 4 rows. Each row had 5 pieces of cake. Tell the computer how to draw the problem." Ian correctly calculated the correct answer before running his code. "This is gonna be 20." When asked how he knew that, he said, "I know three times five equals 15. So just add another five."

On the next activity, D-3, students were given the following prompt: "Laura, Troy, David, and Jill each have six pillows. How many pillows do they have in all?" Ian changed the block inputs (the repeat row block to a three, and the repeat item block to a six) and assembled the blocks. He then said that the answer would be 18 and ran his code. In this case, he initially did not extract the correct information from the problem because he miscounted the number of people. The program then told him that he did not fully address the prompt and gave him the hint directing him to look at how many people were in the problem. He then changed the three to a four and successfully completed the problem.

Running the algorithm to get an answer is one task. Extracting information from a word problem or everyday situation is a separate task. Removing complexities from the algorithm allows students to focus on extracting information from the word problem.

The students appeared to enjoy writing iterative programs to compute products. When asked about his experience, Ian said,

> I like that activity...It gave me some challenges, some math problems...and I liked it because it was fun.

Many of the other students shared similar views.

> *Faith*: I really liked the programming the most.
>
> *Matias:* Coding is kind of fun.

Some students specifically indicated that the activities helped them learn multiplication. Matias said, "I did learn some times [tables]. That was helpful." When asked what was most helpful he said, "The [programming] blocks." Emily shared the same perspective: "I learned more about times tables." From both students' perspectives the programming activities helped them understand the principles behind the times tables.

For some of the students, however, the activities were not always seen as helpful. Derek initially said," It's fun and it kinda helps [with math],"but later explained that there were both "hard" and "easy" parts and that he did not think it improved his ability to do the math. Interestingly, his test scores did improve despite his perception.

Throughout the tests and activities, Derek struggled to identify when to multiply and when to add. However, he experienced several technical difficulties during the intervention making for a poorer experience.

4.3 Performance on Test Items

Figure 4.1 shows that improvement from pre to post-test was mixed between questions from the CCSSM standards. Our sample size was too small to do statistical tests, especially

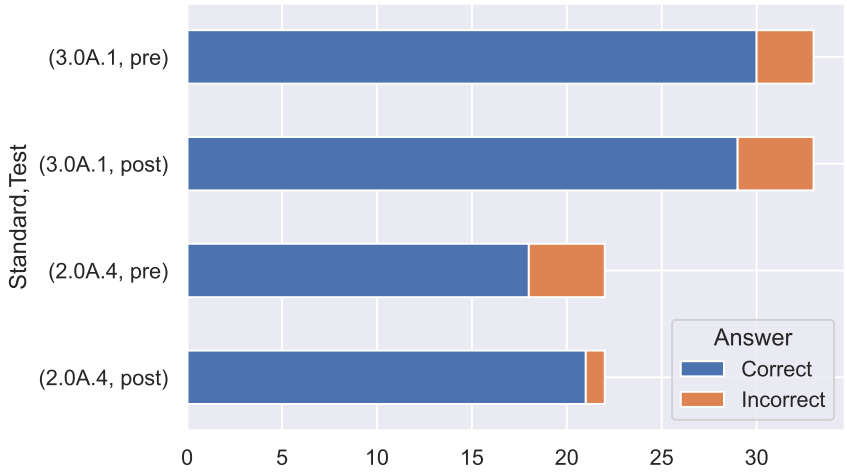

Fig. 4.1: As shown, some student scores improved (4), some stayed the same (4), and some performed worse (3), from pre to post-test.

with the only very modest improvements in performance from the pre-test to the post-test (from 48 correct to 50 correct).

We analyzed the reasons students got test questions wrong. From the 11 incorrect answers (both intermediate answers that were changed and final submitted answers) we discovered five reasons for answering incorrectly:

1. Student performed an iterative process incorrectly.
2. Student confused multiplication and addition.
3. Student used answer field as work space.
4. Student formulated the problem incorrectly.
5. Unknown: It was unclear why the student answered the question incorrectly.

Initially, it appeared that the reasons for answering incorrectly during the pretest were different from the reasons for answering incorrectly during the post test. After testing the codes for interrater reliability the codes were adjusted for clarity and then the sections were coded again by another researcher. This final round of coding helped determine that there were too many unknowns to draw any conclusions from the 11 occurrences. More data from the students would be required to draw conclusions.

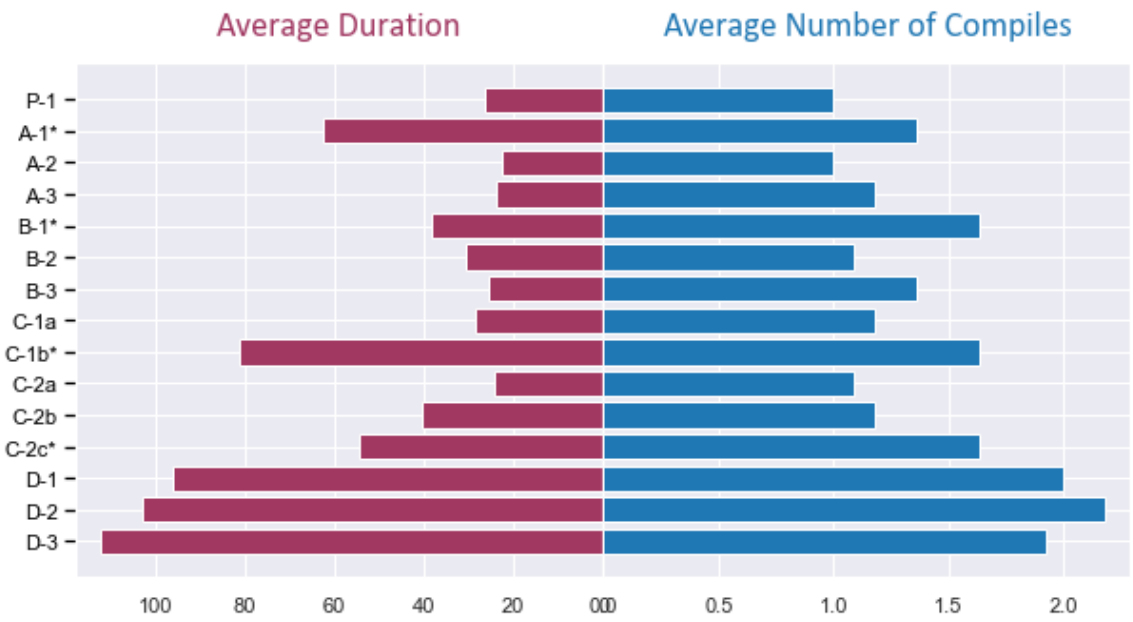

Fig. 4.2: Duration increased when new concepts were introduced. See Section 4.4 for activity descriptions. * indicates an 'increment' activity.

4.4 Use of Visual Programming

Visual programming has been used in both classroom and research settings. In both of these settings, the purpose is sometimes for learning to code and other times for coding to learn. In our study, all of the students were able to learn how to code with the visual blocks and use them to solve problems. We observed that activity duration increased for all students when a new programming concept or block was introduced, or when some form of scaffolding was removed. As noted in Figure 4.2, these "increment" activities saw increases in the number of times the students ran their code before successfully completing the activity. But subsequent activities with the same expectations were completed faster and with fewer runs.

In this study, we also observed instances when students were using coding to learn multiplication. In the following excerpt, we observe that scaffolding provided by the code blocks along with their shape helped one student deduce that her first attempt to answer the problem was incorrect. She realized that she needed to solve the problem in smaller steps.

> *Emily*: I think they [the students in the story problem] have 21 [pillows]. Pretty sure... Wait ... oh, I had to do six pillows. And then I had to do it four times.

Visual programming is not just entering numbers into a calculator. The intervention's activities required the students to understand the prompt, assemble the blocks correctly, provide the appropriate inputs, and then observe the code execution. Each piece was built to reflect the real process of the operation being taught.

Of the seven students interviewed, four of them had previous experience with coding and programming, three said that they had not. Considering that all of them were able to describe the function of the blocks and assemble them correctly despite varying levels of exposure indicates they can code using visual programming.

4.5 Use of Visualization

In addition to using blocks to program, visual programming allows for teaching to be supplemented with rich visualizations that demonstrate specific principles. In this study, a visualization was used to demonstrate the multiplication process through animations by adding the items to a grid one by one. Referencing the right-hand panel on the activity page as seen in Figure 3.2, several parents highlighted the use of visualization to demonstrate the multiplication process:

> The icons are soccer balls. So, it is not just five it is five soccer balls. That is really helpful for them to solidify [the multiplication process].

> It looks like something that would help them learn, help them visualize more. I think they need as much visualization as possible.

However, there was no strong indication that the students used the visualization of the multiplication matrix. In future work, a deeper analysis of different areas of visualization would be of interest. This could answer questions such as, "In what ways does animation timing affect learning outcomes?" and, "Which visualizations are more helpful: the programming blocks, or the execution result of the programming blocks?"

4.6 Threats to Validity

The participants in this study were drawn from students attending a university laboratory school. Parents of students in these settings are often more engaged in their child's learning and this may not accurately represent the general population. Parents were encouraged to only intervene when their child showed increased levels of stress. For those not observed there is no evidence that this was followed. In the case of one student, there were many technical difficulties with the application itself. This may have proved to be stressful to the student. This can also explain the increased duration and parental involvement of this specific student's activities.

CHAPTER 5

Conclusion

The purpose of our study was to explore the ways students' use of iteration influences their learning of multiplication with the support of visual programming activities. We found that the students successfully used visual programming to solve word problems that required multiplication. Students used iteration and the visual programming block structure to learn and strengthen their understanding of multiplication. Generally, they found the visual programming activities both helpful and enjoyable.

Visual programming can be a tool for developing a deeper understanding of multiplication by focusing on the underlying principle of iteration. It can also provide guided learning activities with clear learning objectives.

5.1 Future Work

There are many opportunities for future work. This study was performed in a single session with each student with students from a single location. It can be adapted and expanded to observe the effects of visual programming activities on learning outcomes over multiple sessions and across multiple locations.

Although performance on test items questions varied, the reasons for answering questions incorrectly in the pre-test and post-test were different. Further study and deeper analysis of test questions and program activities could provide better insight as to why the pre and post-test results showed improvement in certain areas and not in others.

Additionally, a deeper analysis on different areas of visualization and its effect on the learning process would provide information on developing visual programming languages and execution to aid in the learning process.

REFERENCES

[1] "Common core state standards," 2010. [Online]. Available: http://www.corestandards.org

[2] T. R. Shapiro, "Cursive handwriting is disappearing from public schools," Apr 2013. [Online]. Available: https://www.washingtonpost.com/local/education/cursive-handwriting-disappearing-from-public-schools/2013/04/04/215862e0-7d23-11e2-a044-676856536b40_story.html

[3] "Common core state standards for mathematics," 2010. [Online]. Available: http://www.corestandards.org/Math/

[4] B. Sherin, "A comparison of programming languages and algebraic notation as expressive languages for physics," *International Journal of Computers for Mathematical Learning*, vol. 6, pp. 1–61, 2001.

[5] S. Y. Lye and J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for k-12," *Computers in Human Behavior*, vol. 41, pp. 51–61, 2014.

[6] S. Papert, *Mindstorms: Children, Computers, And Powerful Ideas.* Basic Books, 2020. [Online]. Available: https://books.google.com/books?id=nDjRDwAAQBAJ

[7] P. S. Moyer-Packenham, C. W. Lommatsch, K. Litster, J. Ashby, E. K. Bullock, A. L. Roxburgh, J. F. Shumway, E. Speed, B. Covington, C. Hartmann, J. Clarke-Midura, J. Skaria, A. Westensckow, B. MacDonald, J. Symanzik, and K. Jordan, "How design features in digital math games support learning and mathematics connections," *Computers in Human Behavior*, vol. 91, pp. 316–332, 2019.

[8] [Online]. Available: http://nlvm.usu.edu/en/nav/vlibrary.html

[9] "About." [Online]. Available: https://scratch.mit.edu/about

[10] S. Papadakis, M. Kalogiannakis, and N. Zaranis, "Developing fundamental programming concepts and computational thinking with scratchjr in preschool education: a case study," *Mobile Learning and organisation*, vol. 10, no. 3, pp. 187–202, 2016.

[11] "About." [Online]. Available: https://www.scratchjr.org/about/info

[12] H. Palmér, "Programming in preschool - with a focus on learning mathematics," *International Research in Early Childhood Education*, vol. 8, no. 1, p. 75–87, 2017. [Online]. Available: https://eric.ed.gov/?id=EJ1173690

[13] G.Fessakis, E. Gouli, and E. Mavroudi, "Problem solving by 5-6 years old kindergarten children in a computer programming environment: a case study," *Computers and Education*, vol. 63, pp. 87–97, 2013.

[14] J.-M. Sáez-López, M. Román-González, and E. Vázquez-Cano, "Visual programming languages integrated across the curriculum in elementary school: A two year case study using 'scratch' in five schools," *Computers and Education*, vol. 97, pp. 129–141, 2016.

[15] A. Sullivan and M. U. Bers, "Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade," *International Journal of Technology and Design Education*, vol. 26, pp. 3–20, 2016.

[16] M. U. Bers, C. González-González, and M. B. Armas–Torres, "Coding as a playground: Promoting positive learning experiences in childhood classrooms," *Computers & Education*, vol. 138, pp. 130–145, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131519300995

[17] J. F. Shumway, J. Clarke-Midura, V. R. Lee, M. M. Hamilton, and C. Baczuk, "Coding toys in kindergarten," *Teaching Children Mathematics TCM*, vol. 25, no. 5, pp. 314 – 317, 2019. [Online]. Available: https://pubs.nctm.org/view/journals/tcm/25/5/article-p314.xml

[18] M. Hamilton, J. Clarke-Midura, J. F. Shumway, and V. R. Lee, "An emerging technology report on computational toys in early childhood," *Technology, Knowledge and Learning*, vol. 25, no. 1, p. 213–224, 2019.

[19] "Blockly." [Online]. Available: https://developers.google.com/blockly

APPENDICES

APPENDIX A

Test Questions

Table A.1: Test Questions

Question ID **(Related Question)**	**Test** ***Standard***	**Question** ***(Answer)***
eli-books (sue-sleeps)	Pre-test *3.0A.1*	Eli reads 7 books each week for 4 weeks. How many books did he read altogether? *(28)*
jones-desks (teacher-rug)	Post-test *2.0A.4*	Mrs. Lee has 3 rows of desks in her classroom. Each row has 5 desks. How many desks are there in all? *(15)*
pedro-coins (sara-coins)	Pre-test *3.0A.1*	Pedro, Allie, and Fred have 5 coins each. How many coins do they have in all? *(15)*
sara-coins (pedro-coins)	Post-test *3.0A.1*	Sara, Julia, Greg, and Mario have 4 coins each. How many coins do they have in all? *(16)*
sue-sleeps (eli-books)	Post-test *3.0A.1*	Sue sleeps 8 hours each night. How many hours does she sleep in 3 nights? *(24)*
teacher-rug (jones-desks)	Pre-test *2.0A.4*	Our teacher's rug has 5 rows of squares with 3 squares in each row. How many squares are on the rug? *(15)*
tray-cookies (zach-stickers)	Post-test *2.0A.4*	Kate placed 4 rows of cookies on a tray. There were 2 cookies in each row. How many cookies are on the tray? *(8)*

* indicates a control group

Continued on next page

Table A.1 – continued from previous page

Question ID **(Related Question)**	**Test** ***Standard***	**Question** ***(Answer)***
zach-stickers (tray-cookies)	Pre-test *2.0A.4*	Zach puts his stickers in 4 rows. There are 2 stickers in each row. How many stickers does Zach have? *(8)*
3x4 (4x3)	Pre-test *3.0A.1*	3 x 4 = *(12)*
4x3 (3x4)	Post-test *3.0A.1*	4 x 3 = *(12)*
round-66 (round-47)	Pre-test* *3.MD.A.1*	Round to the nearest ten. 66 *(70)*
round-47 (round-66)	Post-test* *3.MD.A.1*	Round to the nearest ten. 47 *(50)*
round-75 (round-53)	Pre-test* *3.MD.A.1*	Round to the nearest ten. 75 *(80)*
round-82 (round-43)	Post-test* *3.MD.A.1*	Round to the nearest ten. 82 *(80)*
round-43 (round-82)	Pre-test* *3.MD.A.1*	Round to the nearest ten. 43 *(40)*
round-53 (round-75)	Post-test* *3.MD.A.1*	Round to the nearest ten. 53 *(50)*

APPENDIX B

Activity Outlines

Activity P-1

Description: This activity starts with a block already on the canvas. The student will click the Run Button in order to compile the code.

Prompt: The blocks on the white page are going to tell the computer what to do. Click the run button below to see what happens

Activity A-1

Description: This activity starts with a block already on the canvas. The student will change the value and then click run.

Prompt: The blocks on the white page are going to tell the computer what to do.Change the 7 on the block to a 4.

Activity A-2

Description: This activity starts with a block already on the canvas. The student will change the value and then click run.

Prompt: Tell the computer to draw 8 books. Change the 5 on the block to an 8.

Activity A-3

Description: This activity starts with a block already on the canvas. The student will change the value and then click run.

Prompt: Tell the computer to draw 7 pieces of candy.

Activity B-1

Description: This activity asks students to drag the needed block onto the canvas. The student will change the value and then click run.

Prompt: Tell the computer to draw 6 cookies. Also drag the block onto the white page.

Activity B-2

Description: This activity asks students to drag the needed repeat item block onto the canvas. The student will change the value and then click run.

Prompt: Tell the computer to draw 5 pillows. Drag the block onto the white page.

Activity B-3

Description: This activity asks students to drag the needed block onto the canvas. The student will change the value and then click run.

Prompt: Tell the computer to draw 5 stars.

Activity C-1a

Description: This is activity C-1 part 1 of 2. It requires the student to drag the needed repeat item row block onto the canvas and change its value.

Prompt: Tell the computer to draw 5 apples.

Activity C-1b

Description: This is activity C-1 part 2 of 2. With the repeat item block on the canvas already, it requires the student to drag the needed repeat row block onto the canvas.

Prompt: Tell the computer to draw 3 rows of 5 apples. To do this drag the repeat block onto the white page. Place the apple block inside of the repeat block.

Activity C-2a

Description: This is activity C-2 part 1 of 3. It asks the student to drag the repeat row block onto the canvas and change its input to the appropriate value.

Prompt: Tell the computer to draw 4 trees.

Activity C-2b

Description: This is activity C-2 part 2 of 3. Starting with the repeat item block on the canvas, it asks the student to drag the repeat row block onto the canvas.

Prompt: Tell the computer to draw 3 rows of 4 trees.To do this drag the repeat block onto the white page. Place the tree block inside of the repeat block.

Activity C-2c

Description: This is activity C-2 part 3 of 3. Starting with all of the blocks on the canvas, it asks the student to modify the value of the repeat row block.

Prompt: Tell the computer to draw 4 rows of 4 trees.

Activity D-1

Description: This is a comprehensive activity that requires the student to draw on all previous knowledge.

Prompt: Jack has 2 pea pods. Each pea pod has 6 peas. Tell the computer how to draw the problem.

Activity D-2

Description: This is a comprehensive activity that requires the student to draw on all previous knowledge

Prompt: Josh cut the cake at a party. He made 4 rows. Each row had 5 pieces of cake. Tell the computer how to draw the problem.

Activity D-3

Description: This is a comprehensive activity that requires the student to draw on all previous knowledge.

Prompt: Laura, Troy, David, and Jill each have 6 pillows. How many pillows do they have in all? Tell the computer how to draw the problem.