Programming as a teaching tool in third-grade mathematics

Eric Bagley

July 2020

Introduction

Transitions move us from where we are to what comes next. They can be seamless or more difficult depending on the situation. Elementary school is full of those transitions. Some students maneuver through these transitions easily while other students find them much more difficult. In the latter situation teachers often need to be creative in order to find ways to help facilitate these transitions.

Learning the basics of mathematics plays a key role in developing logical and abstract thinking skills and problem-solving skills. For some students, mathematical concepts are simple; they just make sense. For others, these concepts don't come easily at all. These preconceptions often originate from a number of reasons but that is a conversation outside the focus in this paper. The fact is, understanding the complexities of mathematics is difficult for many students to understand.

One of the most challenging parts of education is determining what to teach and how to teach it. Some recent attempts at education reform include the No child left behind Act, the Every Student Succeeds Act and the Common Core Standards. Many education reforms, or at least parts of them, introduce positive and promising results while others perform less than optimal. Regardless of what system is in place, there is a need for teaching tools that help students develop skills that will help them in all aspects of life. Creativity, problem-solving, critical thinking, and communication are vital for someone to become well rounded, responsible individuals that contribute to their home-life, work, and communities.

The focus of the proposed study is not to propose a system that outlines education reform nor to argue that there is a need for it. Its focus is much more

specific: third-grade mathematics. Third-grade is a common time for students to begin learning about multiplication and division, fractions, structures of arrays and area, and describing and analyzing two-dimensional shapes, place value, patterns, and problem-solving, number operations, and measurement and data analysis. The key principles behind learning these concepts are to make sense of problems and attempt to solve them, reason abstractly and quantitatively, construct viable arguments and critique the reasoning of others, model with mathematics, and to look for patterns.

By age 8, children start valuing relationships more, think about the future, and become more independent from parents and family. They can more accurately describe their feelings and understand more about their place in the world. They are beginning to see and understand things from another person's point of view. Their ability to solve problems increases and their attention span gets longer. In math at school, they should be pretty comfortable with addition and subtraction, and they are becoming more proficient with multiplication.

Technology is increasingly becoming a more integral part of our daily lives. One might argue that teaching multiplication and division are no longer necessary because technology can perform those operations for us. Although the latter statement may be true students benefit from skills developed while learning those mathematical operations. Teaching these topics, or at least the principles behind them, are ever more important because technology often does much of the thinking for us. Integrating technology and computational thinking will be extremely valuable to students in their current and future everyday life.

One of the major transitions from second-grade to third-grade mathematics is moving from array-based thinking, using addition, subtracted, repeated addition, and repeated subtraction, to a unit-based, more abstract, thinking, using multiplication and division. This later leads to fractions and ratios. This transition is often difficult for students and a non-traditional approach may prove beneficial.

Another major goal of mathematics at this age is to help the students understand the fundamental pieces of numerical operations and how they are related to one another. Although quick answer algorithms exist, without a strong understanding of the functions behind them it is difficult for students to truly understand the concept. If this is the case, students often just plug in the numbers and hope that they turn out. This may provide correct solutions, but it doesn't lend itself well to solving problems that haven't already been solved. This is detrimental to subsequent learning because every step builds on the previous. If this happens, later on, teachers find that they need to fill in the gaps. This could introduce remedial coursework that may not be needed by other students and may

cause disinterest or lost opportunities for deeper learning.

Understanding the aforementioned misconceptions and shortcomings offer us key insights that can help us develop different teaching tools that help students fully understand fundamental concepts. We propose using programming as a method for teaching multiplication and division at a third-grade level as it could provide an appropriate transition from the group based thinking students have developed from second-grade to the unit based, more abstract thinking they need to develop for future problem-solving skills. It has the potential to assist students in understanding fundamental concepts and fill learning gaps before they become detrimental to both the individual student's learning and the learning of their classmates. It offers an appropriately tailored transition using programming to help them solidify the basics and move to solve more advanced topics.

This study will have a control group and a test group involving two parts, quantitative analysis, and qualitative analysis. The qualitative portion is built off a pre-test and post-test. The qualitative analysis will consist of various pieces: recordings of thinking out loud, pre- and post-interviews, and observation of classroom instruction and participation.

1 Literature Review

Using programming as a teaching method for topics outside of computer science is not novel. There have been many studies related to pedagogy and programming across all educational levels. Within each level, a wide range of topics are covered. Computer Science principles are used, however, to facilitate the learning process. These principles include computational thinking, problem-solving, planning, and creativity. The aforementioned principles have been used to teach language, writing, mathematics, physics, and other science topics.

The studies reviewed can be placed into three categories: ones that teach computer science principles, ones that teach programming languages, and those that use computer science principles and programming languages to teach subjects outside of computer science.

The first area addressed in a number of studies have research objectives that focus on determining appropriate ways to teach a specific computer science principle. Other studies focus on determining appropriate ways to teach programming syntax such as Phanon [EDWARDS]. Based on the target audience there is a variety of teaching software and a variety of different languages that aim to teach different principles. For example, block-based visual programming targets

younger or more novel students, mixed block-text based programming languages such as Droplet can be used for older or familiar students, and text based languages are more easily understood by secondary and post-secondary students. Each of these have varying levels of scaffolding. Block programs heavily utilize scaffolding by providing hints and guiding programmers develop code using close to natural language while text based languages can have little to no scaffolding.

Some programs aim to provide dynamic scaffolding that offers more intensive supports initially but gradually removes some or all scaffolding support for more advanced users.

Learning through Code

The second area addressed in a number of studies is using software programs or programming languages as tools to teach educational subjects outside of computer science. Different than virtual manipulatives or other computer games, programming has proved to be beneficial in helping students develop problem-solving, planning, meta-cognition and other skills [1, 2].

2 Research Question

How does visual programming affect the academic outcomes of multiplication and division for third grade students?

Hypothesis

Visual programming can increase understanding of multiplication and division in third grade mathematics. Success in visual programming activities are good indicators that students understand concepts of division and multiplication. Visual programming can provide more appropriate transitions from one mathematical concept to a more advanced mathematical concept.

3 Methods

Due to the small sample population size and the scope of the study, we don't expect the study to be statistically significant. However, we do expect that it

can provide insight for future studies and contribute to integrating computational thinking into the classroom through mathematics.

The study will be carried out during the USU Spring 2021 semester.

Organization

The study will be divided up into programming activities. Each programming activity will have specific learning objectives that focus on learning a specific education standard. Each activity will also highlight different programming concepts.

These programming activities can be viewed as homework assignments accomplished either in the classroom or at home. One or two assignments will be given each week, or every day that a new multiplication or division topic is taught.

Gathering Data

The study will be mainly quantitative but it may also involve some qualitative data. It will involve a control group and a test group, preferably taught by the same teacher. The purpose of this study is for academic and practical purposes. It contributes to the well studied topic of using programming as a teaching tool and it provides more specific application to elementary school.

Quantitative Data

Qualitative data will be gathered at various points during the study. Before the study students will participate in a pre-test. The purpose of the pre-test is to evaluate where each student stands and if they are meeting the current standards.

Before each activity in the study a short questionnaire will be given.

During the activities we can gather other various forms of data. These may include: the total time it takes for the student to complete the activity, the total number of blocks that they use to complete the activity, the total number of blocks they drag onto the canvas but don't use, the number of times they run the compiler and observe the program running, their final code block solution, and whether or not they successfully completed the activity among other things. This data can be used both for observing performance of a single activity and activities over time. If students take much longer than expected it may mean that there are flaws in the programming environment.

After each activity students will complete a second questionnaire for them to reflect on what they learned and to see how they felt about the activity. Coupled with pre-activity evaluations we may see trends.

After the study is complete a post-test will be given to determine if students met the learning objectives.

Qualitative Data

Qualitative data may include written feedback from the teacher, parents, or the student. This can give us general insight on the study overall. Where the hypothesis accurate? Does the quantitative data match up with the qualitative data? If so, in what ways? If not, where was the discrepancy?

Participants

Students, Teachers, Researchers, Student's Parents (possibly) The students will participate in the programming activities. They may carry them out individually, as a class, or in small groups. They will provide feedback about the programming environment, learning activity, and their general experience.

Teachers will introduce the course material and the learning activities. They may also introduce the programming environment. They help observe the students and their progress. They may provide feedback about the study, the programming environment, the learning activities, and their general experience. The teacher works with the researchers to develop the programming environment and learning activities for the students.

Researchers may provide feedback about the study. With advising from teachers, they prepare the programming environment and learning activities. They also analyze the data and report findings.

Programming Environment

Math Topics

4 Implications and potential outcomes

Students will learn math. Computational thinking in real-world situations

5 Conclusion

References

[1] Pamela Fernstrom Doublas Fuchs, Lynn S. Fuchs. A conservative approach to special education reform: Mainstreaming through transenvironmental programming and curriculum-based measurement. *American Education Research Journal*, 30(1):149–177, 1993.

[2] Nicholas Zaranis Stamatios Papadakis, Michail Kalogiannakis. Developing fundamental programming concepts and computational thinking with scratchjr in preschool education: a case study. *Mobile Learning and organisation*, 10(3):187–202, 2016.