STUDENTS' COMPUTATIONAL THINKING IN TWO MATHEMATICS BLOCK-BASED PROGRAMMING ENVIRONMENTS: RESEARCH DURING COVID-19

J. Enrique Hernández-Zavaleta[1], Sandra Becker[1], Douglas Clark[1], Corey Brady[2], and Natalie Major[3]
1. *Werklund School of Education - University of Calgary*
2. *Peabody College - Vanderbilt University*
3. *Canadian Rockies Public Schools*

ABSTRACT

This paper analyzes the computational practices that four 7th and 8th grade students engaged in when learning geometric transformations in two different online block-based programming environments**.** The data sources include video footage of students' interviews in Zoom where they shared their screens and cameras. The findings determined that students utilized in particular, decomposition and pattern recognition as important computational thinking practices required for learning in STEM disciplines. The paper also describes the changes made in how research method, data collection, and analysis configured opportunities to study computational thinking in remote locations due to the restrictions brought on by COVID-19**.** We identified three main challenges in the transition to online research: (a) recruiting research participants which included instituting necessary revisions to ethics protocols; (b) rethinking data gathering and analysis techniques along with interactions with participants in virtual settings; (c) dealing with glitches associated with technologies and virtual communication media in just-in-time ways. We conclude that even given the challenges with researching during COVID-19, there are still opportunities for rich, robust research in online settings.

Keywords: *computational thinking, geometric transformations, block-based programming*

INTRODUCTION

Several scholars have suggested that integrating computational thinking can productively transform STEM education (Sengupta et al., 2018; Wilensky, Brady, & Horn, 2014). Mathematics is a subject that shows many disciplinary overlaps with computer sciences (Hoyles and Noss, 2020; Weintrop, et al., 2015; Wing, 2008). In particular, programming as a practice has a geometric engagement tradition since Logo's environment appeared in the 1980's. Such programming environments allow for the study of geometric shapes and its transformations in order to create geometrical meaning as students program the path of a turtle (Edwards, 2009; Papert, 1980).

Although there is a long history of research that has explored the development of students' understandings about geometric transformations through programming, there has been less work to date studying the development of students' computational practices as they learn about geometric transformations. The primary aim of this presentation is to share the computational practices demonstrated by 7th and 8th grade students in two computational environments, a video game and Scratch, in the context of learning about geometric transformations. A secondary aim will be to describe how COVID-19 restrictions led to a pivot in regards to the research method, data collection and analysis, and preparation of manuscripts while still configuring opportunities to study computational thinking in remote locations.

The global pandemic has shuttered public life in many ways, but STEM educators have not halted their duties as "homeschooling" has become the "new normal." These settings bring in new challenges for educators and researchers as they face uncertain environments that demand adaptative capacity to cover the students' learning necessities (Manning, 2020). In this respect, STEM education research can provide suggested strategies and activities that allow students to develop skills in physically distanced scenarios with in some cases, a lack of accessible communication and access (Bakker & Wagner, 2020). This paper focuses on addressing these issues by describing some computational practices grade seven and eight students performed when interacting within a conceptually integrated videogame and a Scratch activity adapted for learning in a virtual setting.

COMPUTATIONAL THINKING

Extending on Wing's (2008) elucidation, computational thinking has been broadly defined by The National Research Council [NRC](2011), as a skill that "everyone, not just computer scientists, can use to help solve problems, design systems, and understand human behavior. [As such,] computational thinking is comparable. . . to the mathematical, linguistic, and logical reasoning. . . taught to all children" (p. 3). According to Sengupta et al. (2018), the core of computational thinking is found in "abstractions," that "are generalized computational representations that can be used (i.e., applied) in multiple situations or contexts" (p. 355). The notion of computational abstraction in "use" (Sengupta et al., 2018) is understood as a practice that considers the concepts (e.g., loops and conditionals) and other practices (e.g., solving-problem, debugging, pattern recognition) of the computer's science. For instance, programming underlies the notion of an abstraction of a process that executes a series of steps and provides an output (solution) to the desired problem (Hoyles & Noss, 2020; Wing, 2008).

Several investigations have characterized concepts (abstractions) and practices (abstractions in use) fundamental to computational thinking development (Brennan & Resnick, 2012; Gadanidis, et al., 2017; Weintrop et al., 2015). For this study we drew on computational practices as suggested by Hoyles and Noss (2020) as follows:

- **Decomposition** involves solving a problem by solving a set of smaller problems (Weintrop et al., 2015; Sinclair & Patterson, 2018).
- **Algorithmic thinking** is the propensity to see tasks in terms of smaller connected steps (Hoyles & Noss, 2020).
- **Abstraction** involves seeing a problem at different levels of details as well as the ways in which expressions within a situation can point beyond the boundaries of that situation. In other words, is a process from the experience to the concept (Hoyles & Noss, 1996).
- **Pattern Recognition** is "seeing a new problem as related to problems previously encountered" (Hoyles & Noss, 2020).
- **Generalization** involves the transition from seeing specific cases only as such to seeing specific cases as generic examples (Hoyles & Noss, 1996).

From our analysis, we noted that the students, though engaging to a greater or lesser extent in all computational practices, resorted most often to (a) decomposition of the sequences or series of individual steps or instructions that can be executed by the computer, and (b) pattern recognition in the form of the loop, a technique utilizing the iterated repetition of a set of instructions over and over again (Brennan & Resnick, 2012; Sinclair & Patterson, 2018), promoting computational efficiency (where the code runs the shortest possible script to achieve the most robust action).

Our study also recognizes the social aspects involved in computational thinking. Sengupta et al. (2018) warn about the fallacy of *technocentrism*, i.e., questions about technology which reference the technology itself, leaving aside the individuals who interact with it. Sengupta et al. state that commonly the learning objectives and the evaluation of computational thinking focus on the

production and improvement of understandings about computational abstractions, instead of focusing on the role of discourse, corporeal reasoning or aesthetic experiences of people, as phenomenological aspects of computational thinking. Adding a social vision of computing and mathematics in research highlights people's productions and their collective experiences. In this way, the uses of the abstractions will differ from individual to individual, depending on their context and their reason for use. Therefore, we acknowledge that the development of computational and STEM thinking together not only relies on conceptual intersections but also on practices and phenomenological aspects.

METHODOLOGY

This study focused on four student cases learning about transformations in two different computational environments, a video game and Scratch. Qualitative comparative case study was selected as methodology because it allowed for inquiry that was exploratory, explanatory, pragmatic, and phenomenological (Harrison et al, 2017). Given COVID-19, we had to re-envision this research in an online as opposed to classroom setting. Revisions to the research plan were submitted to the university ethics board (CHREB) and approved prior to engaging in the online work with students. Due to limited response to participation requests, authors 1, 2, and 4 worked with four students individually, each in four separate sessions, two in the game and two in Scratch. Each student brought a unique background to the research study as indicated in Table 1.

Table 1. Students' block-based programming and mathematical experience

Student	Grade	Block-based programming experience	Mathematics profile
Zach	7	Extensive experience creating games in Scratch	Previous year's teacher indicates average performance in mathematics
Simon	8	No block-based programming experience	None provided.
Paul	8	No block-based programming experience	Parent indicates strong mathematical capability but often underachieves
Eric	8	No block-based programming experience, but observed friends	Parent indicates struggle and lack of confidence in mathematics

The design of Transformation Quest, led by Author 3, draws on conceptual integration and disciplinary integration as indicated by Clark, Sengupta, et al. (2015) where mathematics and computational thinking concepts are integrated directly into the mechanics as a central focus for reward and achievement, rather than being embedded as an activity that appears after completion of other goals in the game.

In playing the game the students use programming blocks with transformations directives to position a red right triangle in a Cartesian plane 20x20 sized; The objectives are related to the collection of magical yellow or blue gems strategically disposed into the cartesian plane, avoiding static enemies that block gems' positions and minimizing the number of transformations blocks (e.g. using the loop block).

The Scratch activity, Code the Quilts, developed by Author 1, was inspired by the work of Lehrer et al., (1998) and features the exploration of code sequences used to develop multiple quilt patterns. In the first session, the students played with the existing code to determine how it led to emergent quilt patterns. In the second session, the students were encouraged to create their own code, or use existing code to design a new quilt pattern.

Due to the constraints related to Covid-19, the video footage of student participation was recorded in a virtual setting (Zoom), where the participants shared their screen with the researchers during the game play and Scratch activity. Researchers one and two engaged in ongoing dialogue while the students at this time. Guiding questions helped understand students' predictions (e.g., What is your plan? Based on your code, can you tell us where the triangle will map?) and explanations (e.g., Can you explain what happened?) It should be noted that when working with student Zach, our first participant, we were joined by his former teacher (Author 4), who interacted with him as well.

Data analysis of the video footage took place using Nvivo 12 in virtual zoom sessions. Assigned codes, developed prior to analysis, were utilized to determine student understanding of computational and mathematical concepts, as well as student implementation of computational and mathematical practices as evidenced by their discourse, embodied expressions, and coding sequences in both environments. In addition, researchers observed and noted student comments related to the game and Scratch experience.

FINDINGS

All four students show evidence of how their prior experiences influenced their understandings of geometric transformations and computational thinking abstractions, depending on their individual context and their reason for use. We provide three examples of student computational and mathematical thinking while engaging in game play and the Scratch activity in an online environment.

Example 1: Pattern recognition in the service of efficiency

In Scratch (International Scratch Wiki Community, 2020), efficiency is linked to more content in larger projects within a smaller file size. Following this idea, one problem that makes programming sequences inefficient is the use of multiple similar scripts (patterns) that can be reduced into one instruction allowing the program to run faster. In this respect, Zach uses his previous experiences with Scratch programming:

> ***A4:*** *Can you just explain more about that loop? Why do you want to repeat it four times? [Figure 1 right dashed rectangle]*
>
> ***Zach:*** *This, this makes it look more efficient. Because if you did this, you'd have to do something like this [Figure 1 right]. This is just incredibly inefficient. So that's probably why you have it there.*

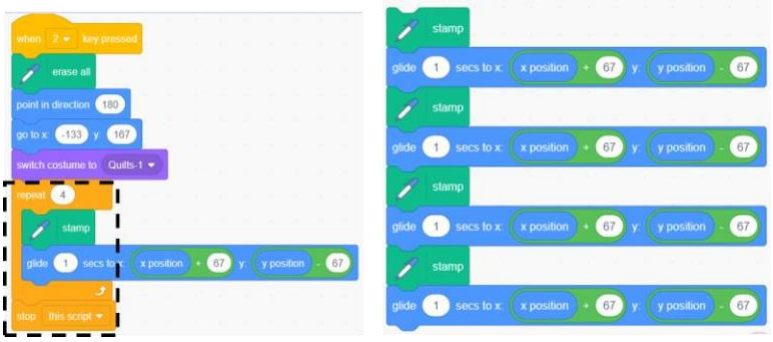

Figure 1. Zach's examples of efficient and inefficient code.

Zach's left code sequence uses a loop. To explain his understanding of efficiency, Zach shows us a counterexample to make his point (Figure 1, right). This counterexample shows a repetition pattern of two instructions that can be synthetized with a repetition block. Zach uses the loop abstraction as an instantiation for efficiency in relation to pattern recognition in computational thinking practice.

Example 2: Decomposition in Transformation Quest

Paul explores Transformation Quest by screen testing block by block until he solves the problem. His decomposition practice, using only individual translations, rather than employing reflections in a pattern is useful for him with no prior programming experience. He explains:

> ***Paul:*** *I will go across to these three [right blue gems], and then down to both these ones*

Later he realizes that to finish the level, he needs to apply at least one reflection across the y-axis, in order to invert the triangle for a successful exit.

> ***Paul:*** *that are reflected on the y-axis to collect this one [blue gem at the left button]. Then we have this one and then down to the bottom, and then across to here.*

Based on his limited experience, Paul does not discern the potential repeating pattern inherent in using the reflection block, therefore his strategy is to decompose single translation moves in the game, using inversion as a reflection property only when necessary. That said, Paul constructs an efficient (in terms of number of blocks used) and pragmatic (in terms of goals achieved) inscription.

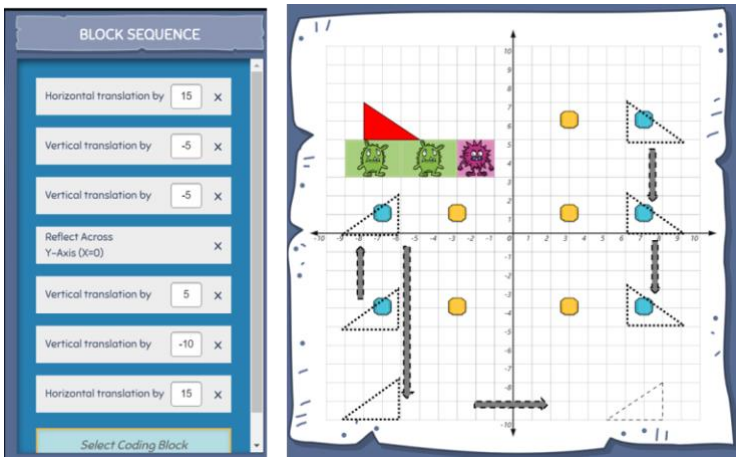

Figure 2. Paul's level 4 solution.

Example 3: Decomposition in Code the Quilts

When asked to create his own quilt, Eric engaged in an aesthetic experience (Sengupta & Farris, 2016) by modifying the original colors of the quilt and creating a tank shape. In order to achieve his goal, Eric indicates that simplicity not efficiency, was crucial. Eric built a sequence reusing the simplest code available (number one) utilizing only the "glide t secs to x y" Scratch block (Figure 3, left).

> ***Eric:*** *Okay, yeah. Well, since I'm gonna be trying to keep this fairly simple. I think I might base it off number one.*

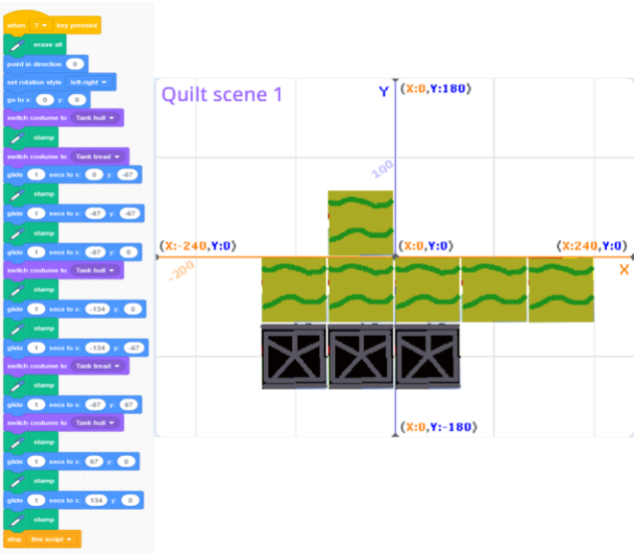

Figure 3. Student Eric sequence and final tank design

Eric modified this code by using decomposition, testing block by block. Although he articulates an emphasis on simplicity, his explanation shows he understands the basic functions of the blocks.

> ***Eric:*** *So, point in direction 90 that basically tells the block which way to point. Orient itself. Go to x and y tells the block where to go. And then switch costumes to quote one gives it the appearance and stamp closet to stay there.*

In this example, Eric was engaged in a personalized programming activity, where his primary focus was on the construction of the tank shape. By necessity, he oriented the computational practices he performed to his overall goal. Due to his limited experience in coding, the decomposition practice was the most affordable way to achieving his goal.

DISCUSSION, COVID-19, AND CONCLUSIONS

In this study, decomposition appears as an intersectional practice of computational and mathematical thinking. As stated by scholars (Edwards & Zazkis, 1993; Francis & Davis, 2018), not only is it a powerful problem-solving technique that can be adopted by novice programmers, it is an experiential-based learning practice that can be used in addition to pattern recognition practice. For Zach, his experience with pattern recognition and the loop block in the service of efficiency allowed him to show his transformation geometry understandings using loop abstractions. For Paul and Eric, their goals in achieving levels in the game and completion of an aesthetic quilt design led them to utilize decomposition to achieve those ends. The pattern recognition and decomposition practices exemplified in this study were crucial in the students' multidisciplinary approach to the development of thinking required for STEM disciplines. In this respect, the activities in the game and in Scratch, even though they took place in remote locations, allowed all students to express phenomenological aspects of computational thinking.

The challenges created by the COVID-19 pandemic, though real, also helped create new opportunities for us as researchers who turned to online settings as sources of interaction. In particular, we identified and faced three main challenges: (a) recruiting research participants which included instituting necessary revisions to ethics protocols; (b) rethinking data gathering and analysis techniques along with interactions with participants in virtual settings; (c) dealing with glitches associated with technologies and virtual communication media in just-in-time ways.

This meant we had to be flexible and open to contending with complications as they arose. By initiating revised ethics permissions from the university ethics review board, we were able to contact participants through known teacher associates and parents. Given the limited number of volunteers, however, we had to shift our online sessions to only one student participant at a time. In addition, we

had to ensure students were comfortable in participating with their webcam on, sharing their screen, and sharing their thinking aloud which allowed researchers to observe and record embodied (e.g., body motion and the mouse position on the screen) and verbal data. There were technical complications that arose during the virtual interviews (e.g., missing data from zoom recordings or difficulties linking to the online game) which presented implications for thorough data analysis.

Even given the challenges faced in conducting online research however, we were able to engage in rich learning experiences for both the students and ourselves, as evidenced by Simon and Eric, who indicated they would like to continue working with us. In this sense, social aspects inherent in these computational thinking activities may enhance the interactions between participants, researchers, and teachers in online settings.

We contend that even given the challenges with conducting research during the time of COVID19, there are still opportunities for rich, robust study in online settings. Though we would have preferred to conduct research in student pairs with a broader range of participants, we did find that students drew on their own experiential background to enact computational practices that assisted them in achieving the goals they set within a game and Scratch environment.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1742257. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Bakker, A., & Wagner, D. (2020). Pademic: lessons for today and tomorrow? *Educational Studies in Mathematics, 104*, 1-4. https://doi.org/10.1007/s10649-020-09946-3

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *American Educational Research Association Meeting*, (pp. 1-25). Vancouver, BC.

Clark, D., Sengupta, P., & Brady, C. (2015). Disciplinary integration of digital games for science learning. *IJ STEM Ed, 2*(2). https://doi.org/10.1186/s40594-014-0014-4

Edwards, L. (2009). Transformation geometry from an embodied perpective. In W.-M. Roth (Ed.), *Mathematical Representation At The Interface of Body and Culture* (pp. 27-44). Charlotte, North Carolina, USA: Information Age Publishing Inc.

Edwards, L., & Zazkis, R. (1993). Transformation geometry: Naive ideas and formal embodiments. *Journal of Computers in Mathematics and Science Teaching, 12*(2), 121-145.

Francis, K., & Davis, B. (2018). Coding robots as a source of instantiations for arithmetic. *Digital Experiences in Mathematics Education*. https://doi.org/10.1007/s40751-018-0042-7

Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education*, 458-477.

Harrison, H., Birks, M., Franklin, R., & Mills, J. (2017, January). Case study research: Foundations and methodological orientations. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research, 18*(1).

Hoyles, C., & Noss, R. (2020, June 19). *Online seminar series on programming in mathematics education.* (C. Bateau, & G. Gadanidis, Eds.) Retrieved November 19, 2020, from Mathematics Knowledge Network: http://mkn-rcm.ca/online-seminar-series-on-programming-in-mathematics-education/

International Scratch Wiki Community. (2020, November 11). *Scratch wiki*. Retrieved from Efficient programming: https://en.scratch-wiki.info/wiki/Efficient_Programming

Lehrer, R., Jenkins, M., & Osana, H. (1998). Longitudinal study of children's reasoning about space and geometry. In R. Lehrer, & D. Chazan (Eds.), *Designing learning environments for developing understanding of geometry and space* (pp. 137-167). Mahwah, NJ: Lawrence Erlbaum Associates.

Manning, A. (2020, June 15). *How teachers are adapting to COVID-19 disruptions is subject of new study*. Retrieved from Phys.org: https://phys.org/news/2020-06-teachers-covid-disruptionssubject.html?utm_source=TrendMD&utm_medium=cpc&utm_campaign=Phys.org_TrendMD_1

National Research Council. (2011). *Report of a Workshop on the Pedagogical Aspects of Computational Thinking.* Washington, DC: The National Academies Press. https://doi.org/10.17226/13170

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: learning cultures and computers.* London: Kluwer academic publishers.

Papert, S. (1980). *Mindstorms, children, computers, and powerful ideas.* USA: Basic Books, Inc.

Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in STEM education. In M. Khine (Ed.), *Computational Thinking in the STEM Disciplines* (pp. 49-72). Springer, Cham. https://doi.org/10.1007/978-3-319-93566-9_4

Sinclair, N., & Patterson, M. (2018). The dynamic geometrisation of computer programming. *Mathematical Thinking and Learning*, 54-74. https://doi.org/10.1080/10986065.2018.1403541

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wilensky, U., Corey, B., & Horn, M. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM, 57*(8), 24-28.

Wing, J. (2008). Computational thinking and thinking about computing. *Phil. Trans. R. Soc. A*, 3717-3725. https://doi.org/10.1098/rsta.2008.0118