Research monograph # 69
April 2017

Computer Coding in the K–8 Mathematics Curriculum?

By George Gadanidis, Western University; with Iain Brodie, Leslee Minniti, and Bronna Silver, St. Andrews Public School, Toronto District School Board

The trend of adding some form of computer coding to curriculum is an international phenomenon. How exactly should computer coding fit in the curriculum? Should it be its own subject? Should it be integrated with other subjects?

Computer coding is creating a buzz in education. Prime Minister Justin Trudeau recently said, "We need to do a lot better job of getting young people to understand what coding is and how it's important, how to program, how to problem solve, how to create the most elegant algorithm possible."[1] BC recently announced that computer coding will be added to all grades of the K–12 curriculum, and Nova Scotia has made a similar announcement. The trend of adding some form of computer coding to curriculum is an international phenomenon; in 2014, England mandated a coding curriculum for all K–12 students.

Where in the Curriculum?

Currently, computer coding is conceived of more as its own curriculum area than as a way of thinking that may enhance existing subject areas. Jeannette Wing proposes, "To reading, writing, and arithmetic, we should add [computational thinking] to every child's analytical ability"[2] (p. 33).

This focus on computer coding in education is not new. It was an important component of Seymour Papert's work with Logo, a programming environment that invites children to write code to move a turtle on the screen. Papert saw Logo both as a coding environment and as a mathematics learning environment. He wrote that Logo "is to learning mathematics what living in France is to learning French"[3] (p. 6).

Benefits of Coding

At the heart of computational thinking – and mathematics – is abstraction. When children write code, they come to…

1. understand in a tangible way the abstractions that lie at the heart of mathematics,
2. dynamically model mathematics concepts and relationships,
3. gain confidence in their own ability and agency as mathematics learners.

Other researchers emphasize that integrating coding with other subjects, especially mathematics, creates pedagogical opportunities. However, a recent review of the state of computational thinking in K–12 concludes that "underinvestigated is the idea of computing as a medium for teaching other subjects"[4] (p. 42).

Implications for Mathematics Education

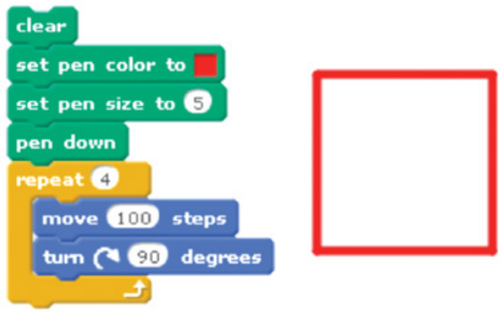
Below we describe three important ways the use of coding in mathematics teaching and learning can enhance student conceptual development.

1. Abstraction Made Tangible

At the heart of computational thinking – and mathematics – is abstraction.[5,6] We use abstraction naturally from a very young age as we develop language. For example, when we come to understand *cat*, we look past all the differences among cats and create an abstract model of essential cat characteristics.

Let's see how abstraction comes into play when we use the blocks-based coding environment Scratch (see scratch.mit.edu) to draw a square. Scratch was developed in 2004 at MIT and its design is based on Logo and the work of Papert.[7]

We could code a square as shown in Figure 1. (The code for our Scratch examples is available at scratch.mit.edu/projects/115404418/#editor for you to use, edit, experiment with, and share with others.) We could also accomplish this task by defining *draw square* as its own code block and using it when needed (see Figure 2). Creating a new code block to replace a group of code blocks is especially useful when we write more complex code.



```
clear
set pen color to 🟥
set pen size to 5
pen down
repeat 4
    move 100 steps
    turn ↻ 90 degrees
```

Figure 1. Draw a square with Scratch.

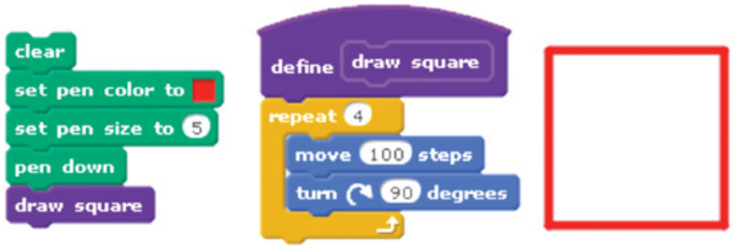
The *draw square* code block helps simplify our code. It also makes it easier to use *draw square* as an object of other code. For example, the code in Figure 3 draws a square, turns 36 degrees, and repeats this 10 times; by randomly changing the pen colour before each square is drawn, the individual squares become more easily visible.

What is happening here mathematically? First, the concept of *square* has been abstracted to its essential elements: move 100 steps, turn 90°, repeat 4 times. Second, this abstraction is conceptually robust as it represents all squares, in the same way that the word *cat* represents all cats. We can edit the code to draw squares of different sizes, orientations, and colours. Third, this abstract version of a square is tangible.[8] It has been turned into a code block that can be moved, manipulated, and acted upon by other code blocks.

Isn't it interesting that with computer coding we can abstract mathematical concepts and at the same time make them feel tangible? Such tangible abstractions help students with conceptual development.

This tangible quality can be enhanced when we use code to "teach" programmable robots to perform mathematics tasks. For example, children can write similar code to instruct Sphero to walk a square. In simplest terms, Sphero is a robot on wheels trapped inside a spherical shell. It moves in the same way as a hamster ball (see sphero.com) and can be coded using a smart phone or tablet application.

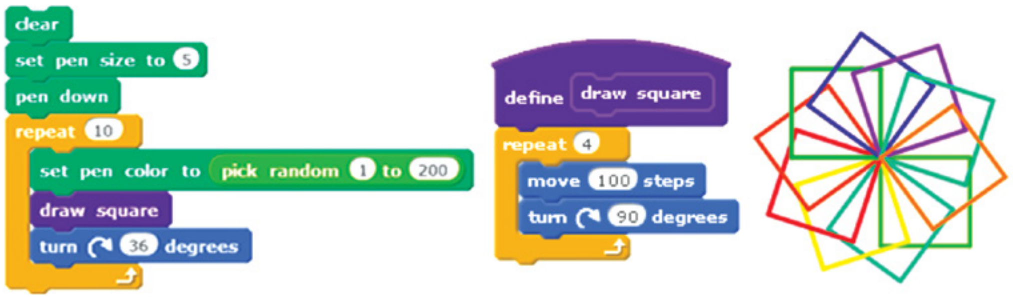
Such experiences help address Ontario curriculum expectations in both Geometry and Spatial Sense ("describe, sort, classify, build, and compare two-dimensional shapes"



```
clear
set pen color to 🟥
set pen size to 5
pen down
draw square

define draw square
repeat 4
    move 100 steps
    turn ↻ 90 degrees
```

Figure 2. Adding a *draw square* code block



```
clear
set pen size to 5
pen down
repeat 10
    set pen color to pick random 1 to 200
    draw square
    turn ↻ 36 degrees

define draw square
repeat 4
    move 100 steps
    turn ↻ 90 degrees
```

Figure 3. Drawing a pattern with rotated squares.

in kindergarten; "identify and describe common two-dimensional shapes" in Grade 1; and "identify and describe various polygons" in Grade 2) and Patterning and Algebra ("using the core of a pattern and predicting what comes next" in kindergarten; "identify, describe, extend, and create repeating patterns" in Grades 1–3).[9,10]

2. Automation and Dynamic Modelling

Jeannette Wing states, "Computing is the automation of our abstractions"[5] (p. 3718). For example, the Scratch code block *draw square* automates the drawing of a square. We create the code once, and we can use it to draw as many squares as we like. Abstraction, together with automation, offers students the opportunity to model mathematics concepts dynamically.

Dynamic modelling allows students to investigate relationships, pose and test what-if questions, and easily share their findings and knowledge with peers, as well as family and friends.

3. Low Floor, High Ceiling[3] and Student Agency

The coding environments available today, especially those that are blocks-based, offer a *low floor*, allowing engagement with minimal prerequisite knowledge, and a *high ceiling*, offering opportunities to extend learning to more complex concepts and more varied representations, just like Papert's Logo.

A low-floor and a high-ceiling environment supports differentiated learning. Students can engage at their ability and comfort levels and investigate extensions as they develop conceptual understanding and gain confidence.

Coding in a low-floor and a high-ceiling environment also supports student agency and gives students ownership of their learning. Students writing code to model a pattern or a relationship are in control. There are many different ways to solve a problem with code and students can use methods that personally make sense. They can also deviate from the task to investigate related problems.

Feedback From Teachers

A few years ago, the three co-authors of this monograph (Iain Brodie, Leslee Minniti, and Bronna Silver) tried coding in their math teaching for the first time. Their comments tell the story of what they did and what they experienced:

- "Grades 7–8 students learned to draw a square in Scratch, so they could then teach the Grades 1–3 students. They also challenged themselves to draw other shapes. Probably the most important word for students has been *curiosity*."
- "Grades 1–3 students were challenged to instruct the teacher on how to 'walk a square' (with the teacher purposely making mistakes when their instructions were not clearly articulated)."
- "In the Grade 1 room, it was great to see the older students with the younger children, their little hands just hovering above the keyboard and sometimes they'd put them down, and the older students were so beautiful at saying, 'No you use your hands, you're going to do it'."
- "It was amazing. I could see children saying, 'What happens if ...?' A lot of times it was my Grades 2/3 students teaching the older kids things."
- "I never heard any of my students say, 'How do you spell this?' That anxiety wasn't there. This program does not force them to do that. It really was a very celebratory experience."
- "What I really loved was the connection we made with the families at home. We said to the students, this is now your project and you need to share this. They were skeptical that their parents could do it, but they came back excited to share what they did with them."

In Sum: Give It a Try!

A number of resources are available to integrate computer coding into the classroom. Over the last few years, we have developed material to support teachers as they use computational thinking in mathematics teaching and learning. Below is an annotated list.

- **The Computational Thinking Community of Practice of the Ontario Math Network** (mathnetwork.ca/ct) offers documentaries of lesson studies from Ontario classrooms and other resources. Funded by the Ontario Ministry of Education (A KNAER project hosted by the Fields Institute for Research in Mathematical Sciences).
- **The computational thinking module** (researchideas.ca/wmt/c6.html) offers background on computational thinking in mathematics education, along with classroom examples, games, and simulations. The module is also available as a free certificate course through Western University and the Fields Institute (see researchideas.ca/wmt/courses.html).
- **Math + Code 'Zine** (researchideas.ca/mc) is published quarterly and supports coding in mathematics education contexts. It is supported by Western University's Teaching Support Centre and the Fields Institute.
- **Computational thinking in mathematics education research** (ctmath.ca), a partnership of seven universities and the Fields Centre for Mathematics Education, researches the use of computational thinking from kindergarten to undergraduate mathematics. Projects, reports, documentaries, and lesson plans will be posted as they are developed.

Don't be apprehensive about integrating computer coding in your teaching. You and your students will love it! As one of the co-authors observed during the lesson, "I was so excited that everybody in the Grade 1 room was successful, viewing themselves capable, able, and mathematicians. Not only the students, but the teachers too."

References

1. *Kitchener Post* (2016). Google opens its doors on Breithaurt; PM Trudeau takes part. Retrieved from http://www.kitchenerpost.ca/news-story/6240341-google-opens-its-doors-on-breithaupt-pm-trudeau-takes-part/
2. Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
3. Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
4. Grover, S. and Pea, R. (2013. Computational thinking in K–12: A review of the state of the fields. *Educational Researcher, 42*(1), 38–43.
5. Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366*(1881), 3717–3725.
6. Aho, A.V. (2012) Computation and computational thinking. *Computer Journal*, 55, 832–835.
7. Logo History. Retrieved from http://el.media.mit.edu/logo-foundation/what_is_logo/history.html
8. Gadanidis, G. (2017 in press). Five affordances of computational thinking to support elementary mathematics education. *Journal of Computers in Mathematics and Science Teaching 36*(2), 143–151.
9. Ontario Ministry of Education (2005). *The Ontario Curriculum Grades 1-8: Mathematics*. Toronto, ON: Queen's Printer for Ontario.
10. Ontario Ministry of Education (2016). *The Kindergarten Program*. Toronto, ON: Queen's Printer for Ontario.

What Works? Research into Practice is produced by a partnership between the Ontario Association of Deans of Education and the Student Achievement Division.

To learn more about how to write a monograph **click here**
Mobilizing Research into Practice in Meaningful Ways
By Dr. Michelann Parr and Dr. Terry Campbell Co-editors

What Works? is updated and posted at: www.edu.gov.on.ca/eng/literacynumeracy/inspire/research/WhatWorks.html

The opinions and conclusions contained in these monographs are those of the authors and do not necessarily reflect the policies, views, or directions of the Ontario Ministry of Education or the Student Achievement Division.

ISSN 1913-1097 What Works? Research Into Practice (Print)
ISSN 1913-1100 What Works? Research Into Practice (Online)