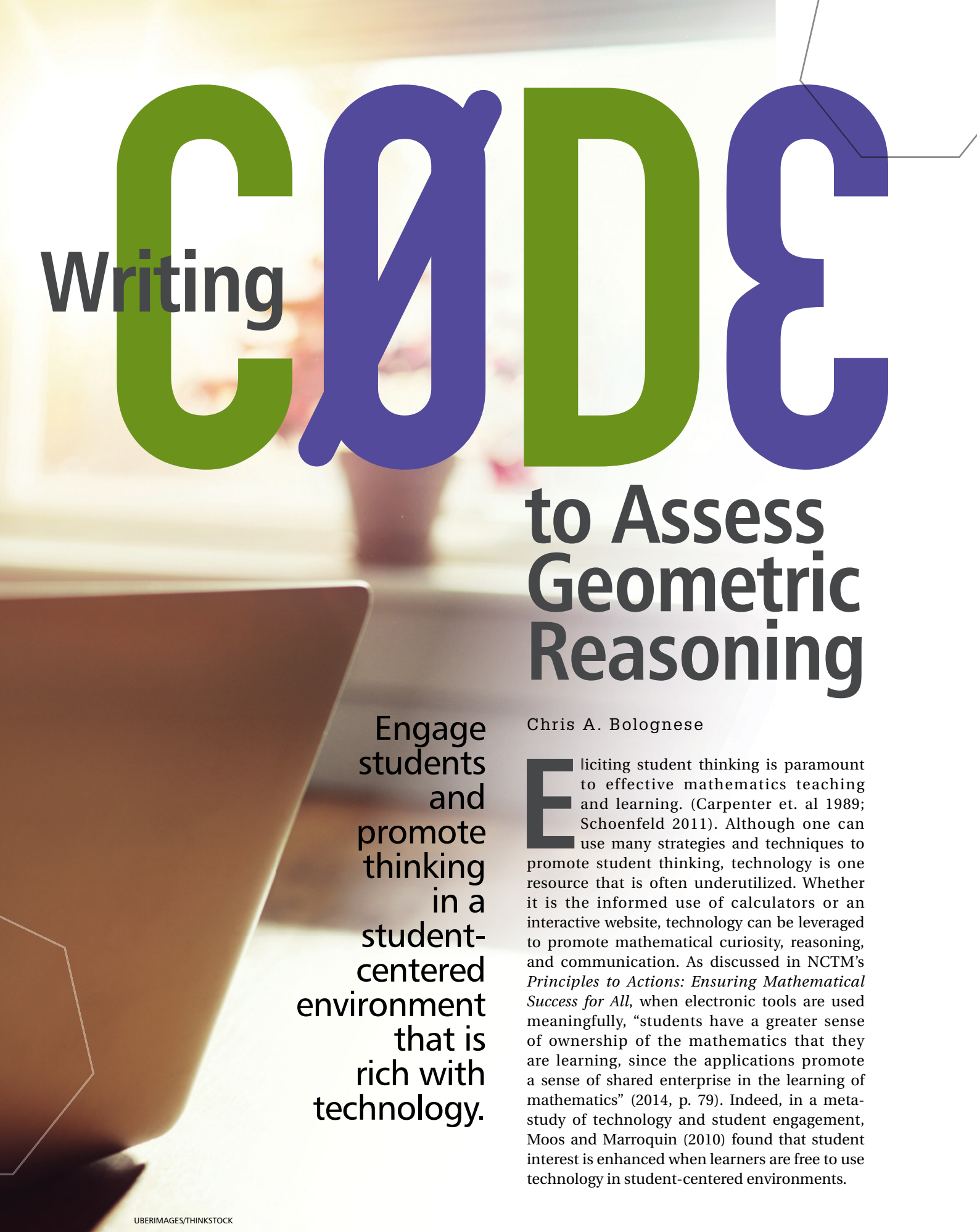


Writing CODE to Assess Geometric Reasoning

Engage students and promote thinking in a student-centered environment that is rich with technology.

Chris A. Bolognese

Eliciting student thinking is paramount to effective mathematics teaching and learning. (Carpenter et. al 1989; Schoenfeld 2011). Although one can use many strategies and techniques to promote student thinking, technology is one resource that is often underutilized. Whether it is the informed use of calculators or an interactive website, technology can be leveraged to promote mathematical curiosity, reasoning, and communication. As discussed in NCTM's *Principles to Actions: Ensuring Mathematical Success for All*, when electronic tools are used meaningfully, "students have a greater sense of ownership of the mathematics that they are learning, since the applications promote a sense of shared enterprise in the learning of mathematics" (2014, p. 79). Indeed, in a meta-study of technology and student engagement, Moos and Marroquin (2010) found that student interest is enhanced when learners are free to use technology in student-centered environments.

UBERIMAGES/THINKSTOCK

FIGURE 1

The *Scratch* platform shows code on the right and the animation on the left.

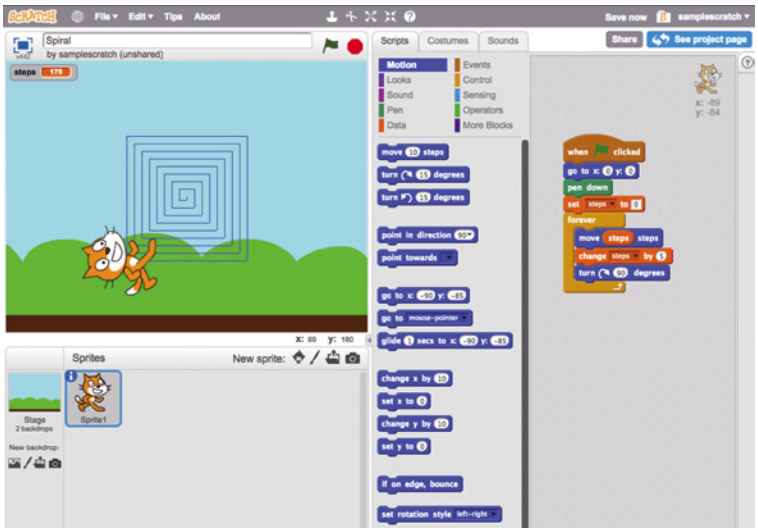
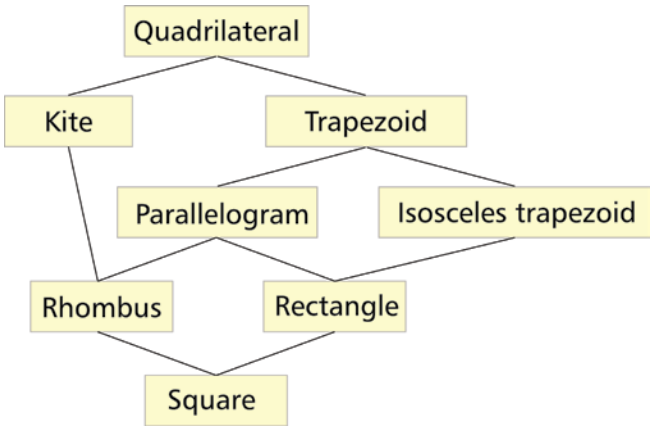


FIGURE 2

The quadrilateral hierarchy reflects the CCSSM learning standard of classifying two-dimensional shapes on the basis of their properties.



The code controls characters called *sprites* on the screen, which is, in a sense, like writing a movie script to direct actors in a play.

With these benefits in mind, this article explores how fifth-grade students' use of technology, specifically writing code using the platform *Scratch*, not only promoted curiosity and mathematical reasoning but also allowed for a more detailed way of assessing students' prior geometric understanding. Examples of student work appear along with suggestions of how anyone can learn to code with *Scratch*.

The coding movement

A recent development in education has been the so-called "learn-to-code" movement. With society's increasing use of (and dependence on) the digital world, some educators, tech gurus, and even politicians feel it is increasingly important to develop citizens who can think computationally. The site Code.org estimates that by 2020 we will have a surplus of more than one million technology jobs that cannot be filled by computer science majors. Even more disparagingly, they found that nine of ten K–grade 12 schools in the United States do not offer any computer science classes. President Obama even issued a challenge in his 2016 State of the Union address calling for U.S. high schools to redesign their offerings and partnerships in science, technology, engineering, and mathematics. If learning to code seems so vital, how can such experiences be meaningfully integrated into student learning experiences, and how can computer coding be connected to other disciplines, such as mathematics?

Coding using *Scratch*

Many platforms and computing languages are available to use in the classroom; a popular choice is the web-based programming platform *Scratch*, developed at MIT, which has existed since 2007. The name derives from the concept of scratching—that is, remixing or reusing someone else's project. In this sense, users can download someone else's source code and edit that code to create a new project. With more than 1500 new projects added each day by users ranging in age from 8 to 16, *Scratch* has grown to become the equivalent of *YouTube* for programming (Resnick et. al 1989).

Scratch is often called a "drag-and-drop," or visual, environment, which allows users to fit together programming commands in the shape of puzzle pieces. This type of coding

permits students to bypass common errors with syntax if they fail to type symbols correctly in the computer language. The code controls characters called *sprites* on the screen, which is, in a sense, like writing a movie script to direct actors in a play.

Figure 1 shows a sample *Scratch* program. The window on the right shows that code was written by dragging in commands from the central window. Common commands that can be used include instructions for changing how the sprite looks and moves, commands to repeat a block of code, and commands to check if a condition has been satisfied (e.g., if the sprite is on the edge of the screen). In the left window, students can see the result of their code once the code is run, often by hitting the green flag icon at the top.

This particular program shown in **figure 1** has the sprite draw a spiral, which is a good coding activity for beginners. When the green flag near the top center is clicked, the sprite will put down the pen and start at point (0, 0), which is the center of the screen. The sprite will then repeatedly move a certain number of steps and then turn 90 degrees. By incrementing the number of steps after each turn, a spiral is drawn on the screen.

Using *Scratch* to build a guessing game

This article showcases an activity designed to preassess fifth-grade students' understanding of polygon properties before they begin a polygon unit. All but two students had no experience with writing any computer code; however, almost all students had much experience with digital games, apps, and other software. This inherent predisposition to interacting with technology is one argument for infusing educational experiences with more coding (Resnick et. al 1989).

A learning standard of the Common Core State Standards for Mathematics (CCSSM) (CCSSI 2010) is for students to classify two-dimensional shapes on the basis of their properties. Since third grade, students have explored how two-dimensional shapes, such as triangles and quadrilaterals, can be related on the basis of their properties (see **fig. 2**). These students had multiple years of exposure to these shapes' properties and their relationships, but we anticipated that students would have difficulty in two specific areas:

FIGURE 3

Since third grade, these students had explored two-dimensional shapes on the basis of their properties and had been evaluated with traditional preassessment items like the one below.

Which words describe this shape?

a. rectangle
b. trapezoid
c. square
d. quadrilateral

1. creating a precise definition of a special type of polygon; and
2. distinguishing a shape's characteristics from its defining attributes.

Rather than administering a traditional preassessment that might consist of multiple-choice items (see **fig. 3**), an open-ended programming task in *Scratch* was designed to elicit student thinking and prior knowledge. The main objectives of this guessing game assessment were threefold:

1. Students would list statements as clues that collectively describe a particular mystery polygon of their choosing. In particular, students would have to identify which clue number, when taken in sequence, determined the specific shape that was being described. Thus, students had to use logic to understand that although a certain polygon was described by all of the clues, only a certain clue helped determine the specific type of mystery polygon described.
2. Students would use the motion commands in *Scratch* to accurately draw their polygon on the screen. This required spatial

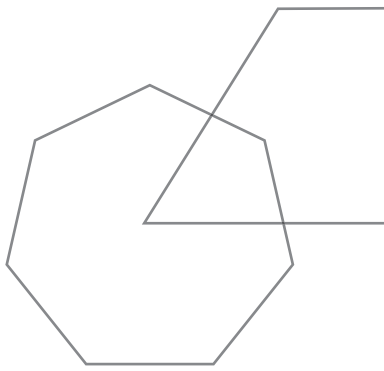
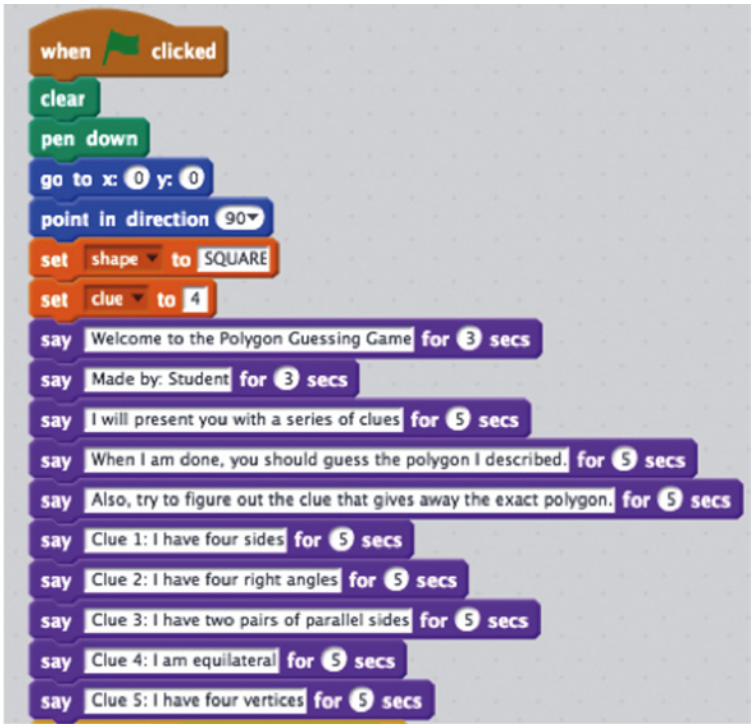


FIGURE 4

Sample code for a quadrilateral guessing game in *Scratch* includes the two variables *shape* and *clue* (shown in orange) that students could change to the desired mystery shape and concluding clue number.



reasoning, understanding of scale, and coordinate geometry.

3. Students would play one another's games and evaluate them for their mathematical accuracy while evaluating their own thinking. Providing opportunities for students to reflect on and express their understandings is paramount (Black and Wiliam 1998).

Because so many of the students were beginners with coding, they received a starter file that they could edit. This editable file can be accessed digitally at https://scratch.mit.edu/projects/93626527/#editor. The file served as an exemplar of a guessing game in which a series of clues are presented and the player of the game must determine by clue 4 that the mystery shape is a square. The clues used in this sample game are below.

1. I have four sides.
2. I have four right angles.
3. I have two pairs of parallel sides.
4. I am equilateral.
5. I have four vertices.

Students could use sample code to augment building their game (see **fig. 4**). In particular, this code uses two variables, *shape* and *clue*, (shown in orange) that can be changed to the desired mystery shape and concluding clue number. The purple commands would include directions that students wanted to give the player as well as the clues that geometrically described their mystery polygon. **Figure 5** displays two screen captures of the animation interface showing the sprite when the code was run using the green flag.

Students created guessing games for many different types of polygons; however, work from two particular students is closely analyzed here. This work showcases subtle similarities and differences in student misconceptions of polygon properties that become evident because of the open-ended nature of the task.

Let's examine two games for the same shape, a trapezoid, by students A and B (see **fig. 6** and **fig. 7**). What do these two games tell us about these students' prior understanding and reasoning? In particular, which student do you think has a better understanding of what it means to be a trapezoid? Further, how does this form of preassessment shed light on student understanding and misconceptions versus data obtained from a more traditional assessment?

The code written by student A showcases six clues; clue 4 is recorded as the clue determining that the shape must be a trapezoid. Student B's clues deduce that the shape must be a trapezoid as well, but by the third clue. Both students consider a trapezoid to be a shape that has *only one pair* of parallel sides, contrary to the more modern inclusive definition that describes a trapezoid as having *one or more* pairs of parallel sides (CCSSI 2010). Additionally, both students have the misconception that a trapezoid cannot possess right angles. Similarly, both students consider trapezoids to have a pair of congruent sides, most likely considering all trapezoids to be idealistically isosceles.

These two sets of clues might look similar, but two essential differences further detail student thinking. Student A's third clue states that the congruent sides are opposite one another; student B's fourth clue does not necessarily state that these congruent sides are opposing. The latter case could produce a nonisosceles

trapezoid with one base congruent to a neighboring side. These subtleties in precision are necessary for students to consider and were addressed more specifically in discussion that followed the preassessment. Language such as *exactly one* versus *at least one* and *opposite* versus *adjacent* were emphasized in subsequent instruction to allow students to be more precise in their descriptive language.

Recall that students were told that the sequence of the clues mattered and that a large part of the game was to determine which clue deductively described their mystery shape. By following student A's logic, once we know by clue 4 that the quadrilateral has four sides, then the shape must be a trapezoid. In fact, by clue 3, we know that the shape must be an isosceles trapezoid. This shows that the student either did not understand the logical requirements of the assignment or had misconceptions as to what requisite information determines a trapezoid.

Student B states that clue 3, "my shape has no right angles," determines that the shape must be a trapezoid. However, by this point, the shape could still be an *isosceles* trapezoid. This logical misconception seems different than that of student A previously. Although both students eventually describe an isosceles trapezoid with their clues, student A's designated clue is far more general than student B's. Such nuances in student thinking rise to the surface only because of the generative nature of the task.

Students received peer feedback on the accuracy of their clues and the precision of their logic. Because classmates played these games, the players of the game often critiqued the reasoning of the game designers, thus engaging in the third of the Common Core's Standards for Mathematical Practice (SMP 3): *Construct viable arguments and critique the reasoning of others*. This led to fruitful conversations for students to consider how they might refine their game to be more mathematically precise. When playing others' games, students filled in a table on which they wrote each clue, the remaining polygons that that could possibly fit this clue, and a brief description of their thinking. The teacher could use this table to provide feedback to a student on the basis of his or her reasoning about another student's game.

FIGURE 5

Two screen captures of the animation in *Scratch* show the interface when the code is run using the green flag.

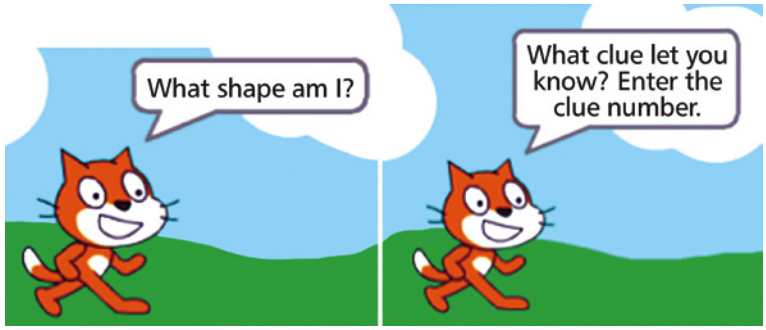


FIGURE 6

The code written by student A showcases six clues; clue 4 is the one determining that the shape must be a trapezoid.

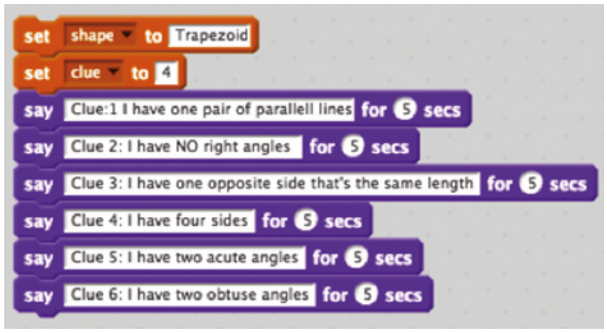


FIGURE 7

Student B and student A reached the same conclusion, but student B did so by the third clue, whereas student A took four clues.

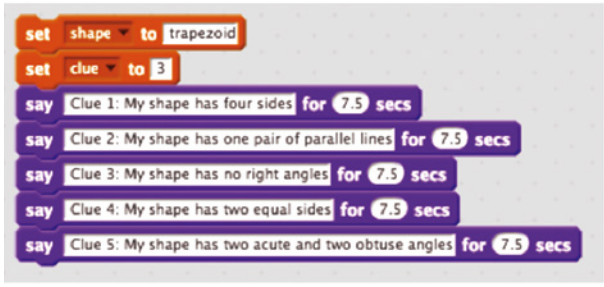


FIGURE 8

Giving step-by-step instructions for the sprite to draw a polygon challenged students, requiring them to reason abstractly and quantitatively. Receiving immediate feedback from running their code helped students make sense of the sprite's moves and turns.

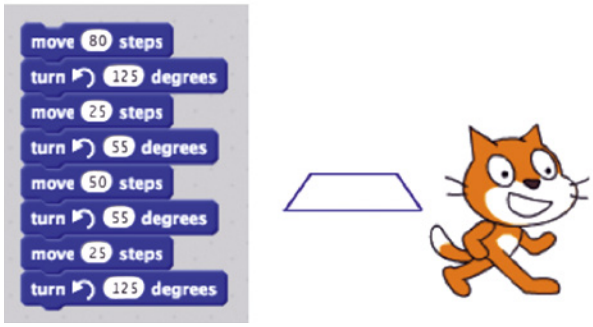
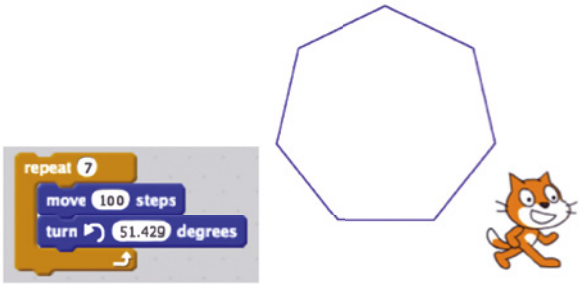


FIGURE 9

Using a loop, students could reduce the number of steps in their code. With repeated moves to draw a heptagon, some students discovered that the sum of exterior angles in any polygon is 360 degrees.



Analyzing student work: Drawing

The second objective was for students to use the motion platform of *Scratch* to guide their sprite character to draw their polygon (see **fig. 8**). Giving step-by-step instructions for the sprite to draw the polygon was challenging, as it required them to employ SMP 2: *Reason abstractly and quantitatively*. The difficulty many students had was to understand that a turn angle constructed the *exterior angle* of a polygon instead of the more familiar interior angle. However, receiving immediate feedback from the drawing that resulted from running their code allowed students to make sense of how the sprite moved and turned. Some students used SMP 8: *Look for and express regularity in repeated reasoning* to make their code to draw a shape more efficiently. Depending on the symmetry of the shape, students could use a loop to abbreviate the number of steps in their code. Repeatedly moving and turning seven times drew a heptagon (see **fig. 9**). One can easily see that the total angle turned is $7 \times (51.429) \approx 360$ degrees. With this insight, some students self-discovered that the sum of exterior angles in *any* polygon is 360 degrees.

Easy to learn, fun to build

This activity demonstrates how technology, specifically coding, can be used as a form of assessment. Students were genuinely engaged in designing and playing one another's games, far more than during a traditional paper-and-pencil test. The activity illuminated student thinking and reasoning about polygons—the correct application of some concepts as well as common misconceptions, assumptions, or imprecisions. Even something as subtle as giving students autonomy to select their own shape sheds light on their understanding and confidence. In fact, of all the many polygons (besides squares) that students could select, only a few students selected polygons of six sides or more; many selected a rhombus, or as some stated, a *diamond*. Surprisingly, no student made a game about a kite, perhaps because its properties are most exclusive.

Now that you have seen a sample activity using *Scratch*, try making your own *Scratch* project. First view the "Getting Started with Scratch" tutorial at https://wiki.scratch.mit.edu/wiki/Getting_Started_with_Scratch or search for existing projects in their search bar. For younger students, you might consider using *Scratch Jr.* at http://www.scratchjr.org/teach.html. In designing future assessments, consider how you might use technology to complement or supplement an otherwise traditional mode and further enhance student motivation. *Scratch* and other similar platforms are easy to learn, fun to build, and have limitless uses with curricula!

Common Core Connections

4.G.1
5.G.1

REFERENCES

Black, Paul, and Dylan Wiliam. 1998. "Assessment and Classroom Learning." *Assessment in Education* 5 (1): 7–74.

Carpenter, Thomas P., Elizabeth Fennema, Penelope L. Peterson, Chi-Pang Chiang,

and Megan Loef. 1989. "Using Knowledge of Children's Mathematics Thinking in Classroom Teaching: An Experimental Study." *American Educational Research Journal* 26 (4): 499–531.
Common Core State Standards Initiative (CCSSI). 2010. Common Core State Standards for Mathematics (CCSSM). Washington, DC: National Governors Association Center for Best Practices and the Council of Chief State School Officers. http://www.corestandards.org/wp-content/uploads/Math_Standards.pdf
Moos, Daniel C., and Elizabeth Marroquin. 2010. "Multimedia, Hypermedia, and Hypertext: Motivation Considered and Reconsidered." *Computers in Human Behavior* 26 (3): 265–76.
National Council of Teachers of Mathematics (NCTM). 2014. *Principles to Actions: Ensuring Mathematical Success for All.* Reston, VA: NCTM.
Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, et al. 2009. "Scratch: Programming for All." *Communications of the ACM* 52 (11): 60–67.
Schoenfeld, Alan H. 2011. "Noticing Matters. A Lot. Now What?" In *Mathematics Teacher Noticing: Seeing through Teachers' Eyes*, edited by Miriam G. Sherin, Victoria R. Jacobs, and Randolph A. Philipp, pp. 223–38. New York: Routledge.
"What's Wrong with This Picture?" Code.org. Accessed August 1, 2015. http://code.org/promote/

Chris A. Bolognese, bolognesechris@gmail.com, is the upper school department chair for the Columbus Academy. He enjoys mathematics contests, mathematical technology, and math teacher circles.