



# Interdisciplinary Journal of Problem-Based Learning

Volume 12 | Issue 2

Article 1

Published online: 5-11-2018

## Tinkering with Logo in an Elementary Mathematics Methods Course

Keri Duncan Valentine  
West Virginia University, [kevalentine@mail.wvu.edu](mailto:kevalentine@mail.wvu.edu)

IJPBL is Published in Open Access Format through the Generous Support of the [Teaching Academy at Purdue University](#), the [School of Education at Indiana University](#), and the [Jeannine Rainbolt College of Education at the University of Oklahoma](#).

### Recommended Citation

Valentine, K. D. (2018). Tinkering with Logo in an Elementary Mathematics Methods Course. *Interdisciplinary Journal of Problem-Based Learning*, 12(2).  
Available at: <https://doi.org/10.7771/1541-5015.1754>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact [epubs@purdue.edu](mailto:epubs@purdue.edu) for additional information.

This is an Open Access journal. This means that it uses a funding model that does not charge readers or their institutions for access. Readers may freely read, download, copy, distribute, print, search, or link to the full texts of articles. This journal is covered under the [CC BY-NC-ND license](#).

# THE INTERDISCIPLINARY JOURNAL OF PROBLEM-BASED LEARNING

---

SPECIAL ISSUE: TINKERING IN TECHNOLOGY-RICH DESIGN CONTEXTS

## Tinkering with Logo in an Elementary Mathematics Methods Course

Keri Duncan Valentine (West Virginia University)

### Abstract

With an increased push to integrate coding and computational literacy in K–12 learning environments, teacher educators will need to consider ways they might support preservice teachers (PSTs). This paper details a tinkering approach used to engage PSTs in thinking computationally as they worked with geometric concepts they will be expected to teach in K–5. Experiences programming in Logo to construct authentic artifacts in the form of two-dimensional geometric graphics not only supported PSTs' understanding of core geometric and spatial concepts, but also helped them to make connections between mathematics and computational literacy. Artifacts and discourse are discussed as they relate to three core considerations: engaging learners to construct authentic artifacts, supporting a communitarian ethos, and supporting various types of rapid feedback.

*Keywords:* computational literacy, geometry, mathematics education, teacher education

---

### Introduction and Background

There is increasing attention focused on integrating computational literacy (Berland, 2016; diSessa, 2001) and computational thinking (Brennan, Balch, & Chung, 2014; Brennan & Resnick, 2012) in both formal and informal K–12 learning environments. Although there is disagreement concerning the spaces in which computational thinking (CT) might be best integrated into K–12 education—“as a general subject, a discipline-specific topic, or a multidisciplinary topic” (Grover & Pea, 2013, p. 40)—many argue for some form of integration in K–12 schools (Barr & Stephenson, 2011; Grover & Pea, 2013; Kafai & Burke, 2014; Weintrop et al., 2015; Wing, 2008). Weintrop and colleagues (2015) acknowledge that such an integration will be new for current teachers in the field, warning that “the inclusion of these practices, in and of itself, offers little guidance for teachers who will be required to realize them in their classrooms” (p. 128).

Yadav, Stephenson, and Hong (2017) argued for integrating CT in teacher education programs as part of their existing coursework—in educational technology courses and/or domain-specific methods courses (e.g., science education methods). Specifically, Yadav and colleagues expressed the need to support preservice teachers (PSTs) to think computationally, “as well as how to teach their students to think

computationally, especially in the context of specific subject areas” (p. 59). Responding to this need for integrating CT in teacher education and considering the goal of this special issue (Akcaoglu & Kale, 2017) to share “models and approaches to teaching and learning by tinkering in technology-rich design contexts,” this paper reports on an approach to engage elementary education majors in tinkering activity with Logo, an object-oriented program, as part of their mathematics methods course. The goal of this activity aimed to support PSTs in thinking computationally as they worked with geometric concepts they will be expected to teach in K–5.

### Context

As a mathematics teacher educator working with elementary education majors, I continually seek ways to engage PSTs to consider the role of computational literacy in their future mathematics teaching practice. Many of my PSTs had few, if any, experiences learning to design or program. They also entered the course with no prior mathematics methods experience, as this is the sole mathematics methods course for PSTs enrolled in the program. This context seemed like a viable space in which to support PSTs to connect mathematical and computational concepts and practices, both as learners and as future mathematics educators.

Toward the end of the semester, three class sessions were devoted to focusing on teaching and learning geometric and spatial reasoning concepts, drawing in part on the K–5 geometry content and practice standards elaborated in the Common Core State Standards for Mathematics (CCSS-M, 2010). Content standards state that learners should “analyze, compare, create, and compose shapes” (kindergarten), “reason with shapes and their attributes” (grades 1–3), “draw and identify lines and angles . . . classify shapes by properties of their lines and angles” (grade 4), and so on. While these concepts comprise the focus for content to be learned, the standards for mathematical practice orient mathematical activity by connecting what is to be learned to the ways in which learners should engage in mathematical activity (e.g., make sense of problems and persevere in solving them, construct viable arguments and critique the reasoning of others, model with mathematics, use appropriate tools strategically, and look for and make use of structure). While the course supports PSTs to understand these connections, they also need to consider the various ways their future students might progress in their thinking, possible difficulties that might arise during learning, and ways to support learners, such as drawing on prior experiences and maintaining cognitive demand.

### Purpose/Goal

In order to support PSTs to better understand the integrated mathematical and computational concepts and practices they will be expected to teach, I engaged them in a design and tinkering activity using Logo. Logo, an object-oriented programming language developed by Papert and colleagues, not only served as a viable tool for naturally integrating geometry and computation, it is also well aligned with one of the six National Council of Teachers of Mathematics’s (NCTM, 2014) guiding principles for school mathematics—“tools and technology as essential resources” for learning (p. 78). While NCTM acknowledges the ability of tools and technology to support general forms of participation (communication and collaboration), the focus prioritizes “what Dick and Hollebrands (2011) call ‘mathematical action technologies,’ which produce mathematical responses based on user input, allowing students to explore mathematical ideas and observe, make, and test conjectures about mathematical relationships” (NCTM, 2014, p. 79).

Working as a mathematics teacher educator for the past four years, I have become aware that my elementary education majors rarely consider the role of technology in their coursework. Like others in many teacher education and certification programs around the United States, PSTs might enroll in a single course related to technology integration. The problem with this model is that PSTs are less focused on the ways in which technologies change the mathematics their students can learn and alternative ways to engage in mathematical

practice. As Weintrop and colleagues (2015) acknowledge, there is a lack of guidance for in-service teachers to integrate CT into their teaching. This is also the case for PSTs.

During the last three class sessions of the semester (9 hours total), PSTs were asked to investigate mathematical action technologies for themselves, as learners, in the context of geometry and space. The goal of the learning activity described in this paper occurred during the second class session and primarily sought to engage PSTs in learning experiences integrating computational and geometric/spatial concepts through the activity of creating geometric designs (e.g., circles, tessellations). A review of literature related to computational literacy, CT, and object-oriented programming is elaborated before describing the tinkering approach used with PSTs.

## Computational Literacy and Computational Thinking

Computational thinking is a construct rooted in the notion of computational literacy. To operationalize the way both terms are being considered, this paper draws on the work of Berland (2016), diSessa (2001), and Brennan and Resnick (2012). Brennan and Resnick (2012) developed a framework for CT that was connected strongly to their experiences working with learners using the programming environment called Scratch (Lifelong Kindergarten Group, 2017). Their framework incorporates three dimensions of CT: computational concepts (e.g., loops, parallelism), computational practices (e.g., testing and debugging, experimenting and iterating), and computational perspectives. Berland’s (2016) and diSessa’s (2001) operationalization of computational literacy is a somewhat different way of framing computational activity wherein they distinguish between the “material, social, and cognitive” dimensions (Berland, 2016, p. 201). Material computational literacy refers to “making things with computation—primarily programming and electronics” (p. 202). Social computational literacy involves “the ability to articulate, discuss, communicate, and understand what you and other people need from those computational artifacts” (p. 202). Finally, cognitive computational literacy is “[l]inking the social and the material”—this is what Berland (2016) refers to as “computational thinking” (p. 202). He argues that making and tinkering offer a “powerful way to become computationally literate” across all three dimensions (p. 201).

### Object-Oriented Programming

One mathematical action technology that PSTs spent the most time exploring was Logo, an object-oriented programming language developed by Papert and colleagues and described in his book *Mindstorms: Children, Computers, and Powerful Ideas* (1993). Logo, also referred to as “turtle geometry,” is

described by Papert as a way in which “[c]hildren can *identify* with the turtle and are thus able to bring their knowledge about their bodies and how they move into the work of learning formal geometry” (p. 56). Creating graphics, for example, requires that learners teach the turtle a language of movement (e.g., forward 100 right 90). Geometry and spatial concepts are rooted in one’s position and movement in space, thus, these concepts can be explored by leveraging one’s embodied nature in the world. For Papert, turtle geometry provided this “bridge” between children’s everyday experiences and formal geometry.

Kafai and Burke (2014) draw heavily on Papert’s (1993) work to argue for the importance of connected coding. Rather than learning to program as a focus of learning, Kafai and Burke agree with Papert that Logo is in a sense a mathematical action technology (as well as an action technology for science and other domains)—it supports learners to reason in meaningful ways about mathematical relationships such as orientation, rotation, interior and exterior angles, and so on. Even more, Logo allows learners to tinker, experiment, conjecture, and see mistakes as a normal part of learning mathematics, providing “objects-to-think-and-share-with” (Kafai & Burke, 2014, p. 23).

## Design Considerations for Supporting Preservice Teachers’ Computational Tinkering Activity

Berland, Martin, Benton, Petrick Smith, and Davis (2013) define tinkering activity as that which “include[s] trial and error, messing around or fussing, finding and using feedback mechanisms (such as testing), or combinations of those activities” (p. 568). Although Berland (2016) doesn’t provide a formalized framework for designing and implementing a makerspace/tinkering learning environment focused on developing computational literacy, he does discuss in detail three core considerations informing my work with PSTs: engaging learners to construct authentic artifacts, supporting a communitarian ethos integrating reflective practice, and supporting various types of rapid feedback (material, social, and cognitive). In addition, the key features of makerspaces identified by Sheridan and colleagues (2014) supported considerations related to framing the process- and product-oriented practices involved in tinkering and making. Each of these considerations is elaborated below.

### Constructing Authentic Artifacts

Papert (1993) argued for a constructionist approach in which learners engage in learning by making and tinkering with objects (e.g., artwork, video games, graphics). According to Kafai and Burke (2014), “The educational theory of constructionism inherently values the making of objects as a core intellectual activity” (p. 23). Berland (2016) emphasizes that

constructionism “meshes well with the Maker Movement, as many makers engage in active construction of authentic artifacts for the benefit (or, at least, potential use by) other people” (p. 203). In a case study by Sheridan and colleagues (2014), several constructionist features of makerspaces were identified. First, tinkering spaces should be multidisciplinary in terms of the approach (process) and work produced (product). Especially important in the context of mathematics education is supporting PSTs to engage in learning that is atypical in mathematics classrooms—one that “break[s] down disciplinary boundaries in ways that facilitate process- and product-oriented practices” (p. 527). Second, these spaces “blend” both formal (e.g., demonstration, critique) and informal elements. Third, makerspaces focus “on learning as production rather than as mastery of a composite set of skills” (p. 526). While this is atypical of traditional mathematics classrooms, geometry is a viable domain in which to support PSTs to experience the potential benefits of production and also make connections to the CCSS-M.

In the second class session, PSTs were given the design objective to move the Logo turtle around on the screen, called a playground (a 2-dimensional boundary with a width of 700 units and a height of 500 units). As we were investigating geometry and spatial concepts, I expected PSTs to try to construct designs incorporating polygonal figures. I also prepared more formal activities to intersperse as PSTs seemed ready, usually manifesting with their asking process-related questions, making connections through their tinkering activity, or working on similar problems that could benefit from collaborative efforts (e.g., creating a more efficient program, determining a particular rotation angle needed). Table 1 (see next page) shows descriptions of planned activities, which include reviewing Logo language and computational terminology; learning to write programs; teaching Logo “words,” or programs; and introducing repeating and orienting commands.

### Communitarian Ethos

The second design consideration for supporting PSTs’ computational tinkering activity involved supporting a “communitarian ethos,” which Berland (2016) defines as the learning environment—the physical and social space where learning happens within a community. These include makerspaces, but can also be nurtured in more formal settings, such as a mathematics classroom. Berland (2016) characterizes this ethos as involving people “teaching each other to make things,” “identifying the wants and needs of the community,” discussing and reflecting wherein “learners notice connections together,” and developing “a *lens*—an understanding of how to read, evaluate, and engage with new situations using an understanding of complex content” (pp. 202–203, 208). Supporting this ethos involved attending to the role of the teacher,

Table 1. Planned activities to support coding in logo with description.

Planned Activities	Description
Reviewing Turtle Academy Logo language and computational terminology	<ul style="list-style-type: none"> <li>Defined command as an instruction that a computer can understand and execute</li> <li>Defined program as a sequence of commands (e.g., fd 100 rt 90 fd 100 rt 90)</li> <li>Reviewed basic commands: <ul style="list-style-type: none"> <li>To move the turtle, use the commands forward (fd) and back (bk) followed by a number (distance)</li> <li>To turn (orient) the turtle, use the commands right (rt) and left (lt) followed by a number (degrees)</li> <li>Clear screen (cs)</li> </ul> </li> <li>Identified Turtle Academy resource page for tinkering activity</li> </ul>
Writing programs to create polygons	<ul style="list-style-type: none"> <li>Challenged PSTs to write code for a set of images</li> <li>Learned repeat command to enable looping</li> </ul>
Teaching Logo a “word”	<ul style="list-style-type: none"> <li>Learned to write a program that Logo can reuse (e.g., To circle)</li> </ul>
Repeating and orienting	<ul style="list-style-type: none"> <li>Combined repeating and orienting commands to create multiple rotated figures</li> </ul>

the students, and the environment in ways articulated by Kurti, Kurti, and Fleming (2014). As the instructor, I sought to support “intense questioning, playful curiosity, and deeper thinking” and encouraged students to collaborate with classmates and take the lead teaching each other when they figured something out in their tinkering process (p. 8). Further, the learning environment, or educational makerspace as Kurti and colleagues phrase it, sought to “invite curiosity,” “inspire wonder,” “encourage playfulness,” and “celebrate unique solutions” (p. 10). The use of rapid feedback, described below, played a large role in creating a communitarian ethos where learners were oriented toward noticing connections and supporting each other with their geometric designs.

### Rapid Feedback

The last consideration, rapid feedback, is a core activity comprising tinkering. Rapid feedback is constant during tinkering activity, where one can see if something works and engage in an iterative design process. According to Berland (2016), “[m]aking can work as a best practice for learning by *triangulating rapid feedback* across the aforementioned three aspects of literacy”—material, social, and cognitive (p. 203). Berland describes several forms of social, material, and cognitive feedback summarized in Table 2 (see next page). Because the majority of PSTs’ learning activity emerged through the construction of artifacts and discourse generated during tinkering activity, these various forms of rapid feedback served as a lens for analyzing PSTs’ engagement in computational literacy.

In the following sections, these three design considerations for supporting PSTs’ computational tinkering activity are elaborated as they emerged during the second of three class sessions.

## Preservice Teachers’ Tinkering Activity

The week before the class session discussed in this paper, PSTs were assigned reading excerpts from *Mindstorms* (Papert, 1993), where Papert elaborates on constructionism as a theory of learning as well as meaningful mathematics learning—geometric and spatial concepts in particular. In addition, they were asked to reflect on young children’s artifacts and discourse as they constructed shapes using Logo, also found in Papert’s book. PSTs were directed to the website of Turtle Academy (<https://turtleacademy.com/>), in conjunction with their reading assignments (Papert, 1993), and asked to use this Logo simulation site to explore programming the turtle. While PSTs reflected deeply on the readings, even posing questions related to the purposes of mathematics education and ideas for considering teaching and learning code in their own classrooms, they did not feel capable (or willing) to try Logo on their own before class. After spending 20 minutes sharing questions and ideas with one another related to the readings, PSTs were invited to tinker with Logo. Throughout the remainder of class (about 1.5 hours), PSTs were free to tinker while I interrupted them a few times throughout to reflect and learn additional codes



Table 2. Forms of rapid feedback elaborated by Berland (2016).

Aspects of Literacy	Feedback Type	Description
Social (community)	Social Feedback	Feedback expressed by the community regarding the value (or lack thereof) of what one is making
	Sociomaterial Feedback	Feedback expressed by the community regarding the proper/improper use of tools
	Sociocognitive Feedback	Feedback expressed by the community “about how they would build something differently” (pp. 203–204)
Material (tinkering)	Material Feedback	Feedback “from the actual building of the artifact” (p. 204)
Cognitive (authentic problem spaces)	Cognitive-Material Feedback	Feedback “from watching other people’s problems wrestling with helping me build my artifact” (p. 204)

(looping commands, structure for writing a program, trying a few focused investigations as a whole class).

As soon as PSTs learned a few basic codes they could enter into Logo, they started tinkering to make the turtle move around on the screen. Most PSTs worked on their own at first, and the images they constructed varied widely. Heather, for example, expressed that she “was just trying to put two rectangles together. I was just playing around.” Figure 1 shows both her commands and the resulting image. Entering basic move commands one line at a time was indicative of many PSTs’ initial activity.

Nora experienced struggles related to the playground dimensions, causing the turtle to run off the screen. She articulated that she did not fully understand the dimensions of the playground. With her goal being to tessellate a pattern filling up the screen, she determined that she would first have to find the edges. Tinkering with various sizes of squares allowed her to determine the distance to the edges (see Figure 2, next page).

Nora followed up on this activity, deciding to create her own border so that she could input numbers that were more easily divisible:

I’m trying to figure out how to get to the edges so that I could make a pattern that takes up the whole [screen]. It’s not quite—it’s like a little under 200 and I don’t want to do something like 190 the whole time because I want to use a nice even number. So I’m making my own box first and then I’m going to draw it.

Nora’s activity exemplifies the type of tinkering activity Berland and colleagues (2013) described. Specifically, she used trial and error to find the borders of the Logo playground and received material feedback “from the actual building of the artifact” in the form of visual tracings (Berland, 2016, p. 204). Nora spent the remainder of the class working on a hexagonal tessellation design, attempting to fill the space within her border. Her continued activity is detailed later in the paper, showing the ways her design process drew on all


Commands	
<pre>&gt;fd 100 &gt;rt 90 &gt;fd 200 &gt;rt 90 &gt;fd 100 &gt;rt 90 &gt;fd 200 &gt;rt 270 &gt;fd 100 &gt;rt 270 &gt;bk 200 &gt;rt 270 &gt;fd 100 &gt;rt 90 &gt;fd 200 &gt;rt 90 &gt;fd 100 &gt;rt 270 &gt;fd 200 &gt;rt 270 &gt;fd 200 &gt;rt 270 &gt;fd 400 &gt;rt 270 &gt;fd 100</pre>	

Figure 1. Heather’s tinkering activity shown as commands and resulting image.

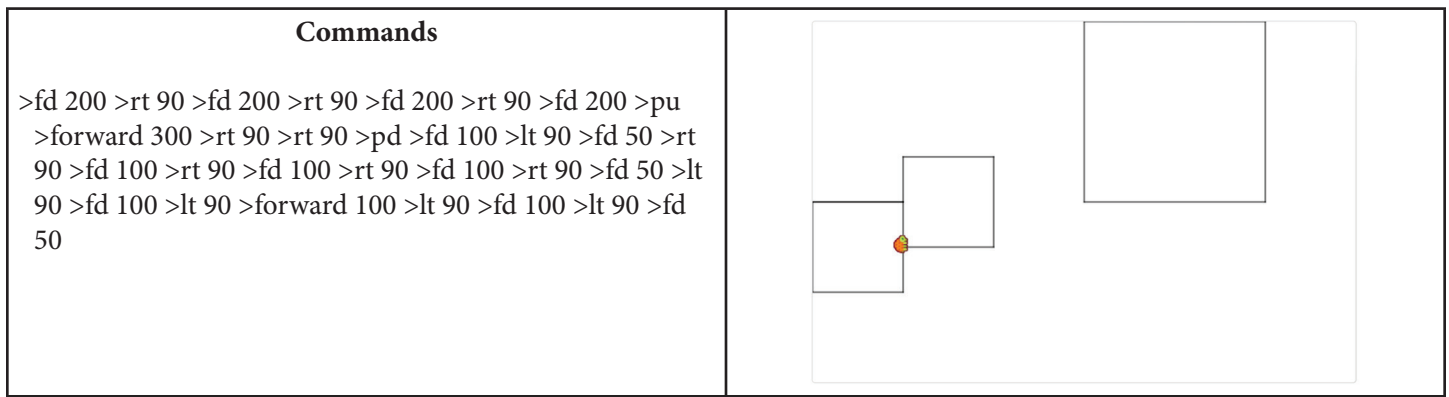


Figure 2. Nora’s initial tinkering commands and resulting image.

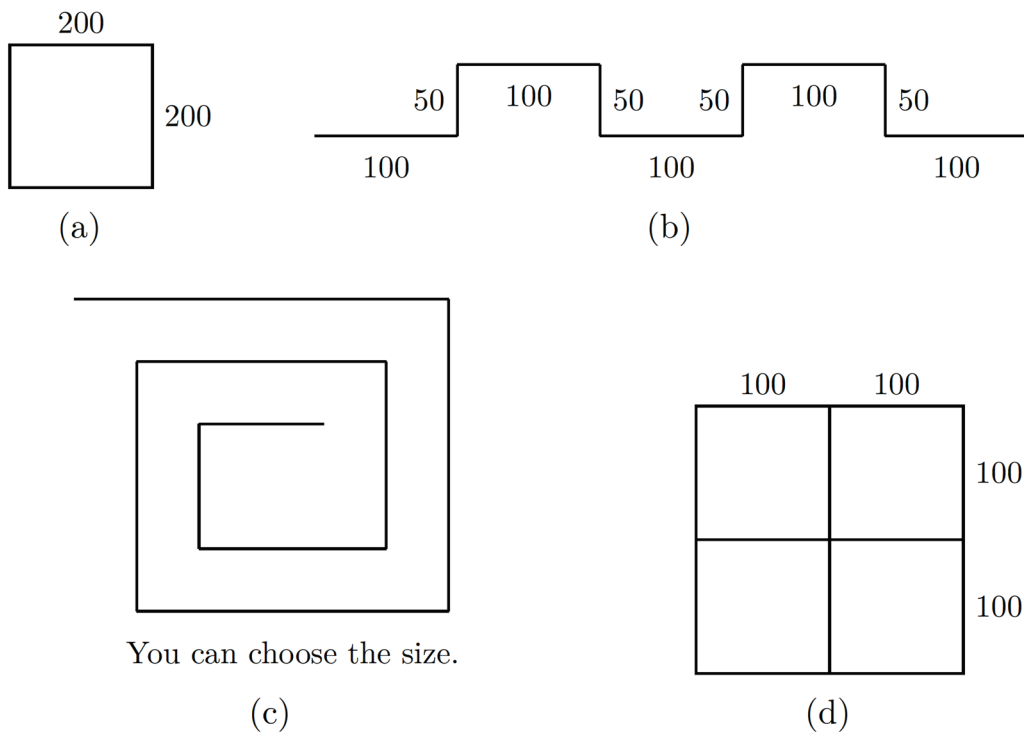


Figure 3. A set of four images (a–d) in which PSTs were asked to write a program. Images from *Programming in LOGO*, an open-education resource from The Center for Informatics Education (ABZ) of ETH Zurich.

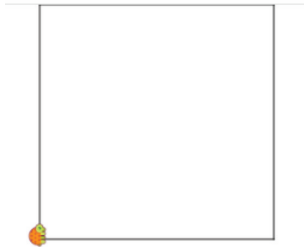
three forms of rapid feedback across the material, social, and cognitive aspects of computational literacy.

**Focused Investigations within Tinkering Activity**

After PSTs had time to tinker with basic commands and discover new ones on their own, I decided to introduce the whole class to the “repeat” command. In a handout given to PSTs, one exercise asked students to try to use “repeat” commands to condense a series of moving and turning commands in order to make a square with 200 units for each side (see Figure 3a).

The repeat command enables Logo to loop commands a specified number of times in order to write more efficient code. Two students, Julie and Lucy, had been trying out repeat commands in their own programs and took a lead role in the class conversation. They not only sought to help their classmates understand the syntax for using brackets with repeat commands, but also to convey the meaning for the repeat command in the context of making a square. The discourse, commands, and resulting image are presented in Table 3 (see next page), demonstrating one way in which a communitarian ethos manifested for PSTs—in this case,

Table 3. Discourse, commands, and resulting image for using the repeat command.

<b>Discourse</b>	
<b>Instructor:</b> So I'm just curious, what are some different ways that you guys would program your turtle to make the "a" image?	
<b>Lucy:</b> You're going to need to make it do everything four times.	
<b>Instructor:</b> Okay, so everything's going to have to happen four times?	
<b>Julie:</b> 2, 3, 4—so yeah! I know what you're going to do.	
<b>Instructor:</b> What am I going to do?	
<b>Julie:</b> You're going to type the word "repeat" and then a space and then a 4 and then a space and then a bracket.	
<b>Lucy:</b> Do everything 4 times.	
<b>Julie:</b> Then fd/forward 200, space, right 90, closed bracket.	
<b>Instructor:</b> Okay, let's see if we agree with what you just said. *Asks class for agreement—not much response at this point.	
<b>Julie:</b> I'm excited now because I made him do it.	
<b>Instructor:</b> Okay, so tell me what all those commands are.	
<b>Julie:</b> So you're telling the turtle to repeat 4 times, move forward with a command of 200. Actually, what you're telling it to do is almost like—you're telling it to make this command of forward 200 right 90 degrees four times. So like you're distributing the 4.	
<b>Instructor:</b> Very algebraic, isn't it? What do you guys think? Does this make sense to you guys—this repeat 4 bracket forward 200 right 90—what is the right 90?	
<b>Julie:</b> To make him turn 90 degrees to the right every time.	
<b>Instructor:</b> So those are my four angles then?	
<b>Julie:</b> Yeah.	
<b>Instructor:</b> And the forward 200 are my four sides?	
<b>Lucy:</b> Yeah.	
<b>Instructor:</b> So let's see what happens when I run this. All right!	
<b>Command</b>	
repeat 4 [fd 200 rt 90]	

taking the lead and “teaching each other to make things” (Berland, 2016, pp. 202–203).

Immediately after creating the square, Julie started talking through the program for image “d” (see Figure 3d). After encountering success, she responded, “I don't even like coding either—but that just felt good to get it!”

As most of the class seemed to express success programming a square, I decided to challenge the class to make a hexagon. They encountered difficulty determining the interior angle measures and ended up creating an octagon with two missing sides. We used this error to finish constructing

an octagon. It was at this point that PSTs started to consider the role of 360 degrees in determining the angle measurements needed to create a hexagon. This information subsequently supported later tinkering activity as PSTs tried constructing polygons with any number of sides. Table 4 (see next two pages) details the way this conversation played out as well as the code and images generated.

In this excerpt detailing PSTs trying to construct a hexagon, they were oriented both to “process- and product-oriented practices” (Sheridan et al., 2014, p. 527). This exemplifies the ways in which constructionist activity supports not only



Table 4. Discourse related to constructing a hexagon with commands and images generated.

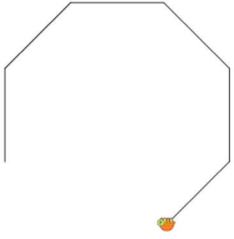
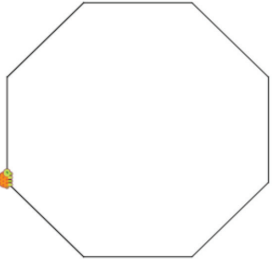
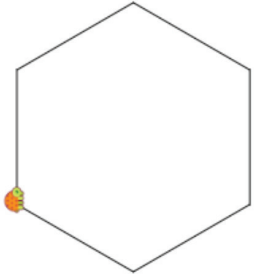
<b>Discourse</b>	
<p><b>Instructor:</b> So I'm just curious—I'm going to challenge you guys. How would we make a hexagon?  <b>Nora:</b> That's what I'm doing—I'm trying.  <b>Matthew:</b> Is it 120?  <b>Julie/Lucy:</b> That's one we were kind of struggling with.  <b>Instructor:</b> Okay—so how many sides does a hexagon have?  <b>All:</b> 6; hex, yeah—6.  <b>Julie:</b> Repeat 6.  <b>Instructor:</b> Okay, so we'll—  <b>Lucy:</b> Yeah, start with repeat 6.  <b>Julie:</b> And we're always going to go forward. Why don't we make it kind of small and start with 100 and then—what I don't know is the degree turn.  <b>Instructor:</b> Okay. So you guys, let's figure it out. We want to make a hexagon. How far is the turtle going to have to turn each time?  <b>Class:</b> Many offer "45 degrees."  <b>Matthew:</b> 120.  <b>Julie:</b> That's [45] what I was thinking too.  <b>Instructor:</b> Where are you guys getting 45?  <b>Class:</b> Many answers [smaller than 90, seems like half maybe].  <b>Instructor:</b> So you're just taking half of 90?  <b>Class:</b> Yeah.  <b>Instructor:</b> Okay, let's try it.  <b>Julie:</b> Well, I'm imagining what 45 degrees looks like. Like a triangle.  <b>Mandy:</b> Yeah.  <b>Julie:</b> Like I'm imagining a 45-degree angle.</p>	
	<p><b>Instructor:</b> Runs program "Repeat 6 [forward 100 right 45]"  <b>Class:</b> Oh (surprised).  <b>Julie:</b> Oh, well, we didn't repeat it enough times.  <b>Instructor:</b> It looks like we're on to another shape, but maybe not a hexagon. So wait, how we can take our mistake and make it—  <b>Matthew:</b> Just change our rotations.  <b>Julie:</b> Make it repeat 8 times.  <b>Instructor:</b> [decided to try repeat 8 first] Okay. You guys think that will work?</p>
	<p><b>Instructor:</b> Runs program "Repeat 8 [forward 100 right 45]"  <b>Class:</b> Oh, cool! We made an octagon.  <b>Instructor:</b> Okay, cool. So now we need to figure out why 45 degrees worked with an octagon.  <b>Julie:</b> What's 360 degrees divided by 8? I'm wondering—it is! It's 45! That's why—so—  <b>Mandy:</b> So you have to do 360 degrees divided by 6.  <b>Julie:</b> So we have to do 360 divided by 6, so 60.  <b>Instructor:</b> So 60—Okay, are you guys ready to try it? Let's do it.  <b>Mandy:</b> I'm excited now.  <b>Instructor:</b> Repeat 6—all right, you want to do the same thing like forward 100? And this time we're going to do—  <b>Class:</b> 60, 60, 60!!!</p>

Table 4, cont'd. Discourse related to constructing a hexagon with commands and images generated.

	<p><b>Instructor:</b> Runs program “Repeat 6 [forward 100 right 60]”</p> <p><b>Instructor:</b> Okay.</p> <p><b>Class:</b> No! Cool! We did it! We figured it out!</p> <p><b>Instructor:</b> So, do you guys feel like you could make—</p> <p><b>Julie:</b> Shapes?</p> <p><b>Instructor:</b> Any polygon?</p> <p><b>Julie:</b> Yeah—I’m ready—give me—challenge.</p>
---	--

their understanding of coding in Logo, but also requires them to draw on geometric properties of shape. Lucy, Julie, Matthew, and Mandy, in particular, not only noticed connections together (communitarian ethos), but also engaged in several forms of rapid feedback (sociomaterial, sociocognitive, and material). For example, sociomaterial feedback emerged after running the following program: Repeat 6 [forward 100 right 45]. When the class expressed surprise at the mistake, Julie exclaimed, “Well, we didn’t repeat it enough times.” She then offered a form of sociocognitive feedback by saying, “Make it repeat 8 times.” At the same time, Matthew suggested that we “change our rotations.” Not evident in this excerpt, but described later, is that Nora was still in the process of trying to tessellate her Logo playground with hexagons. This particular exchange above served as a source of cognitive-material feedback for her design, wherein she received feedback “from watching other people’s problems wrestling with helping me build my artifact” (Berland, 2016, p. 204).

### The Circle

At this point, someone in the class suggested that we all work together to try to make a circle. During this conversation, everyone started offering suggestions and working through problems together, even challenging one another’s suggestions, further exemplifying a communitarian ethos and engagement in the three aspects of computational literacy (social, material, and cognitive). When Julie admitted she only knew how to make straight lines, another classmate, Lucy, offered that the turtle would have to “do a 360.” Julie was struggling to understand, saying, “Yeah, but he’s got to draw at the same time because otherwise he’ll just spin in a circle and not move. You’ve got to make him go forward.” Many students offered suggestions, such as using pi, determining a radius, and so on. At this point, I left the class to tinker on their own. Constructing a circle became the goal of many students for the remainder of the class, while others continued tinkering with their own designs. The discourse in

Table 5 (see next two pages) shows Heather, Kyle, Julie, Matthew, Lucy, Mandy, and me working through possibilities for making a circle in Logo.

This collaborative tinkering episode highlights the typical way in which PSTs engaged in constructing artifacts, in this case, variant circle graphics. This episode in particular shows the ways in which various social and material feedback were used to construct circle designs. For example, while Kyle’s classmates critiqued his idea of using a 180-degree rotation [repeat 180] in conjunction with a “right 2” command, the material feedback from Logo revealed a “baby circle.” However, PSTs continued to consider how they would build the circle differently, engaging in sociocognitive feedback.

At this point, PSTs continued tinkering in Logo to construct various geometric designs. Matthew, drawing on this joint discourse and the resulting images of circles, decided to try to make an infinity symbol (see Figure 4, following Table 5), exemplifying the way cognitive-material feedback played a role in his design. Not only did he draw on Kyle’s idea of “repeat 180,” he also played around with different widths.

Julie and Lucy also continued constructing circles; however, they were focused on understanding why Kyle’s idea of using “right 2” changed the size of the circle. They decided to try different values for the right movement to better understand how it changed the circle’s size (e.g., right 10 and 20). When they entered a value of “right 2,” their larger circle ran off the screen, creating the image shown in Table 6 (following Figure 4). The discourse shows the conversation related to their thinking and tinkering.

This led to a conversation about trying different values for the repetition, the forward movement, and the right 20 value so that the image would reveal the “sides” that were really being created. While Lucy and Julie’s continued work constructing circles also exemplifies the use of cognitive-material feedback like Matthew’s, their activity was more of an investigation to better understand why the different values (e.g., right 1 versus right 2) produced different-size circles. In

Table 5. Discourse, commands, and images generated while constructing a circle.


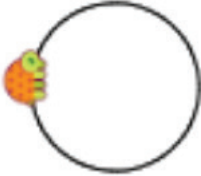
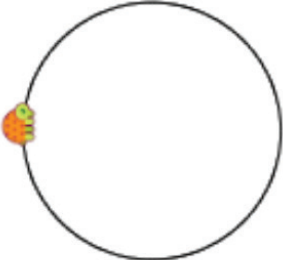

<b>Discourse</b>	
<p><b>Instructor:</b> (to Heather) So what are you wondering—can you do what?</p> <p><b>Heather:</b> Well, it's easier with that [referring to polygons with a certain number of sides] because you could take the number of sides and do 360 divided by the number of sides. But with the circle, there's not really sides.</p> <p><b>Instructor:</b> So, we're probably not going to make a perfect circle.</p> <p><b>Kyle:</b> Would it be 180?</p> <p><b>Julie:</b> 180 what?</p> <p><b>Kyle:</b> Repeat 180—it starts to make a circle, right?</p> <p><b>Julie:</b> So that makes him do it, but how do you make him draw it? Like how do you make him draw the arc?</p> <p><b>Instructor:</b> Hold on—I like what you said. So, Kyle, you said repeat 180.</p> <p><b>Kyle:</b> Then I don't know what else to do.</p> <p><b>Julie:</b> Nothing, I didn't get an arc.</p> <p><b>Instructor:</b> So, what are we going to have to do—let's say we just want to make an arc—what do you think we would do?</p> <p><b>Kyle:</b> Move him forward.</p> <p><b>Instructor:</b> He's going to have to go forward, right?</p> <p><b>Kyle:</b> Right.</p> <p><b>Matthew:</b> Right.</p> <p><b>Instructor:</b> Some amount. Does it matter how much?</p> <p><b>Lucy:</b> Not really.</p> <p><b>Class:</b> Suggests numbers such as 1.</p> <p><b>Instructor:</b> 1? Okay. But if he goes—</p> <p><b>Kyle:</b> So, he has to turn, so would that be right?</p> <p><b>Julie:</b> 180—360.</p> <p><b>Kyle:</b> 1, right?</p> <p><b>Instructor:</b> 1?</p> <p><b>Julie:</b> 1 because that should make half a circle, right?</p> <p><b>Instructor:</b> That will make a half circle?</p> <p><b>Julie:</b> Possibly.</p> <p><b>Instructor:</b> [runs code]</p>	
	<p><b>Command</b></p> <pre>repeat 180 [fd 1 rt 1]</pre>
<b>Discourse</b>	
<p><b>Julie:</b> Yes, it does!</p> <p><b>Heather:</b> How did it do that?</p> <p><b>Julie:</b> Now do repeat 360 [forward 1 right 1]</p> <p><b>Instructor:</b> Okay.</p> <p><b>Kyle:</b> Wait, wait, wait—no, no, no—stop.</p> <p><b>Julie:</b> What?</p> <p><b>Kyle:</b> We're already going right—we need to keep going right—try putting 2 for right.</p> <p><b>Julie:</b> Why, though?</p> <p><b>Matthew:</b> Because it's 360.</p> <p><b>Julie:</b> Why are you putting 2 for right?</p> <p><b>Kyle:</b> Because you're doing it twice.</p> <p><b>Julie:</b> No.</p> <p><b>Instructor:</b> [Runs Kyle's idea: Repeat 180 forward 1 right 2]</p>	

Table 5, cont'd. Discourse, commands, and images generated while constructing a circle.

	<p style="text-align: center;"><b>Command</b></p> <pre>repeat 180 [fd 1 rt 2]</pre>
<p style="text-align: center;"><b>Discourse</b></p> <p><b>Kyle:</b> See, it made a baby circle.  <b>Julie:</b> No, it made a baby circle. But the more right you turn, the more narrow your circle will be.  <b>Matthew:</b> Why don't you use the 360, then?  <b>Kyle:</b> Oh, okay, got you.  <b>Julie:</b> The greater your . . . can we go negative on the right? No, you can't go negative.  <b>Kyle:</b> Can we do 360?  <b>Instructor:</b> I don't know—let's try it [repeat 360 forward 1 right 1].</p>	
	<p style="text-align: center;"><b>Command</b></p> <pre>repeat 360 [fd 1 rt 1]</pre>
<p style="text-align: center;"><b>Discourse</b></p> <p><b>Mandy:</b> I just did it—I did a circle!!! I was doing what you were just saying and I just had it repeat twice.  <b>Lucy:</b> How did you [Mandy] do that [with the width and color]?  <b>Instructor:</b> Okay, tell us how to make the color.  <b>Mandy:</b> Setcolor and then choose a number.  <b>Heather:</b> 5  <b>Mandy:</b> And then setwidth—I chose a random number—10.  <b>Instructor:</b> Okay [Instructor enters Mandy's code].</p>	
	<p style="text-align: center;"><b>Commands</b></p> <pre>setcolor 5 setwidth 10 repeat 360 [fd 1 rt 1]</pre>
<p style="text-align: center;"><b>Discourse</b></p> <p><b>Class:</b> Wow! Oh, I like that! That is really cool! Try 15.  <b>Instructor:</b> These are awesome—I haven't seen the thick circle; this is great!</p>	

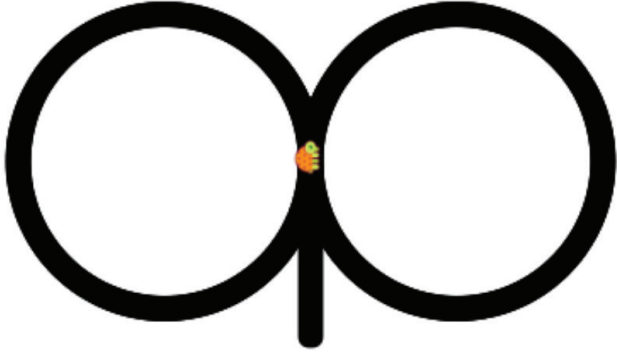
	<p style="text-align: center;"><b>Commands</b></p> <pre>setwidth 10 fd 100 repeat 180 [fd 3 rt 2] repeat 180 [fd 3 lt 2]</pre>
---	--

Figure 4. Matthew's infinity symbol image and commands.

Table 6. Julie and Lucy's commands, resulting image, and discourse related to constructing circles.

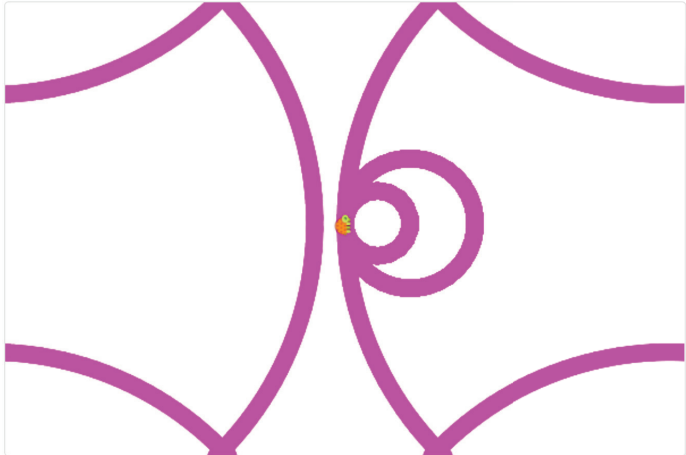
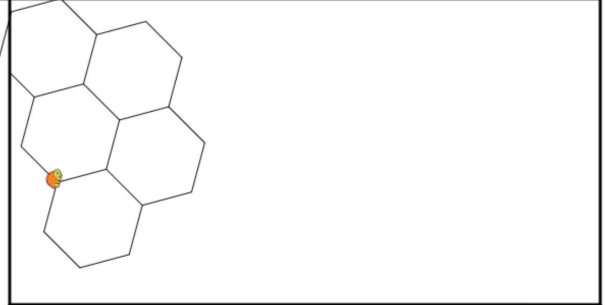
	<p style="text-align: center;"><b>Commands</b></p> <pre>setcolor 5 setwidth 15 repeat 360 [fd 2 rt 2] cs repeat 360 [fd 10 rt 10] repeat 360 [fd 10 rt 20] repeat 360 [fd 10 rt 2]</pre>
<p style="text-align: center;"><b>Discourse</b></p> <p><b>Lucy:</b> We don't know why that happened. We're trying to figure out the correlation between this number and this number in terms of how to make them bigger and smaller and manipulate the circle.</p> <p><b>Instructor:</b> Wait, explain that to me again.</p> <p><b>Julie:</b> Okay, so we knew that when Kyle did forward 1 right 2, we knew we got a circle this size. But then when we tried forward 10, right 10, we ended up with this size, which is very similar to forward 1, right 1. So, we can't seem to figure out why 10 and 1 don't seem to make anything different. Then we tried forward 10, right 20, similar to when Kyle said forward 1, right 2—this gave us a similar smaller circle. So, then we tried to figure out—okay so what are—why—</p> <p><b>Lucy:</b> What's changing?</p> <p><b>Julie:</b> Why? Why are these shapes the same size as when we do forward 1, right 1—forward 1 right 2. Why is there zero—why is there no difference?</p> <p><b>Lucy:</b> Then we did that—repeat 360 [fd 10 rt 2]—and ended up with this [larger circle that runs off the screen].</p> <p><b>Julie:</b> Then we did this to try to manipulate—is it the right or is it the forward that changes?</p> <p><b>Instructor:</b> And then that made it go off the screen which is what made it do that big thing—it was really just a big circle. Okay, so one thing—because you have the setwidth on so wide, you're not seeing something. Consider changing your setwidth to a lower number.</p> <p><b>Julie:</b> Just go back to 1.</p>	



Table 7. Discourse, commands, and images related to Nora's tessellating project.

<b>Discourse</b>		
<p><b>Instructor:</b> You got it “To Hexagon.” So what is your “To Hexagon”?</p> <p><b>Nora:</b> Well, because I already had 2 sides, I only needed it to do 4. I'm trying to figure out, like how to do a program to just reset it in between things [the turtle's location when drawing the next hexagon].</p> <p><b>Instructor:</b> What if you do rotate 180 and then go forward?</p> <p><b>Nora:</b> Yeah, that's what I've been doing—I've been rotating 120 and then going forward the amount 50 and then doing the hexagon, so I guess—do I have to . . .</p> <p><b>Instructor:</b> [noticed that Nora created a “To Move” program] To Move—yes, teaching it to move.</p> <p><b>Nora:</b> . . . right 120 forward 50</p> <p><b>Instructor:</b> Yeah, and then end.</p> <p><b>Nora:</b> I did the wrong angle—run—ah, he went the wrong way.</p>		
		<b>Commands</b>
		<pre>&gt;pu &gt;fd 150 &gt;lt 90 &gt;fd 300 &gt;rt 180 &gt;pd &gt;setwidth 3 &gt;fd 590 &gt;rt 90 &gt;fd 300 &gt;rt 90 &gt;fd 580 &gt;rt 90 &gt;fd 300 &gt;pu &gt;rt 90 &gt;fd 50 &gt;rt 45 &gt;setwidth 1 &gt;pd &gt;fd 50 &gt;rt 60 &gt;fd 50 &gt;rt 60 &gt;fd 50 &gt;rt 60 &gt;fd 33 &gt;pu &gt;fd 17 &gt;rt 60 &gt;pd &gt;fd 50 &gt;rt 60 &gt;fd 50 &gt;pu &gt;right 60 &gt;fd 50 &gt;pd &gt;lt 60 &gt;fd 50 &gt;repeat 4 [rt 60 fd 50] &gt;pu &gt;lt 60 &gt;fd 50 &gt;lt 60 &gt;pd &gt;repeat 4 [fd 50 lt 60] &gt;pu &gt;rt 120 &gt;fd 50 &gt;pd</pre>
	<pre>TO hexagon   repeat 4 [rt 60 fd 50] END &gt;hexagon</pre>	<pre>TO move   pu lt 180 fd 50 pd END &gt;move &gt;hexagon</pre>

this way, they were engaging in many of the standards for mathematical practice (e.g., make sense of problems and persevere in solving them, model with mathematics, look for and make use of structure) while attempting to connect the Logo code to the resulting image.

### Teaching Logo a Program/Word

Toward the end of class, I decided to interject one last time in order to show PSTs the structure for writing programs in Logo using the previous example of the circle and square. This involved teaching Logo a word (i.e., program) using the structure “TO square > Repeat 4 [forward 100 right 90] > END.” Nora, the student who was trying to tessellate

hexagons to fill in the screen, integrated this new technique to support her work. Table 7 shows our conversation, her code, and the resulting image created.

While Nora was unable to complete her tessellated design in class, she was able to construct a relatively complex geometric pattern, in large part by drawing on social, material, and cognitive dimensions of computational literacy. Similar to the way in which Matthew's infinity design showed evidence of cognitive-material feedback related to the circle discourse, Nora's design indicates the way she incorporated the discourse and struggles of her peers to create her partial hexagons (notice that her program “TO hexagon” only draws four of the sides). Not only does she integrate the thinking of her peers,

she continues to revise her code to integrate more advanced techniques, such as teaching Logo programs like “TO move.” She combines the different ideas learned through tinkering and classroom discourse to eventually combine the programs “move” and “hexagon” in order to tessellate the pattern.

## Discussion

A community ethos and the making of genuine artifacts emerged as PSTs worked to construct designs in Logo. Notable in the discourse during tinkering is PSTs’ integrated engagement in learning mathematical concepts (e.g., determining the interior angle measures of regular polygons) and computational concepts (e.g., looping, sequencing), as well as engaging in computational and mathematical practices. The goal of this tinkering activity aimed to support PSTs to integrate computational thinking/computational literacy into their learning of geometry and spatial concepts. As Lucy and Julie demonstrate, not only were they trying to investigate connections between the Logo code and the resulting image, but they were also engaging in many of the mathematical practices simultaneously. While artifacts were constructed by individuals (e.g., Nora’s tessellated hexagon patterns, Matthew’s infinity symbol), some were driven by collaborative efforts, such as making a circle. Although not apparent in the data shared, these PSTs have spent months together developing as mathematics educators. It would be shortsighted to assume that one could simply ask a group of students to tinker with Logo and expect to spark the kinds of conversations that occurred during this class session. PSTs had many experiences learning together as well as shared norms for learning (e.g., making mistakes is part of learning, we can learn from attending to the thinking and strategies of others).

Drawing on Berland’s (2016) descriptions of rapid feedback, there is evidence that PSTs engaged in all the feedback types described. Material feedback was common, occurring as soon as PSTs clicked “run” on their programs. In this way, they could determine their errors or successes—the feedback was immediate. Usually this material feedback led to a social interaction, both with me and their classmates. By the nature of the data collected (a lapel microphone worn by me), many of the student-student interactions were not captured. However, the discourse related to generating a circle represents the types of conversations students were having. Regarding social feedback, classmates were genuinely excited by all the images created—even those that the creator didn’t intend. I also noticed that my own comments to PSTs were overwhelmingly positive. There was a contagious energy among PSTs—they wanted to show one another their creations and both gave and received much praise. Sociomaterial feedback was also common as classmates wanted to help one another

successfully construct their designs (either bringing ideas to fruition or working through bugs in their code). They would share commands such as “setwidth,” “setcolor,” and even ways of writing repeating commands, evident in Julie and Lucy’s discourse above. As the instructor, I found myself providing sociomaterial feedback as well—sometimes as simple as telling them the proper syntax for writing codes (no space between set and color; repeat needs to indicate what you are repeating) and suggestions for troubleshooting such as trying smaller values to prevent the image from running off the screen. Although in the early stages I wasn’t expecting to see much sociocognitive feedback, as everyone was encountering coding with Logo for the first time, the conversation about creating a circle shows PSTs offering differing opinions on how to best write code. Kyle’s idea, for example, using a repeat 180 command and moving the turtle forward 2 units each time, was countered by Matthew, Julie, and Lucy, who felt that repeat 360 and move forward 1 unit might be a better approach. This even led Julie and Lucy to continue exploring the differences between these various differences to figure out what changes appeared in the circle images.

## Conclusion

While the educational technology community is embracing makerspaces and creative/connected coding activity, as well as design and systems thinking among youth learners, these spaces and ways of thinking they engender are sometimes difficult for teachers to implement in formal K–12 learning environments. Many lament a teacher education system that doesn’t yet prepare teachers to support, for example, computer science education. And while this might increase the number of teachers prepared to teach computer science high school courses, there is a need to consider the full range of K–12 education.

The approach described in this paper is one attempt to support the computational literacy of elementary PSTs, namely, by creating opportunities for PSTs to experience learning situations, such as makerspaces, for themselves in order to see themselves as doers—or rather, makers—as they engage with mathematical and computational concepts and tools. These experiences programming the Logo turtle to construct authentic artifacts in the form of two-dimensional geometric graphics not only supported PSTs’ understanding of core geometric and spatial concepts, but also helped them to make connections between mathematics and computational literacy.

While elementary PSTs need opportunities to experience computing concepts and practices such as tinkering and constructing authentic artifacts, their conceptual knowledge of computing (and in this case mathematics) is only a

foundation for helping them consider the thinking and practices they will need to facilitate for their future classrooms. Future work might consider creating opportunities for PSTs to plan their own constructionist-oriented mathematics lessons and try these out with classmates and in their field placements. In this way, they can work to consider the thinking of their K–5 students, their response to students, and the kinds of questions they might pose.

## References

- Akcaoglu, M., & Kale, U. (2017, November 18). *Call for manuscripts: Special issue: Tinkering in technology-rich design contexts*. Retrieved November 18, 2017, from <https://spark.adobe.com/page/ZSSkH3ZyYa8KF/>
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48–54.
- Berland, M. (2016). Making, tinkering, and computational literacy. In K. Peppler, E. Rosenfeld Halverson, & Y. B. Kafai (Eds.), *Makeology: Makers as learners* (Vol. 2, pp. 201–209). Abingdon, UK: Taylor & Francis.
- Berland, M., Martin, T., Benton, T., Petrick Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, 22(4), 564–599.
- Brennan, K., Balch, C., & Chung, M. (2014). *Creative computing*. Harvard Graduate School of Education. Retrieved from <http://scratched.gse.harvard.edu/guide/files/CreativeComputing20141015.pdf>
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Presented at the American Educational Research Association, Vancouver, British Columbia, Canada. Retrieved from <http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf>
- Common Core State Standards Initiative. (2010). *Common Core State Standards for Mathematics*. Washington, DC: National Governors Association Center for Best Practices, Council of Chief State School Officers.
- Dick, T. P., & Hollebrands, K. F. (2011). *Focus in high school mathematics: Technology to support reasoning and sense making*. Reston, VA: National Council of Teachers of Mathematics.
- diSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38–43.
- Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.
- Kurti, R. S., Kurti, D. L., & Fleming, L. (2014). The philosophy of educational makerspaces: Part 1 of making an educational makerspace. *Teacher Librarian*, 41(5), 8–11.
- Lifelong Kindergarten Group. (2017, July 16). *Scratch—Imagine, Program, Share*. Retrieved July 16, 2017, from <https://scratch.mit.edu/>
- National Council of Teachers of Mathematics. (2014). *Principles to actions: Ensuring mathematical success for all*. Reston, VA: National Council of Teachers of Mathematics.
- Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). New York, NY: Basic Books.
- Sheridan, K., Halverson, E. R., Litts, B., Brahms, L., Jacobs-Priebe, L., & Owens, T. (2014). Learning in the making: A comparative case study of three makerspaces. *Harvard Educational Review*, 84(4), 505–531.
- 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.
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society*, 366(1881), 3717–3725.
- Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, 60(4), 55–62.

---

Dr. Valentine is an Assistant Professor in Mathematics Education at West Virginia University. Her research interests focus on learners' mathematical reasoning, particularly related to geometry and spatial concepts. Dr. Valentine also investigates the design and various forms of learning occurring in nontraditional learning environments (e.g., summer game design camps, leisure, and play).