Bridging the Divide: Exploring Affordances for Interdisciplinary Learning

Madison Knowe, Melissa Gresalfi
Madison.l.knowe@vanderbilt.edu, Melissa.gresalfi@vanderbilt.edu
Vanderbilt University

Abstract: This study investigates how the design of hybrid mathematics and computational activities influences the ways in which students leverage ideas from both disciplinary topics. We examine two design cycles of a computer programming summer camp for middle school students which foreground computational thinking and then mathematics alongside computational thinking respectively. We review the rationale for each design iteration, the trends we saw in students' engagement, and the implications for students' reasoning. Findings of this study demonstrate the importance of thinking critically about the boundary objects that are included in design that support students to make bridges between multiple disciplinary practices.

Introduction

There is significant support for the integration of computational thinking into mathematics classrooms due to content similarities in the domains of computer programming and mathematics (Pérez, 2018; Weintrop et al., 2016). These similarities suggest that learning programming and mathematics simultaneously might be a reasonable way to accomplish a goal of integrating computer science into K-12 classrooms (Harel & Papert, 1990; Gadanidis et al., 2017). However, little research has investigated these claims or, more profoundly, how the synergies between these disciplines might be designed for in such a way that both mathematical thinking and computational thinking would develop. Indeed, bringing two fields together necessarily creates tensions for designers, teachers, and learners; it is these tensions that constitute the focus of this paper. Specifically, we examine two rounds of design in a design-based research project that took two different approaches to support students to leverage mathematical and computational tools with personal agency. We review the design and rationale for each round of implementation, and then consider the implications for students' interdisciplinary reasoning as it is connected to design.

Background and framing

Computer science education is becoming increasingly common in the K-12 curriculum, and educators are adapting to the demand to equip their students with new computational thinking skills (Yadav et al, 2016). This has led teachers and other educational leaders to incorporate computer science through the adjustment or expansion of frameworks, pedagogies, standards, and lessons in the K-12 curriculum (Sentance & Csizmadia, 2016; Yadav et al, 2016).

However, it has been well-established that integrating different learning contexts is neither easy nor straightforward; whether the goal is to connect two disciplines or to connect in-school with out-of-school learning, there is significant evidence that learning is supported and transformed by the norms, tools, and practices of the contexts in which activity is situated (Nasir & Hand, 2008). Central to this idea is that the ways that learning environments are designed (for example, inviting students to think collaboratively one or two open-ended questions versus asking them to work independently on a set of 20 similar questions) fundamentally transforms the nature of what students ultimately learn. An important implication of this work is that "knowledge" is neither static nor abstract but is instead constituted in relation to the learning context (Engle, 2006), and therefore it is no simple or straightforward thing for knowledge to travel between contexts or to connect with new and different knowledge communities (Redish & Kuo, 2015). If learning is truly a *joint accomplishment*, then careful attention needs to be paid both to design and to learning.

How should an activity that invites mathematical and computational thinking be designed? Can one idea be introduced in service of another? Can one set of practices find a home amongst a set of sometimes intersecting but distinctly different practices? To better understand these questions, we draw from work on hybridity, which involves integrating norms and practices from one cultural activity into another. Although there have been many frameworks offered that address hybridity, here we rely on the idea of *boundary crossing* as a lens to examine student activity. As Akkerman & Bakker (2011) write, "A boundary can be seen as a sociocultural difference leading to discontinuity in action or interaction" (p. 133). While the practices and content of mathematics intersect with that of computer programming, there are clear sociocultural differences between the disciplines of mathematics and computer programming. This is especially true with respect to school-based mathematics, the context in which the participants of this study are most familiar with mathematics, which often involves engaging a set of facts and rules to be taught and practiced as an end in itself, in contrast to computer programming which includes design (Kafai, 2016), iteration, and revision. In this work we explore whether and how the particular designs that we undertook served to create bridges between the two disparate worlds of mathematics and programming. Using in-depth analysis of student and teacher discourse, we consider how the boundaries were drawn around these worlds, and how individuals and tools served as boundary objects to connect them. In using this lens, we seek to better understand not whether one design was more successful than another, but rather, how our specific design decisions played out with respect to how different worlds were connected.

Methods

The designs and data presented in this paper come from a larger design-based research study about how student thinking about computation and mathematics might co-develop. Design-Based Research (DBR) is an approach to conducting research with the goal of uncovering why something works and involves testing conjectures and investigating the relationship between activity and outcomes (Sandoval, 2014). As such, DBR is explicitly a theory-building activity, and requires intentional specification of the theories that underlie both initial assumptions and interpretations of data. In particular, theories of how people learn necessarily inform the initial designs undertaken. In contrast to implementation studies that examine whether a particular design is effective, design-based research includes multiple iterations and should specify the changes implied for next rounds of implementation. These cycles of research are a key distinction between understanding why this particular design might function and developing a more nuanced understanding of the relationship between designs and learning more broadly. Thus, the goal of these design iterations is to move beyond understanding what works to consider and be able to account for why it might have worked, with an eye towards not only the success of this design, but the potential success of future designs.

Participant description

The study focuses on the first two rounds of DBR; two free five-day summer camps that were held in two consecutive years. The camps, called *Code Your Art,* were designed for and attended by middle school students and advertised in public middle schools in a medium-sized city in the southern United States. In each year two classes were taught that focused on similar sets of activities; in Year one 30 students enrolled and were split across two classrooms; in Year two 24 students were enrolled and split across two classrooms. Each class was co-caught by two experienced teachers who taught elementary or middle school mathematics during the regular school year. Each classroom was also supported by teaching assistants and researchers who rotated through the classes as needed. The researchers, teachers, and teaching assistants made up the research team that designed and implemented these camps. For this analysis we will focus our interaction analysis on one participant from each year who both consented to all data collection for this study.

Design description

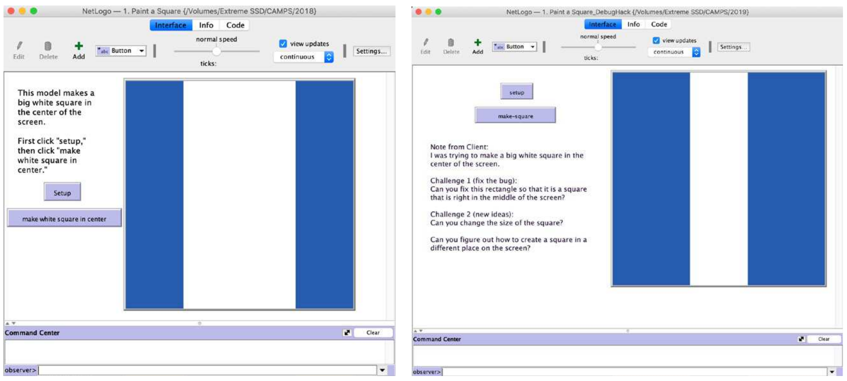
Code Your Art camp introduced students to the ways that a programming environment could produce visual effects and dynamic images. Students used a programming environment called NetLogo (Wilensky, 1999), a multi-agent environment that uses a text-based programming language to manipulate static pixels (patches) and movable agents (turtles). Students were told that by the end of the week their goal was to create a “final design,” which they would share with visitors at a “Gallery Walk” on the final day. Each day students learned a new set of ideas or strategies for creating different visual effects, with the idea that students would use and incorporate those effects that they found relevant to their envisioned design. The explicit stated goal of the research team was for students to see NetLogo as a set of tools that they could use expressively, but how to best position students to think of programming in this way was an open question. One specific question involved whether and how to engage the underlying mathematical ideas that are central to NetLogo, including the coordinate plane, inequalities, number lines, slope, angles, absolute value, distance, quantity and scale.

In Year 1 the design of the camp foregrounded computational thinking, accomplished by offering students a set of NetLogo models that they could explore and modify. Although many of these models involved mathematical ideas, the intentional design of the camp was to engage with underlying mathematical ideas when they were raised by students, such that students’ goals for their own expressive models guided the conversations. In Year 2, based on analyses from Year 1, the design team modified the camp, developing a set of activities that foregrounded mathematics and mathematical connections to NetLogo. Our analysis will focus on how this design iteration changed students' engagement in authoritative problem solving and expressive design in this hybrid math and computational learning space.

Debugging activities

At the end of the third day of Year 1, teachers and researchers designed a set of *debugging models* as an emergent response to significant growth but also some common challenges that students were facing. These models were intended to help students wrestle with and understand the challenges that they were facing as well as see how much they had already learned in NetLogo in order to shift students' ownership of the code.

Each debugging model contained a short description of what should happen in the model if it worked properly; students could read and edit the code embedded in a button in order to fix the "bug." In this analysis, we will focus the first of seven debugging models, *Paint a Square*. The Paint a Square debugging model is a hybrid activity in that its solution requires thinking about mathematics and about programming to create a white square by operating on a set of "patches," squares laid out on a coordinate grid in the model. Changing the color of specific patches requires specifying the patch's x-coordinate (pxcor) and y-coordinate (pycor) using the NetLogo coding language. In Paint a Square, when students clicked the "Setup" button, all of the patches in the window turned blue. When they clicked "make white square in center" a vertical white stripe appeared (Figure 1 Left). The code that produced this effect in Year 1 was "ask patches with [pxcor > 50 and pxcor < 150] [set pcolor white]" which asked all of the patches in the model with an x-coordinate greater than 50 and less than 150 to turn white. Because the window was set up for all students to operate on a coordinate grid with x-coordinates and y-coordinates from 0 to 200, this button colored a white stripe in the middle of the screen. In Year 2 the model was adapted to give directions from the perspective of a client and provide the student with "challenges" rather than a statement about what the model was expected to do (Figure 1 Right). In Year 2 the coordinate plane that is operating behind the NetLogo model ranged from -16 to 16 on both the x and y axis, and therefore the buggy code given to students in Year 2 was "ask patches with [pxcor > -5 and pxcor < 5] [set pcolor white]".


NetLogo — 1. Paint a Square {/Volumes/Extreme SSD/CAMPS/2018}
Interface
Info
Code
Edit
Delete
Add
Button
normal speed
ticks:
view updates
continuous
Settings...
Setup
make white square in center
Command Center
Clear
observer>
NetLogo — 1. Paint a Square_DebugHack {/Volumes/Extreme SSD/CAMPS/2019}
setup
make-square

This model makes a big white square in the center of the screen.

First click "setup," then click "make white square in center."

Note from Client:
I was trying to make a big white square in the center of the screen.

Challenge 1 (fix the bug):
Can you fix this rectangle so that it is a square that is right in the middle of the screen?

Challenge 2 (new ideas):
Can you change the size of the square?

Can you figure out how to create a square in a different place on the screen?

Figure 1. (Left) The Year 1 model after pressing "Setup" and then "make white square in center"; (Right) The Year 2 model after pressing "Setup" and then "make-square."

As stated, we argue that these debugging activities were specifically hybrid activities as "fixing the bugs" required using mathematics within a text-based computer programming activity. To solve this debugging model, students needed to use correct NetLogo syntax, have an understanding of patches as an agent in NetLogo, and use appropriate mathematical concepts such as the coordinate plane. Because of this intersection of mathematics and computational thinking, this set of activities in the camp was ideal to investigate the foregrounding of computational thinking in Year 1 as compared to the foregrounding of mathematics in Year 2. Our analysis will therefore specifically focus on Paint a Square, the first of these seven debugging activities. Because Paint a Square was students' first engagement with debugging this analysis gives a good representation of how the design of the camp and the implementation of the activities that led up to debugging led to differences in how students leveraged ideas from mathematics and computation in the debugging models.

Data collection

Data was collected throughout both camps through whole group video recordings, screen capture video from software installed on every computer, pre- and post-camp written surveys, and interviews. A go-pro was worn by a teacher in each class, a pivoting camera tracked and recorded teacher-student interactions, a standing camera recorded whole group interactions, and individual screen captures recorded individual students' work. Most of the screen-capture videos have a record of both the entire screen and the students face, but on some videos only the screen is visible. The focus of the analysis will be only on the data collected about the foregrounding design and student engagement in the Paint a Square debugging activity described above.

Analysis

Our analysis focused on two aspects of the camp: 1) the ways the designs created opportunities to bridge the disciplines of mathematics and computer science, and 2) how students' engagement bridged the two disciplines, and what designed supports appeared to help make that occur. To make sense of designed opportunities, we reviewed all whole-class activity that took place before the debugging episodes and marked times when mathematics and computational thinking co-occurred. We then considered those episodes, paying attention to when they took place and in what form.

To understand student engagement, we focus on two cases (Brianna from Year 1; Caleb from Year 2), who are representative of the trends in disciplinary approaches to problem solving in the first debugging activity, Paint a Square (Gresalfi et al, 2020), from each of their respective years. These cases represent times when the coordinate plane was operationalized concurrently with the NetLogo code. Using methods of discourse analysis, we looked for times when a participant made connections between the mathematics behind the NetLogo model (such as inequalities and the coordinate plane) and the code that communicates with the NetLogo model. For example, to draw a vertical line at x=-50 and a horizontal line at y=150, one might type "ask patches with [pxcor = -50 and pycor = 150] [set pcolor white]". This code would color a single patch located at the coordinate (-50, 150) white and therefore not reach the goal of the participant, but it would demonstrate that the participant is making a connection between coloring the patches at pxcor = -50 white and coloring the x-coordinates in the model that create a vertical line at x=-50 white. Evidence of bridging math and computer programming holds no matter if the code accurately achieves the mathematical goal in mind. In contrast, if a participant wanted to color a single patch at (-50, 150), they might type "ask patches with [pxcor < -50, 150] [set pcolor white]". This would demonstrate that the participant is not attuning to how the code pxcor relates to the x-coordinate or the relationship of the inequality to the values typed in the code, and therefore not engaging in bridging math and computer programming. These scenarios are also dependent on the context in which the code is typed and include the discourse around the creation of the code.

Findings

In the moment-to-moment interactions between teachers and students, we saw evidence that the different designs in Years 1 and 2 offered different resources that allowed participants to leverage computational and mathematical disciplinary ideas and practices. Specifically, we demonstrate how the Year 1 focus on ensuring that students' work and thinking drove the introduction of new mathematical content meant that there were clear boundaries between the worlds of math and computer programming, that were mainly bridged only by adults who served to broker relationships. In the examples from Year 1, students were reliant on one-on-one interactions with adults to help them connect mathematical ideas with the concepts and syntax of NetLogo. In contrast, the design in Year 2 meant that there were several tools that were provided by adults in anticipation of students needing mathematical connections to NetLogo that appeared to serve as boundary objects to connect the two disciplines. Our analysis of the whole group video data in both years revealed differences in the timing, resources, and motivations behind the opportunities that served as boundary objects for the Paint a Square debugging activity which resulted in changes in the ways in which students leveraged mathematics and computation concurrently.

Opportunities to bridge mathematical and computational thinking

As noted above, in Year 1 the design of the camp approached activities with an explicit goal of open exploration and modification of models in NetLogo. This meant that the underlying mathematics of the models was engaged as students generated a need for those ideas. Therefore, while students did often initiate conversations based around a need for a bridging opportunity between mathematics and NetLogo, it was generally adults who offered information about those bridges, becoming brokers between mathematics and NetLogo. In addition, in Year 1 an designed bridging activity took place on the second day of the camp that asked students to act as patches by holding colored cards and standing on a coordinate grid. The activity introduced a new set of commands that involved patch coordinate location (for example, "Ask patches if [pycor > 1] [set pcolor green]"). Before the commands were given, teachers noted that students were standing on a coordinate plane and each student was assigned a point (for example, 0,1) on which to stand. They engaged in a discussion about how the coordinate plane they were standing on was a limited version of the NetLogo model which was a limited (only positive integers) version of the coordinate plane. In this discussion, teachers also reviewed some basic properties of the coordinate plane. Students then held up various colored cards based on their location in response to NetLogo commands given. We argue that this moment of coordinate plane review offers a bridge between NetLogo and the mathematics of the coordinate plane.

In Year 2, based on analyses from Year 1, the design team modified the camp, developing a set of anticipatory activities that foregrounded mathematics and mathematical connections to NetLogo. For example, on the first day of camp in Year 2 students engaged in a Stadium Card activity similar to Year 1 but were much more explicit about the coordinate plane. Teachers led a whole group discussion that involved every student in the room contributing something they knew about the coordinate plane. At the same time, a researcher modeled students' inputs on the NetLogo model on the projected screen while another teacher drew students' suggestions on a white board. As students gave suggestions, the NetLogo model and white board representation of a coordinate plane were used interchangeably. This allowed students and teachers to bridge traditional school-math language such as words like "quadrant 1" and "x-coordinate" in place of or in conjunction with the NetLogo equivalent such a "top right of the model" or "pxcor." This is just one of four similar bridging activities that took place before debugging. This anticipatory design of utilizing the mathematics allowed for boundary crossing between mathematics and computation to occur collectively rather than in on-on-one interactions as it did in Year 1. The ways in which these boundary crossing opportunities were enacted had repercussions for student engagement and interactions with teachers in each year. We will also note that there were drawbacks to this emphasis on mathematics, as some students' interest in the camp waned that day, which was a noticeable difference from Year 1. We will look more closely at the result of these activities on student engagement through two representative cases from each year.

Year 1 paint a square

Our case in Year 1 focuses on Brianna, age 12, as she learns the connections between her model and the coordinate plane toward the beginning of her work with Paint a Square. At the start of the activity, she overheard a teacher saying that squares are equal on all sides. This led Brianna to change all of the inequality symbols in the code to equal signs and to make four statements in the code saying pxcor = 150, creating a thin vertical white line at x=150. She becomes frustrated and says "Oh, my gosh. I need help." A teacher (T) comes over to the table to check in. In this episode Brianna demonstrates confidence in her ability to use NetLogo as she adds new code and is also attempting to incorporate the mathematical properties of a square into her model.

Code:
```
ask patches with [pxcor = 150 and pxcor = 150 and pxcor = 150 and pxcor = 150] [set pcolor white]
```

1 **T:** How's it going at this table?
2 **Brianna:** Terrible.
3 **T:** Terrible?! You want to talk about it?
4 **Brianna:** Yeah.
5 **T:** Okay, what are you looking at?
6 **Brianna:** I'm trying to turn this into a square but it ain't working.
7 **T:** Okay. What's going on?
8 **Brianna:** I added two more pxcors because there are, because they're supposed to have four equal sides.

Brianna is attempting to utilize her knowledge about both mathematics and NetLogo, as her code is reasonable and her ideas about a square are sound. However, she doesn't yet seem to know how these mathematical properties can be communicated in her NetLogo model; it appears that she believes that "pxcor" relates to the length of the side of the square instead of the x-coordinate of a patch in the model. While it is true that she was attempting to connect mathematics with the NetLogo syntax, this connection was superficial and did not engage with the overlapping content between the code and coordinate grid. To truly hybridize the two, an adult needed to serve as the "boundary object" in the interaction. In this second episode that immediately follows, the teacher reads Brianna's code and assesses her knowledge of this code's connection to the coordinate plane. Based on Brianna's response, the teacher proceeds to teach Brianna about the connections between the code that she has typed, the coordinate plane in the model, and the resulting white stipe that she sees appearing when she runs her model.

11 **T:** *(continues reading code)* "and pxcor = 150 and set pcolor white." Okay, so just this, when it says, "pxcor = 150," where are all the patches with pxcor = 150?
12 **Brianna:** I don't know.
13 **T:** Well, what do they mean? Let's look. Let's hit 'Okay'. (*Brianna runs model)* And if you - oh, no. What happened when you clicked that button?
14 **Brianna:** Just made a skinny line.
15 **T:** It made a skinny line [chuckle]. That's not what you wanted, was it?

16 **Brianna:** No.
17 **T:** No. Awesome. Okay, so where, if I'm looking on the XY plane, where are my X coordinates? Are they horizontal or vertical?
18 **Brianna:** Horizontal.
19 **T:** Horizontal. Awesome. Okay. Where do you think maybe the X coordinate 150 is?
20 **Brianna:** Somewhere over here. (*Gestures to the screen)*
21 **T:** Somewhere over here. Okay. What did you ask the patches to do? Yeah, open that up. (*Right clicks on 'make white square in center' button and opens the edit code tab)* When you say, "Ask patches with pxcor = 150, turn your color white," what are the patches doing? Which ones are turning white?

In line 17 the teacher makes a direct connection to the coordinate plane, and Brianna responds correctly to inquiries about the x-axis (line 18) and the general layout of the coordinate plane in her model (line 20). The teacher then proceeds to help Brianna cross the boundary between mathematics and NetLogo by relating her code "pxcor =150" to the patches on the coordinate plane that she sees turn white in her model (line 21). This line of inquiry and instruction continues between Brianna and the teacher who works one-on-one to connect the NetLogo model and the code used to communicate with it to mathematical concepts needed to solve the problem. Throughout, Brianna communicates to the teacher what she wants to do mathematically in the model and the teacher helps Brianna to understand how to communicate these mathematical concepts to NetLogo. Together, they finalize a model that makes a big white square in the middle of the model.

Brianna's interactions with the teacher were indicative of many students' engagement with Paint a Square in Year 1. It often seemed that they knew how to use the mathematical concepts needed to solve this model, but they are unable to bridge the gap of communicating the mathematics with NetLogo. Therefore, while mathematics and computation are used frequently and concurrently in the debugging models, it is almost always prompted by the mathematical ideas and their connections to the NetLogo code in one-on-one interactions with a teacher. As Akkerman & Bakker (2011) state, "individuals crossing boundaries show how they not only act as bridge between worlds but also simultaneously represent the very division of related worlds" (p. 140). This seems to be the case in examples from Year 1; the boundaries between mathematics and programming remain very clear, and requires the engagement of a broker, in the form of a teacher, to help bring those worlds into contact and conversation. Akkerman & Bakker (2011) characterize this kind of boundary learning as *identification,* which involves "…a process in which previous lines of demarcation between practices are uncertain or destabilized because of …increasing similarities or overlap between practices. The reported processes of identification entail a questioning of the core identity of each of the intersecting sites" (p. 142).

Year 2 paint a square

Our case in Year 2 focuses on Caleb, age 12, during his first attempt on his debugging model. He begins by changing "pxcor < 5" to "pycor < 5" in the original code which changes all the patches in the bottom right corner of the model white. This was a common first attempt in both Year 1 and Year 2. A teacher looks over Caleb's shoulder as he runs his model and engages him in conversation. During the following interaction, Caleb uses some of the school-math vocabulary that was incorporated into the camp by students the day before and uses the concepts from that discussion to create a unique potential solution.

Code 1:
```
ask patches with [pxcor > -5 and pycor < 5] [set pcolor white]
```

1 **Caleb:** Huh
2 **T:** Oh. You got a square but it's just not in the middle
3 **Caleb:** Oh, I see.
4 **T:** He had a square, but it wasn't in the middle. *(directed at another teacher)*
5 **Caleb:** Oh, I know what to do! I know what to do! I know what to do. *(rubs hands together with excitement)*. So, you do, um, you do, you do a square in the first quadrant, second quadrant, third quadrant, fourth quadrant. *(motions in the air to four corners of an imaginary box)* =
6 **T:** = I mean. Okay! Okay! Okay! =
7 **Caleb:** = And it's a square. *(Copy and pastes first line of code three more times and changes each of the inequalities to be either less than or greater than 1 or -1. Runs code to get a smaller white square in the bottom right of the model.)*

Code 2:
```
ask patches with [pxcor > -1 and pycor < 1] [set pcolor white]
ask patches with [pxcor > 1 and pycor < -1] [set pcolor white]
ask patches with [pxcor > -1 and pycor < -1] [set pcolor white]
ask patches with [pxcor > 1 and pycor < 1] [set pcolor white]
```

8 **Caleb:** Wait, what?!

Despite this confusion when getting a square in the corner of his interface when he ran his code (line 1), Caleb quickly assesses this feedback (line 3) and excitedly begins a new strategy, to make one square in each quadrant (line 5) which would come together to make one big square in the middle. His use of the uniquely school-math term "quadrant" as it relates to the coordinate plane in his explanation was common in Year 2 as it referenced the whole group discussion and brainstorming activity that involved identifying the four quadrants of the coordinate plane. With this idea of using quadrants in mind, Caleb replaces all of the numbers in the model to four combinations of 1 and -1, representing the four coordinates closest to the origin in the (line 7 and Code 2). The result is unexpected (line 8). After making a few unsuccessful edits (not shown here but named Code 3 and Code 4) he returns to Code 2 and proceeds with his potential solution.

14 **Caleb:** Oh, I could just delete this. *(deletes all inequalities)*
15 **T:** But you've got to have some kind of inequality!
16 **Caleb:** I could just do that. *(Replaces all inequalities with equal signs. Runs code. Gets error due to 1 missing equal sign)* Oh and right here *(types in equal sign)*. Setup. Make Square *(Runs code. Four patches turn white surrounding the origin of the coordinate plane)*

Code 5:
```
ask patches with [pxcor = -1 and pycor = 1] [set pcolor white]
ask patches with [pxcor = 1 and pycor = -1] [set pcolor white]
ask patches with [pxcor = -1 and pycor = -1] [set pcolor white]
ask patches with [pxcor = 1 and pycor = 1] [set pcolor white]
```
17 **Caleb:** *Smiles and looks around to people in his vicinity.*
18 **T:** Ohhhhhh! Okay! Okay! We got 4 squares! *(smiling)* They are -
19 **Caleb:** They are in the middle! *(turns computer to show the people around him)*
20 **T:** You need one square though. That IS kind of awesome.

While, Caleb's code didn't initially work because he wasn't attending to the inequalities in the code, it was apparent from his conversation with the teacher (line 5) that he was attempting to paint the patches located at (-1,1), (1,-1), (-1,-1) and (1,1) white. This is confirmed when he replaces the inequalities with equal signs in Code 5. With his code, Caleb gets 4 small squares that are separated by the points on the x-axis and y-axis of his model which remain the background color of the model. While this isn't what the "client" in the model asked for, both Caleb and the teachers in the room are impressed and proud of this move toward a solution (lines 17-20). This episode demonstrates Caleb's use of relevant mathematical concepts based on the whole group discussion about the coordinate plane that preceded the debugging activity, but also his ability to concurrently use those mathematical concepts with the NetLogo code to manipulate the objects that make up the coordinate plane in the model to authoritatively produce an interesting and unique solution alongside but without direct instruction from a teacher.

Caleb demonstrates behavior and interactions with teachers that were typical of his fellow campers in Year 2. Notably, in Year 2, while students still interacted with teachers regularly, the mathematical ideas and the ways to communicate those ideas to NetLogo were largely initiated and pursued by the students. As in Caleb's case, students often even ignored or corrected teacher's advice or suggestions, but they did seek assistance when needed to develop new syntax vocabulary or mathematical connections when models didn't work as expected. Students in Year 2 appeared to demonstrate more authority in their problem solving as a result of having tools necessary to utilize the mathematical content that they knew alongside the computational skills they learned about NetLogo during the camp. This led to more unique solutions to problems and increased ability to develop expressive designs through code.

Conclusion and discussion

In this paper we have focused our analysis on two cases which are examples of the kinds of interactions that we routinely observed in each year of the camp. However, we are mindful that single contrasting case studies are not intended to offer generalizations, and thus we offer a set of conjectures that we feel come out of this analysis. While there are many factors that contributed to differences between students' experiences in the two years of the Code Your Art camps, the representative cases of hybrid math and computational reasoning that we have explored demonstrates that the design of boundary crossing activities was crucial to students' engagement in this hybrid learning activity. These bridging activities are ones which enabled students to explore and make connections between multiple disciples. While students in Year 1 successfully completed the Paint a Square activity and did so with high levels of engagement, their success required an expert teacher to broker the connections between mathematics and the computer model in one-on-one interactions, much like we saw in Brianna's episode. This means that students in Year 1 were constrained in exercising their own agency in exploring different solutions to the debugging problems. Students in both years were certainly productive and knowledgeable of both mathematics and computation, but they demonstrated differences in their agency, problem solving, and expressive designs in the context of debugging. This indicates that the ability to utilize mathematics and NetLogo concurrently is significant to authoritative problem solving in this debugging model. As seen in Year 1, it wasn't enough for students to know the mathematics and to know NetLogo, they had to shift into using both of these disciplines concurrently, which was made more likely to occur independently in Year 2 as a result the implementation of collective boundary crossing activities. Students' engagement with these activities, which elicited relevant mathematical and computational knowledge and skills, demonstrated how both disciplines were connected and could be used concurrently and fluidly to pursue and find unique and interesting solutions to this debugging model.

In order for students to productively engage in learning spaces that require engaging multiple disciplines to solve problems, the designers of those spaces could attune to the areas of intersection between the two disciplines. Preparing students to bridge disciplines requires that the design elicit and demonstrate necessary ideas from each discipline but also demonstrate how these disciplines are connected. As in the Year 2 iteration of the camp outlined in this study, this could come in the form of designing bridge activities within which students can learn to engage in a new form of hybrid participation, one that is neither strictly mathematics nor strictly computer programming. This could offer the potential to develop authority in their problem solving that leads to unique solutions to open problems as the result of the resources available to students based on design. As we see technology and the computer sciences expand into nearly every domain of our global community, the implications for this kind of participation in multidisciplinary learning and problem solving with computers across various disciplines is far reaching and relevant now more than ever.

References

Akkerman, S. F., & Bakker, A. (2011). Boundary crossing and boundary objects. Review of educational research, 81(2), 132-169.
Engle, R. A. (2006). Framing interactions to foster generative learning: A situative explanation of transfer in a community of learners classroom. The Journal of the Learning Sciences, 15(4), 451-498.
Gadanidis, G., Hughes, J. M., Minniti, L., & White, B. J. (2017). Computational thinking, grade 1 students and the binomial theorem. Digital Experiences in Mathematics Education, 3(2), 77-96.
Gresalfi, M., Brady, C., Knowe, M., & Steinberg, S. (2020). Engaging in a New Practice: What Are Students Doing When They Are "Doing" Debugging?.
Harel, I., & Papert, S. (1990). Software design as a learning environment. Interactive learning environments, 1(1), 1-32.
Kafai, Y. B. (2016). From computational thinking to computational participation in K--12 education. Communications of the ACM, 59(8), 26-27.
Nasir, N. I. S., & Hand, V. (2008). From the court to the classroom: Opportunities for engagement, learning, and identity in basketball and classroom mathematics. The Journal of the Learning Sciences, 17(2), 143-179.
Pérez, A. (2018). A framework for computational thinking dispositions in mathematics education. Journal for Research in Mathematics Education, 49(4), 424-461.
Redish, E. F., & Kuo, E. (2015). Language of physics, language of math: Disciplinary culture and dynamic epistemology. Science & Education, 24(5-6), 561-590.
Sandoval, W. (2014). Conjecture mapping: An approach to systematic educational design research. Journal of the learning sciences, 23(1), 18-36.
Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. Education and Information Technologies, 22(2), 469-495.
Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127-147.
Wilensky, U. (1999). Center for connected learning and computer-based modeling. In NetLogo. Northwestern University.
Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. TechTrends, 60(6), 565-568.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1742257. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We are very grateful to Dr. Corey Brady, Amanda Bell, Lauren Vogelstein, Selena Steinberg, and Candice Love for their feedback on this manuscript.