Chapter 10
Playfully Coding Science: Views from Preservice Science Teacher Education

Pratim Sengupta, Beaumie Kim, and Marie-Claire Shanahan

Abstract There is now a growing body of research focused on integrating computational thinking and modeling in teacher education, ranging from studies that investigate preservice teachers' perceptions of computational thinking to those that evaluate the efficacy of computational tools that can support such integration. Our work extends this literature by investigating how preservice science teachers can be introduced to computational thinking and modeling by playfully designing computer simulations and games for modeling kinematics and ecological interdependence. Adopting a phenomenological research agenda, we focus on how preservice science teachers experience coding and computational modeling as pedagogical experiences for science education. In doing so, our goal is to contribute to an epistemological, rather than an instrumental, understanding of computational thinking and modeling in the context of preservice science teacher education.

Keywords Computational thinking · Play · Phenomenology · Science education · Modeling · Teacher education

10.1 In.troduction

Over the past several years, computational thinking (Wing, 2006) has emerged as one of the centerpieces in K-12 STEM education. Computational thinking has been commonly positioned by scholars as involving analytical skills that draws on concepts and practices from computer science, as well as a more fundamental ability that can be used by and useful for all people (Wing, 2006). Computational thinking in Wing's (2006) words involves "… breaking down a difficult problem into more familiar ones

P. Sengupta (✉) · B. Kim · M.-C. Shanahan
Werklund School of Education, University of Calgary, Calgary, AB, Canada
e-mail: Pratim.sengupta@ucalgary.ca

P. Sengupta et al. (eds.), *Critical, Transdisciplinary and Embodied Approaches in STEM Education*, Advances in STEM Education,
https://doi.org/10.1007/978-3-030-29489-2_10

that we can solve (problem decomposition), using a set of rules to find solutions (algorithms), and using abstractions to generalize those solutions to similar problems" (p. 33). In the context of science and STEM education, computational thinking must be thought of contextually in light of disciplinary practices, which involves developing epistemic and representational practices such as thinking algorithmically, use of data structures and other relevant forms of computational abstractions, and designing and creating computational artifacts such as programs and simulations (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013; Weintrop et al., 2016).

There is now a growing body of research focused on integrating computational thinking and modeling in teacher education, ranging from studies that investigate preservice teachers' perceptions of computational thinking (Bower & Falkner, 2015; Sands, Yadav, & Good, 2018) to those that evaluate the efficacy of computational tools that can support such integration (Kalogiannakis & Papadakis, 2017). Our work seeks to extend this body of work by investigating how preservice science teachers can be introduced to computational thinking and modeling by playfully designing computer simulations and games for modeling kinematics and ecological interdependence.

However, following Sengupta, Dickes & Farris (2018), rather than adopting a technocentric (Papert, 1987) approach, where the primary (and often the sole) emphasis remains on evaluating computational artifacts generated by participants to assess how they have used and applied computational abstractions, our approach is phenomenological in nature. In a phenomenological approach participants' *sense experience* becomes objects of inquiry (Sengupta, Dickes & Farris, 2018). Merleau-Ponty (1962) defined sense experience as a dynamic and dialectical form of experience: "that vital communication with the world which makes it present as a familiar setting of our life" (Merleau-Ponty, 1962, p. 61). The emphasis on understanding participants' sense experience must necessarily go beyond the sphere of givenness—i.e., the world as it is already known—and reveal participants' originary sense-making (Merleau-Ponty, 1962; McMahon, 2017). In the present context, this means that our focus is not on assessing the efficacy of our pedagogical approach in terms of helping preservice teachers apply computational abstractions re-contextualized as computational models of scientific phenomena. Instead, we are interested in preservice science teachers' originary sense-making of coding and computational modeling as pedagogical experiences for doing and learning science. In doing so, our goal is to contribute to an epistemological, rather than an instrumental, understanding of computational thinking and modeling in the context of preservice science teacher education.

10.2 Research Question

Specifically, we ask the following research question: how do preservice science teachers view computing and coding as pedagogical experiences in the context of doing science through playfully engaging in computational modeling and game design?

10.3 Background

10.3.1 Productive Uncertainty and Play in Science Education

Studies of scientists at work reveal an image that is far from being one of certitude, despite the latter being the more commonly represented image of science in public education (Duschl, 2008). For example, Pickering (1995) illustrated how scientific advancement necessitates a deep entanglement of theories and materiality, and of conceptual and representational work, thereby rendering a far more nuanced character than what is commonly represented in the public imagination. Ochs, Gonzales, and Jacoby (1996) highlighted the central role of interpretive work in creating scientific knowledge, and illustrated how this interpretive uncertainty is also tied to the representational infrastructure. This is echoed by Daston and Galison (2007), who pointed out that as representational technologies evolve and new ones emerge in order to support scientific advancement, their use, on the other hand, often results in new forms of uncertainty and interpretive work.

There is a growing recognition among science educators that an essential aspect of the teachers' work is developing a more nuanced view of scientific uncertainty and supporting students in such nuanced scientific inquiries. Aikenhead (2003) identified that grappling with the feeling of "playing in the subculture of science" both as insider and outsider is essential to humanistic pedagogies. Manz and Suárez (2018) proposed some strategies that can promote such pedagogies, such as beginning with complex phenomena, iterating on investigations, and leveraging variability in students' ways of conducting investigations. Similarly, Farris, Dickes, and Sengupta (2019) argued that when teachers pay attention to students' errors and uncertainties during the process of designing computational models in science classrooms, they can support students to deepen their engagement with scientific practices.

We believe that positioning computational modeling as playful engagement with science and computing can also support preservice teachers' engagement in such experiences that value, rather than ignore, interpretive uncertainties. Playful learning environments can greatly facilitate learning of complex topics across a range of STEM disciplines (Berland & Lee, 2011; Sengupta, Krinks, & Clark, 2015; Kim & Ho, 2018; Sengupta & Shanahan, 2017). The notion of play challenges the expectations of disciplinary rigidity that often keep newcomers from participating in scientific inquiry (Sengupta & Shanahan, 2017). Playful engagement with virtual learning environments can help learners reshape the learning activities even within a structured setting, such that the activities are both personally meaningful and relevant to the disciplinary context of learning (Farris & Sengupta, 2016; Kim & Ho, 2018). As Kim and Ho (2018) and Sengupta and Shanahan (2017) pointed out, central to positioning personal meaningfulness alongside disciplinary relevance is the harnessing, rather than discarding, of possibilities that emerge from interpretive flexibilities and uncertainties. Participants' dilemmas and uncertainties, we therefore believe, can become resources in playful engagement with STEM disciplines.

10.3.2 Computational Thinking and Modeling in Teacher Education

Several scholars have argued for democratizing computation by integrating computing with other disciplines and existing courses (such as science and math) that all children are required to take, rather than trying to create room for computer science as a new curricular domain (Sengupta et al., 2015; Wilensky, Brady, & Horn, 2014). We posit that the same arguments must be extended for preservice teachers, especially given that many of them may not have prior experience in computational modeling and programming. Rather than learning computer programming as a separate discipline, as Yadav, Stephenson and Hong (2017) also argued, we believe that in their science-methods courses, preservice teachers could be introduced to computational thinking through computational models. Such experiences, we believe, can help them deepen their (future) students' engagement with conceptual and representational practices that are central to the development of both scientific and computational expertise in a reflexive manner (Sengupta et al., 2013).

A growing body of literature advocates engaging preservice and in-service science teachers in computational thinking and modeling through positioning them as creators of computational models and artifacts. For example, Wilkerson et al. (2016) found that when preservice teachers are provided opportunities for constructing simulations in science, they are able to engage in practices that are central to scientific modeling, such as model evaluation and revision, in ways that are deeply connected to key conceptual ideas relevant to the phenomenon being modeled. Leonard et al. (2018) also found that culturally and contextually embedded game design activities and robotics can support teachers in developing dispositions central to computational thinking. Our work seeks to contribute to this literature by offering insights into how preservice teachers frame (and re-frame) code and coding from a pedagogical perspective in the context of scientific modeling, as part of their teacher preparation coursework.

10.4 Our Pedagogical Approach: Integrating Playfulness and Mathematization to Support Preservice Science Teachers' Computational Work

Our work is premised on the position that engaging preservice teachers in computational modeling can be supported by emphasizing playfulness in their interactions with computational artifacts. This is particularly important given that many preservice science teachers may not have prior experience with programming (e.g., Yadav et al., 2017). Positioning computational and scientific work as play would allow teachers, regardless of their prior background, to interact with computing in their *regimes of competence* (diSessa, 2001). diSessa (2001) reminded us "that resources for learning don't always look just like the product of learning" (p. 84) from the

perspective afforded by the regime of competence. This means that learners (broadly speaking, and including preservice teachers in this case) may begin from a place that may not be initially recognizable as the putative discipline to be learnt. Instead, the regime of competence—e.g., practices and hobbies that the learners may already be interested in outside the discipline—may offer learners productive resources, using which they can develop a deep and meaningful relationship with the discipline (Azevedo, 2018).

Our previous work with in-service science teachers offers some insights into what such regimes of competence might look like (for science teachers). For example, we found that framing programming as "mathematizing" in the science classroom can serve as a productive pedagogical approach for integrating programming in the K-12 science classroom (Sengupta et al., 2013, 2015; Sengupta, Brown, Rushton, & Shanahan, 2018; Dickes, Sengupta, Farris, & Basu, 2016; Farris, Dickes & Sengupta, 2019). In this approach, programming is used in the context of creating computational models of scientific phenomena through designing discrete mathematical representations of units of change, for representing change over time. Similarly, Sands et al. (2018) found that all teachers in their study—both primary and secondary—viewed mathematical work in the classroom as a form of computational thinking. This is also not surprising given the heavy emphasis on mathematical work within K-12 science curricula, as we found in our work with K-12 teachers both in the USA and Canada (Sengupta, Brown, Rushton, & Shanahan, 2018; Farris, Dickes & Sengupta, 2019).

Such a reframing of computing as *mathematizing* is in line with a phenomenological approach in which computational thinking is viewed not as merely a set of prescribed performances of technological and symbolic dexterity, rather as more complex and heterogeneous forms of experience (Sengupta, Dickes & Farris, 2018). While at first glance, it might seem counterintuitive to claim that an emphasis on mathematizing may position teachers and preservice teachers in regimes of competence, Farris, Dickes and Sengupta (2019) found that when teachers and students amplify (rather than ignore) the interpretive dilemmas and uncertainties involved in (and inherent in) mathematizing code for designing scientific models, their work can become progressively more creative and at the same time, computationally more intensive.

With this in mind, we designed a set of learning activities in which preservice teachers were presented with two computational simulations designed in the NetLogo Web platform: Lunar Lander (Fig. 10.1), and Bird-Butterfly-Flower Ecosystem (Fig. 10.2). In both the simulations, the activities involved not only manipulating parameters and variables that controlled the simulation, but also modifying the underlying NetLogo code. Modeling in NetLogo involves instantiating the individual elements of a system - the "agents" - and simulating their interactions using NetLogo code. A particular affordance of NetLogo code is that it has been shown to be effective in supporting science learners and newcomers to computing engage deeply with computational modeling, by drawing upon their intuitive and embodied knowledge (Wilensky & Reisman, 2006). In addition, agent-based modeling and programming is fundamentally aligned with mathematization because the

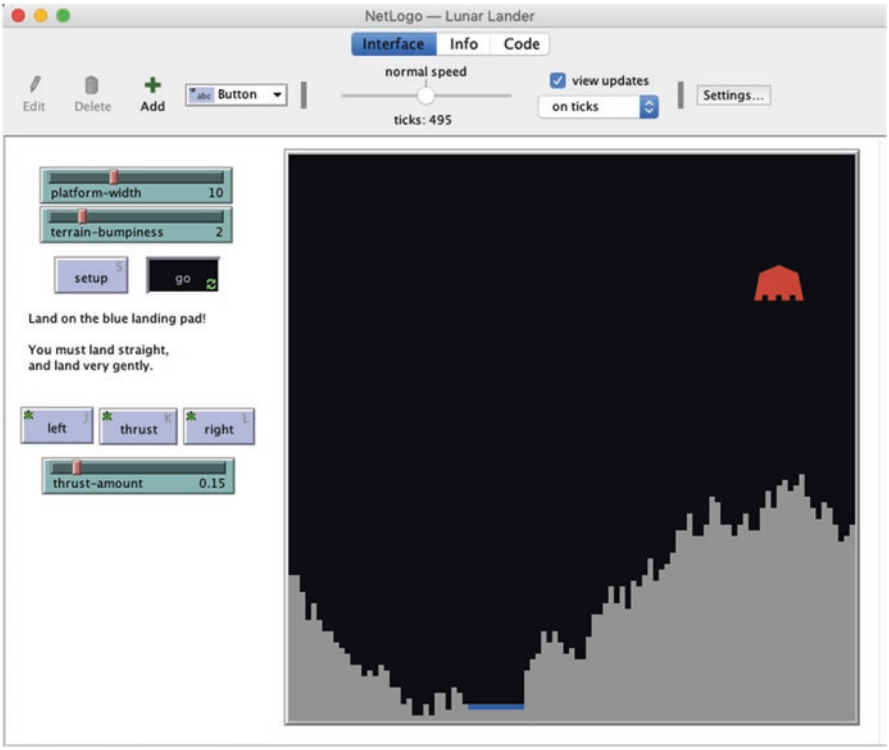

Fig. 10.1 A screenshot of the Lunar Lander NetLogo simulation (Game)

activity of programming the behavior of agents requires the learners to define an event using discrete mathematical measures (Sengupta et al. 2015).

In the Lunar Lander simulation, the goal of the "player" was to land the spaceship safely on the lunar surface. This meant that the ship had to land at very low speed, and vertically. Controlling the trajectory of the ship involved adding thrust (sudden bursts of acceleration) along any of the four directions: top, down, left, or right. Based on the classic premise of early moon landing games (e.g., Atari's 1979 Arcade Game *Lunar Lander*), we used a modified version of the Lunar Lander simulation in the NetLogo Models Library (Wilensky, 1999). The NetLogo simulation is designed as a game and we modified it such that it would be very difficult for players to land successfully. We did so by modifying the underlying NetLogo code so that the length of the landing strip on the lunar surface was nearly equal to the width of the ship. That is, in trying to land, the ship would inevitably crash due to hitting rocks adjacent to the landing strip. The framing of the simulation as a game and their activity as game hacking, we posited, would encourage our participants to dig deeper into the code.

The Bird-Butterfly-Flower simulation (Dickes & Sengupta, 2013) modeled predator–prey dynamics in an ecosystem of flowers, butterflies, and birds. The overarch-

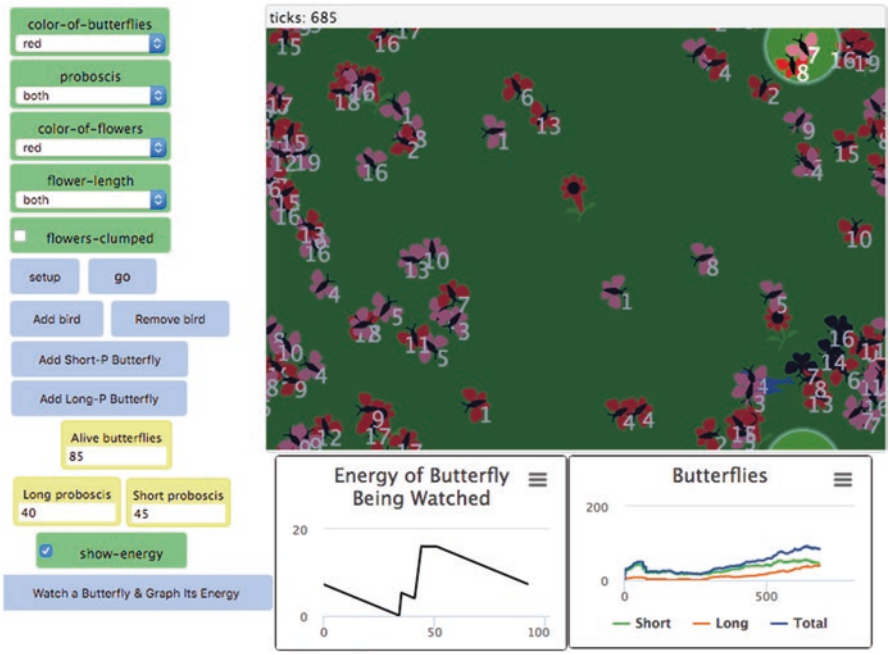

Fig. 10.2 A screenshot of the Bird-Butterfly-Flower simulation

ing goal of this activity was to identify the model parameters that resulted in a thriving butterfly population, manipulating variables such as proboscis length, flower length, flower location, color of flower, color of butterfly, and speed of predator movement. In addition, similar to the Lunar Lander simulation, participants also had the opportunity to significantly expand the scope of the simulation by introducing additional predators, altering structure–function balance by changing the morphological characteristics of the agents, etc.

In order to encourage and scaffold preservice teachers to "hack" and "debug" the NetLogo code, we commented the code heavily and provided a printed programming guide (see Figs. 10.3 and 10.4). The comments within the NetLogo code explained how each line of code affected the simulation. The printed programming guide explained how relevant code segments represented the science concepts and mathematical relationships, as well as how to change relevant code segments in order to change different elements of the simulation (e.g., changing the gravity on the moon, the size of the landing strip, acceleration and deceleration due to the thrusters). In class, we also encouraged the students to dig into the code as part of the "playful" experience and answered their questions to help them along their journey.

Change Acceleration based on Y coordinate

```
92      [ setxy (xcor + xvel) (ycor + yvel) set size 1 stamp set size 10]
93      ;; exert the force of gravity
94      set yvel yvel - 0.001
95      ;; detect crashes and insufficiently soft landings
96      if [pcolor] of patch-at 0 -2 != black ;or pcolor = yellow
```

The goal here is to tie the ship's speed or acceleration to its y coordinates.

The line of code that we are looking for here is:

set yvel yvel - 0.001

This can be changed to:

set yvel (yvel - 0.001) - (ycor * 0.0001)

What does this change do? Every step, the new y-velocity, which is specified by the "variable" **y-vel**, is equal to the old y-velocity minus the y coordinate multiplied by 0.0001. This number was based on playing around with the code; it is quite likely that you may want to try out other options for better outcomes.

Fig. 10.3 Excerpt from the Lunar Lander Activity Guide

How Do Butterflies Avoid Birds?

Here's the relevant code:

```
to avoid-birds
 if count birds in-radius 3 > 0
 [
 set heading towards one-of birds
 rt 180
 fd 0.4
]
end
```

Each butterfly looks for birds within a radius of 3 units.

If they see a bird, they first face the bird (set heading towards one-of birds), then turn away from the birds (rt 180), and them move away from them by a distance of 0.4 units (fd 0,4).

You can change the angle of turning and/or the distance by which the butterfly moves away from the bird. What do you think will happen as a result of your change to the overall population?

Fig. 10.4 Excerpt from the Bird Butterfly Flower Simulation Activity Guide

10.5 Method

10.5.1 Setting and Participants

We conducted our study in a secondary science education methods course in a Canadian research-intensive university. The third author of the paper was the instructor of the course, and all three authors collaboratively designed the computational artifacts and programming guides used in the study. All three authors were present during Day 1, whereas the first two authors were present during Day 2. There were 27 students in the class, a majority of whom participated in two intensive sessions on computational modeling. All the students were enrolled in a two-year, after-degree B.Ed program with a specialization in secondary science. All students held a Bachelors or Masters degree (or beyond) in a scientific or engineering discipline, and several of them had professional careers in their previous specializations.

10.5.2 Data and Analysis

The data for this study comes from two days of classroom activities, totalling to approximately 6 h of class time. During Day 1, participants worked on the Lunar Lander simulation and game, while on Day 2 they worked on Bird-Butterfly-Flower simulation. During each day, the participants were introduced to key elements of the underlying code of the relevant simulation or game by one of the researchers. Participants worked in groups of two or three throughout the duration of the study. The researchers also visited each group to work with them as needed both on their computational modeling and their pedagogical focus. They also led class discussions in which participants shared reflections on their experiences of the classroom activities.

Overall, the activities were framed as playful pedagogical exercises. That is, the participants were first asked to take on the perspective of their future students by engaging with the computational models and games following the programming and activity guide. They were specifically encouraged to discuss with their partners the challenges and successes that would emerge in their interactions. They were then asked to redesign the simulation or game in order to deepen their students' engagement and/or to address the challenges they experienced initially.

We collected three forms of data: (a) students' written memos on their experiences relevant to the course, (b) computational artifacts designed by students, (c) video and audio recordings of classroom conversations and interviews with the participants. These interviews were conducted while the researchers were working with the participants in small groups. During the interviews, we asked the participants to explain their challenges and how they were planning on addressing them, as well as how and why they would further modify the underlying NetLogo code.

We conducted thematic analysis using the check coding method (Miles & Huberman, 1994) to analyze the interview data, classroom conversations, and student artifacts. The class discussions provided us with initial insights into emergent *themes*, i.e., views that were shared by several groups of participants. We then analyzed interviews with specific students and small groups in order to develop a more detailed understanding of the participants' experiences referred to during the class discussions. We iteratively compared our emergent observations as evident from both these data sets, and identified three salient themes that were experienced broadly by the class. All the authors collaboratively discussed and identified these themes.

10.6 Findings

Our analysis identified three themes along which preservice teachers' viewed code and coding as pedagogical objects and experiences for deepening scientific inquiry.

10.6.1 Theme 1: Interacting with Code Can Deepen Conceptual Engagement in Science

In order to understand preservice teachers' understandings of how coding can support scientific inquiry, in the first illustrative case, we focus primarily on Adela and Jerry, who participated in all the modeling activities reported in this chapter. Adela has a Master's degree in biology, and Jerry had recently completed a doctorate in astrophysics. Neither of them identified themselves as coders or had taken any computer science course, although both of them had some prior experience with programming. In the excerpt below, we illustrate how Adela's and Jerry's experiences with playful coding shaped how they saw coding in their (future) science classrooms as a pedagogical tool.

In Excerpt 1, Adela and Jerry are explaining to Pratim one of the changes they were thinking of making to the NetLogo code in order to make it easier to land the spaceship. They were discussing the possibility of changing the code so that as the ship used the thruster, it would also become lighter. Jerry immediately recognized this as senior level undergraduate physics, because this uses the notion of differential mass. Adela, on the other hand, was not convinced that it would be difficult for students to understand the idea, because for her, the notion of differential mass, in this context, involved "the concept of burning something a.. and losing mass with it" (Turn 3). Jerry argued that the students would still not be familiar with the formal mathematics involved, but Adela argued that "as a teacher and you would provide them the code you wouldn't expect them to come up with it but you could have them to explore the results of it" (Turn 6).

Excerpt 1: Lunar Lander

Turn 1: Adela**:** What we were thinking of is the laws of thermodynamics, so conservation of mass, so as you're using your thruster your ship will get lighter

Turn 2: Jerry**:** That's a very crazy 4 year physics, with differential mass, yeah I think that's well beyond

Turn 3: Adela**:** The concept of burning something a.. and losing mass with it?

Turn 4: Jerry**:** Yeah you can talk about it but mathematically speaking they will not have [the understanding]

Turn 5: Pratim: Yeah but I think what she's saying is that this would actually make that understandable

Turn 6: Adela**:** Right so as a teacher and you would provide them the code, you wouldn't expect them to come up with it but you could have them to explore the results of it

Turn 7: Dorothy**):** If you set up the code then they can manipulate it

Turn 8: Adela**:** Yeah yeah exactly

This excerpt is insightful in two senses. First, Adela's explanation here is indicative of the framing of code itself as a pedagogical object—i.e., as an object that can be manipulated by the students for deepening their conceptual engagement with science. This is also supported by another preservice teacher in the class (Dorothy), who agreed that code can be "set up" in such a way so that it can be manipulated by the students (Turn 7). Second, as Pratim interpreted and re-articulated Adela's comments (Turn 5), another important implication is that through interacting with the code, secondary students, by using their intuitive understanding of "burning something" and "losing mass," can begin to explore conceptual issues that are typically reserved for upper undergraduate physics.

10.6.2 Theme 2: Coding for Scaffolding as a Form of Productive Uncertainty

The idea of "setting up" the code, as Dorothy put it in Excerpt 1 (Turn 7)—i.e., designing code in order to pedagogically support particular forms of student interactions—deserves further unpacking, because it can bring to light a form of productive uncertainty experienced by the participants during their own playful engagement with the simulation and the code. We found that several participants realized the need to modify the code not only for deepening students' conceptual engagement with science, but also to scaffold their experience of play. In the process, they experienced dilemmas regarding whether scaffolding their students would help or limit their scientific inquiry. We see this dilemma as a form of productive uncertainty that in turn deepened their own understanding of the relationship between coding and pedagogical design in the science classroom. The following excerpt (Excerpt 2), which reports a conversation between Marie-Claire and two participants, Ronnie and Negin, provides a rich illustration:

Excerpt 2

Turn 1: Ronnie: So like if it tell you the cause of death then you can adjust to your play

Turn 2: Marie-Claire: Right, so if it tells you more specific feedback

Turn 3: Ronnie: Right exactly - it took us a few tries to realize that we were going too fast

Turn 4: Marie-Claire: So it wasn't just about position whereas it might be different for someone else who is slightly off position

Turn 5: Ronnie: Yes exactly

Turn 6: Marie-Claire: Cool, yeah that makes sense -Did you manage to land?

Turn 7: Negin: Once

Turn 8: Marie-Claire: So if you wanted to change it to add feedback- would you give categories of feedback like - "you died due to speed, You died due to" and sort of have

Turn 9: Ronnie: Yeah, what is the alternative?

Turn 10: Marie-Claire: I'm not sure. How would you envision that feedback?

Turn 11: Ronnie: Directly that you died due to the velocity, was too fast, the speed was too fast

Turn 12: Negin: That's very straightforward though- you could maybe show this guy falling, and he falls off the track you could show him falling off the track and rolling over or blowing up if it was too fast.

Turn 13: Negin: Or maybe some warning

Turn 14: Marie-Claire: -Beep beep beep beep

Turn 15: Ronnie: Yeah some warning that you are going to fast - so before you die you have the chance to save yourself

Turn 16: Marie-Claire: Or abort mission - eject eject - Haha - but I assume you do know what you want to change

Turn 17: Ronnie: Yep, here it was very vague - I was like what are we doing, why did we die?

Turn 18: Negin: Yep and then I increased the platform here, and I still died, so I was like there is something going on here

Turn 19: Marie-Claire: Right - so it's not alignment, and it's not

Turn 20: Ronnie: But there is a good thing in not giving any feedback for learning because then it makes it more inquiry - when we did figure it out that it was due to velocity - there is a good aspect of not having any warning

Turn 21: Marie-Claire: Yeah - I was going to ask that - do you think there is something - is there something for students to learn in that process of figuring out why - why did I die

Turn 22: Ronnie: Yeah, because we eventually figured it out - because it was too fast right because we were concerned because it landed right perfectly on the blue line - so we eliminated that

Turn 23: Marie-Claire: Other elements

Turn 24: Ronnie: Yeah, so it's not totally bad that there is no - but at the same time I don't know….

Turn 25: Marie-Claire: I wonder if it depends on what you want them to get out of it

Turn 26: Ronnie: Yeah - maybe you can have the option you can play it without any feedback - and first play it like that and figure out why you died - and then you can switch it to a different mode

At the beginning of the excerpt, the participants explained to Marie-Claire that they were thinking about changing the code so that the player (or student) would get a visual feedback that would specify and explain the cause of the crash of the spaceship in the Lunar Lander simulation. Ronnie mentioned that it took her and Negin (her partner) a few attempts to figure out that they were crashing because the speed of the spaceship was too high (Turn 3). Negin explained that they arrived at higher speed as being the cause of the crash only after they made alterations to the code to "increase the landing"—i.e., to flatten the landing surface (see Fig. 10.5a, b), so that it would be easier for the ship to land, and then they realized that doing so still did not solve their problem (Turn 18). At this point, Ronnie pointed out an advantage of not receiving feedback from the system (simulation) earlier regarding their crashes: it made them inquire more deeply into the issue. They started thinking about multiple factors that could be responsible for the crash (Turn 20) in a systematic manner: "Yeah, because we eventually figured it out - because it was too fast right because we were concerned because it landed right perfectly on the blue line - so we eliminated that" (Turn 22). Ronnie further commented that not receiving feedback can be helpful for learning, but at the same time, she was unsure (Turn 24).

We find this uncertainty to be productive along two dimensions. First, from a pedagogical perspective, the lack of feedback, as both Ronnie and Negin realized through their own experiences, could also be an opportunity for students to dig deeper into the code as well as conceptual issues in physics. Second, from an epistemological

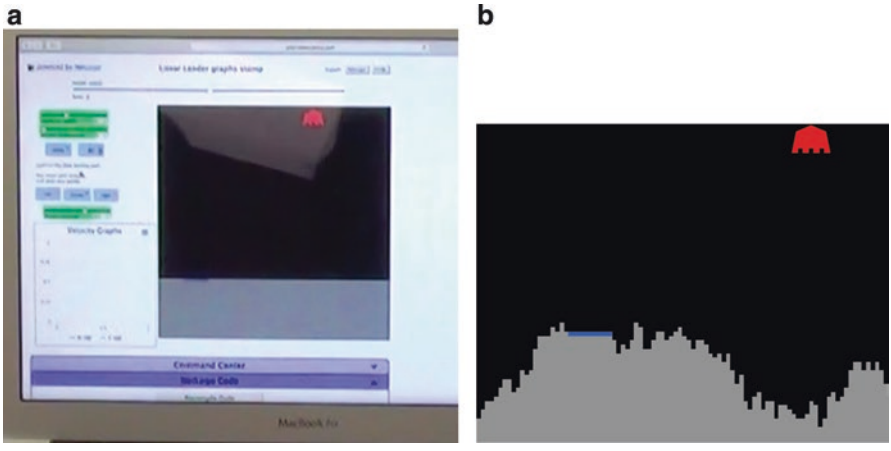

Fig. 10.5 (**a**) (left): Ronnie and Negin's altered simulation showing a flat lunar landing surface; (**b**) (right): The original simulation as provided to the students had a rocky terrain as the lunar landing surface

perspective, for both Ronnie and Negin, in their roles as preservice teachers, this allowed them to view code and coding as pedagogical objects that can deepen their students' scientific inquiry. In this view, coding was not merely a skill to be learnt, rather a means to further student engagement and scientific inquiry.

10.6.3 Theme 3: Coding to Deepen Students' Personal and Playful Engagement with Science

The following excerpt (Excerpt 3) reports a classroom-wide discussion led by Beaumie and Pratim, in which they asked the class to discuss the changes to the code and the simulation that they had considered in order to engage their future students, especially those who may not be interested in the topic. The context of this conversation was the Bird-Butterfly-Flower simulation, which, along with the printed programming guide, provided students opportunities to manipulate the variables in the simulation and the underlying code.

Excerpt 3

Turn 1: Beaumie: So I was wondering actually some of you talked about how, because it was a game and you can think about different colours, there is a different entry point for kids - is there anything that you would change or do something with this other simulation that would provide a different entry point for kids who are not interested in biology?

Turn 2: Adela: One thing after you said lunar lander is made to be able to pick a single butterfly because it would be too messy otherwise and have the trail of the butterfly so you could see its "behaviour"

Turn 3: Pratim: See the graph

Turn 4: Class: That's just energy

Turn 5: Adela**:** Right but I mean to track [motioning with hands tracking]

Turn 6: Pratim: Oh you want to see the path

Turn 7: Mel: You want to see the story of a specific butterfly

Turn 8: Adela**:** Yeah

Turn 9: Mel: You can build a relationship with the butterfly [laughing]

Turn 10: Adela: No, no, no I am totally on that

Turn 11: Pratim: That is exactly how we wrote the paper about the simulation - about how the students can build a relationship with the butterflies on the screen

Turn 12: Mel: Even if you had the lifecycle of the butterfly - so we were talking about having caterpillars and certain birds that would only eat butterflies - it seemed the lifecycle of one bird or butterfly makes it more personal… and some students might want to follow the story of it more

[…]

Turn 15: Pratim: Did any of you feel that this is more simulation and lunar lander is more of a game?

Turn 16: Emily: I felt this was more of a simulation because there is no way to win it

Turn 17: Pratim: And do you think that has implications for learning

Turn 18: Emily: Not necessarily- because I made up my own challenges like, can I fill the screen with butterflies or how many birds does it take to get rid of all the butterflies? So I made up my own goals and played with the different settings that way, but students who might not be able to create their own goals may be bored very easily.

The conversation that ensued consists of two distinct parts. In the first part (Turns 1–12), the conversation focused on altering the code and the simulation in order to help students develop a personal connection with the phenomena being modeled. Here, Adela pointed out (to Pratim) that based on her experience with the Lunar Lander simulation, where the focus was on the behavior of a single computational agent, she realized that focusing on the behavior of a single butterfly in the second simulation would make it easier for students to understand what is going on in the model (Turn 2). When Pratim inquires whether Adela wanted to see the path of the butterfly (Turn 6), Adela's partner, Mel, clarified that the goal was not to merely observe the path of the butterfly and graph its energy change over time (the simulation already allowed participants to do that); instead, their goal was to see the "story" (Turn 7) of the butterfly, i.e., its life cycle. Doing so, according to Mel, would enable the students to "build a relationship" with the butterfly (Turn 9). As Mel further elaborated, poignantly, in Turn 12: "Even if you had the lifecycle of the butterfly - so we were talking about having caterpillars and certain birds that would only eat butterflies - it seemed the lifecycle of one bird or butterfly makes it more personal… and some students might want to follow the story of it more."

In the second part (Turns 15–18), the conversation focused on how coding and other interactions with the simulation were also playful, even though unlike the Lunar Lander game, the Bird-Butterfly-Flower simulation wasn't initially framed as a game. This was evident in the words of Emily, (Turns 16 and 18), who explained that even though there was no way to "win" the simulation, she still made up her own "challenges" such as "can I fill the screen with butterflies or how many birds does it take to get rid of all the butterflies?" (Turn 18). Emily further implied that because she was able to create new goals on her own in order to interact with the simulation (this involved her altering the underlying code along with her partner, as well as changing the variables on the simulation's graphical interface), she was engaged in the activity, noting that students who might not be able to "create their own goals may get bored very easily."

There are two insights to be gained from this conversation. First, it is well established in the educational computing literature that an important affordance of agent-based models is the ability of the learners to easily take on the perspectives of the computational agents in the models (Levy & Wilensky, 2008). Adela and Mel's comments (Turns 1–13) echo this finding. This is important because they believed that this could be pedagogically important, as it would allow their students to "build a relationship" with the scientific phenomenon. At the same time, the variations in their approaches indicate how they use their regimes of competence as productive

resources in creating such relationships. Second, Emily's comments (Turns 15–18) illustrate that even coding and working with computer simulations could be framed as play, which in turn could encourage students to take risks that might take them even deeper within the discipline.

10.7 Summary and Discussion

This chapter makes two contributions. Along one dimension, we present a pedagogical approach for integrating computational modeling in preservice science teacher education by emphasizing preservice teachers' interpretive dilemmas, flexibility, and uncertainties in playfully interacting with the code and computational models. Along another dimension, we also illustrate how code, coding, and computational models get reframed by preservice teachers as pedagogical objects and experiences for doing and teaching science. The analysis presented here focuses on the participants' conversations about both code and pedagogy, a key feature being their deeply intertwined nature.

For example, in Theme 1, we saw how participants, through their own interactions with altering the underlying code of the Lunar Lander model, realized that they could make modifications to the code in order to facilitate their (future) students' engagement with key scientific concepts relevant to understanding the phenomenon represented by the game. In Theme 2, we saw how participants also came to a similar realization—that they could alter the NetLogo code in order to facilitate their students' engagement with scientific concepts—but also experienced a form of productive uncertainty. Upon reflecting on the value of their own scaffolded experience of coding, they wondered whether scaffolding their students might also prevent them from the form of deep explorations that the participants themselves experienced. And finally, in Theme 3, we saw how participants experienced playfulness even when they were presented with a simulation rather than a digital game, realizing the value of being able to set their own goals in their exploration of both the simulation and the underlying code. Across these themes, the frame of mathematization is present throughout—as our participants' engagement with the underlying code often involved altering underlying mathematical parameters and units of measurement (e.g., altering the speed and acceleration of the Lunar Lander, and rate of reproduction in the Bird-Butterfly-Flower simulation). The framing of mathematization is significant given recent findings from several studies that teachers do indeed view coding as mathematization in their classrooms (Sands et al., 2018; Sengupta et al., 2015; Farris, Dickes, & Sengupta, 2019).

Furthermore, as evident in Theme 3, participants also wanted to make alterations to the code so that their students could get opportunities to follow the narrative of an individual agent as a means to help them develop a deep understanding of the complexity of ecological interdependence. Herein lies an often noted affordance of agent-based modeling—that it provides opportunities for learners to take on perspectives of the computational agents, and even draw upon their own embodied and

intuitive knowledge to make sense of the computational representations (Wilensky & Resnick, 1999; Levy & Wilensky, 2008; Farris & Sengupta, 2014). This is in no way an insignificant insight: as Keller (1984) noted, it was thinking like the agent (e.g., a chromosome) enabled the Nobel Laureate Barbara McClintock to make significant advances in her research on human genetic structures.

Overall, as stated in the beginning of this paper, our goal here is to present an epistemological perspective of how preservice science teachers view computational models and code in science education. Each of the three themes we have identified here are examples of epistemological stances in the sense that they reveal how the participants connected their experiences of computational modeling and coding in science with how they would support their future students' scientific inquiry. In this sense, our work also advances a phenomenological agenda in educational computing (Sengupta, Dickes, & Farris, 2018), as our emphasis is on identifying preservice science teachers' sense experiences (Merleau-Ponty, 1962) of coding and computational modeling—i.e., their framing and reframing of what code, coding, and computational modeling can become and can look like in their own imagined futures in science classrooms. The themes we have identified in our analysis, we believe, offer some useful resources that preservice teachers already bring to the table, and a pedagogical approach to build on these resources. We hope our work will inspire more scholarship on phenomenologically grounded, epistemological investigations of computing in K-12 teacher education.

References

Aikenhead, G. S. (2003, August). Review of research on humanistic perspectives in science curricula. In *4th Conference of the European Science Education Research Association (ESERA), Research and the Quality of Science Education. Noordwijkerhout, The Netherlands (August 19–23)*.

Azevedo, F. S. (2018). An inquiry into the structure of situational interests. *Science Education, 102*(1), 108–127.

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning (IJGBL), 1*(2), 65–81.

Bower, M., & Falkner, K. (2015, January). Computational thinking, the notional machine, preservice teachers, and research opportunities. In *Proceedings of the 17th Australasian Computing Education Conference (ACE 2015)* (Vol. 27, p. 30).

Daston, L., & Galison, P. (2007). *Objectivity*. Brooklyn, NY: Zone Books.

Dickes, A., Sengupta, P., Farris, A. V., & Basu, S. (2016). Development of mechanistic reasoning and multi-level explanations in 3rd grade biology using multi-agent based models. *Science Education, 100*(4), 734–776.

Dickes, A. C., & Sengupta, P. (2013). Learning natural selection in 4th grade with multi-agent based computational models. *Research in Science Education, 43*(3), 921–953.

DiSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

Duschl, R. (2008). Science education in three-part harmony: Balancing conceptual, epistemic, and social learning goals. *Review of Research in Education, 32*(1), 268–291.

Farris, A. V., Dickes, A. C., & Sengupta, P. (2019). Learning to interpret measurement and motion in fourth grade computational modeling. *Science & Education*. https://doi.org/10.1007/s11191-019-00069-7

Farris, A. V., & Sengupta, P. (2014). Perspectival computational thinking for learning physics: A case study of collaborative agent-based modeling. In *Proceedings of the 12th international conference of the learning sciences* (pp. 1102–1107). ICLS.

Farris, A. V., & Sengupta, P. (2016). Democratizing children's computation: Learning computational science as aesthetic experience. *Educational Theory, 66*(1–2), 279–296.

Kalogiannakis, M., & Papadakis, S. (2017, August). Pre-service kindergarten teachers acceptance of "ScratchJr" as a tool for learning and teaching computational thinking and science education. In *Proceedings of the 12th Conference of the European Science Education Research Association (ESERA), Research, practice and collaboration in science education* (pp. 21–25). Dublin: Dublin City University and the University of Limerick.

Keller, E. F. (1984). *A feeling for the organism, 10th anniversary edition: The life and work of Barbara McClintock*. New York: Macmillan.

Kim, B., & Ho, W. (2018). Emergent social practices of Singapore students: The role of laughter and humour in educational gameplay. *International Journal of Child-Computer Interaction, 16*, 85–99.

Levy, S. T., & Wilensky, U. (2008). Inventing a "mid-level" to make ends meet: Reasoning between the levels of complexity. *Cognition and Instruction, 26*(1), 1–47.

Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. *Journal of Teacher Education, 69*(4), 386–408.

Manz, E., & Suárez, E. (2018). Supporting teachers to negotiate uncertainty for science, students, and teaching. *Science Education, 102*, 1–25. https://doi.org/10.1002/sce.21343

McMahon, L. (2017). Phenomenology as first-order perception: Speech, vision, and reflection in Merleau-Ponty. In K. Jacobson & J. Russon (Eds.), *Perception and its development in Merleau-Ponty's philosophy* (pp. 308–334). Toronto: University of Toronto Press.

Merleau-Ponty, M. (1962). *Phenomenology of perception*. New York: Routledge.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Thousand Oaks: Sage.

Ochs, E., Gonzales, P., & Jacoby, S. (1996). "When I come down I'm in the domain state": Grammar and graphic representation in the interpretive activity of physicists. *Studies in Interactional Sociolinguistics, 13*, 328–369.

Papert, S. (1987). Information technology and education: Computer criticism vs. technocentric thinking. *Educational Researcher, 16*(1), 22–30.

Pickering, A. (1995). *The mangle of practice: Time, agency, and science*. Chicago, IL: University of Chicago Press.

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking. In *Computational thinking in the STEM disciplines* (pp. 151–164). Cham: Springer.

Sengupta, P., Brown, B., Rushton, K., & Shanahan, M. C. (2018). Reframing coding as "Mathematization" in the K12 classroom: Views from teacher professional learning. *Alberta Science Education Journal, 45*(2), 28–36.

Sengupta, P., Dickes, A. C., Farris, A. V., Karan, A., Martin, K., & Wright, M. (2015). Programming in K12 science classrooms. *Communications of the ACM, 58*(1), 33–35.

Sengupta, P., Krinks, K. D., & Clark, D. B. (2015). Learning to deflect: Conceptual change in physics during digital game play. *Journal of the Learning Sciences, 24*(4), 638–674.

Sengupta, P., Kinnebrew, J., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K12 science education using agent-based modeling: A theoretical framework. *Education and Information Technologies, 18*, 351–380.

Sengupta, P., & Shanahan, M. C. (2017). Boundary play and pivots in public computation: New directions in STEM education. *International Journal of Engineering Education, 33*(3), 1124–1134.

Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in STEM education. In *Computational thinking in the STEM disciplines: Foundations and research highlights* (pp. 49–72). Cham: Springer International Publishing.

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). *NetLogo*. Evanston: Center for Connected Learning and Computer-Based Modeling, Northwestern University. Retrieved from http://ccl.northwestern.edu/netlogo/

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—an embodied modeling approach. *Cognition and instruction, 24*(2), 171–209.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology, 8*(1), 3–19.

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM, 57*(8), 24–28.

Wilkerson, M. H., Andrews, C., Shaban, Y., Laina, V., & Gravel, B. E. (2016). What's the technology for? Teacher attention and pedagogical goals in a modeling-focused professional development workshop. *Journal of Science Teacher Education, 27*(1), 11–33.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62.