Computational Modeling With Multilingual Learners

Integration across four science units

By Alison Haas, Scott E. Grapin, Lorena Llosa, and Okhee Lee

While the vision for science education through A Framework for K–12 Science Education (NRC 2012) and the Next Generation Science Standards (NGSS) continues to take hold in classrooms across the nation, computational modeling is becoming increasingly essential in school and society. Computational models, or “representations of phenomena that can be simulated by a computer” (Weintrop et al. 2015, p. 137), are reshaping the way science is practiced in increasingly diverse classrooms, as all students, including multilingual learners (MLs), can use computational models to develop and test explanations of phenomena. However, teachers lack high-quality science curricula that integrate computational modeling in purposeful ways and with explicit attention to student diversity.

The purpose of this article is to share our approach to integrating computational modeling across a yearlong, fifth-grade NGSS-designed curriculum. First, we highlight the affordances of computational modeling with MLs (National Academies of Sciences, Engineering, and Medicine 2018). Then, we illustrate how these benefits are evident in the four science units that make up our curriculum. Finally, we conclude with recommendations, which address differentiation and assessment, for teachers interested in integrating computational modeling into their own science units.

Affordances of Computational Modeling With MLs

Developing models and using computational thinking are two science and engineering practices in the NGSS. In the NGSS classroom, students develop and use a wide range of models (e.g., physical, diagrammatic) to explain phenomena. The practice of developing and using models is particularly beneficial for MLs, who benefit from using multiple meaning-making resources to communicate ideas (Grapin 2019). One way for students to develop computational models is through blocks-based programming environments, which allow users to program agents in a system and give those agents dynamic rules of behavior (Klopfer 2003). Thus, integrating agent-based computational modeling into the science classroom affords all students, and especially MLs, new opportunities to develop and
communicate their understanding of complex systems. In Table 1, we list five key affordances of computational modeling that motivated our curriculum design.

**TABLE 1**

Affordances of computational modeling.

<table>
<thead>
<tr>
<th>Affordance number</th>
<th>Computational models allow students, including MLs, to . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>make sense of causal mechanisms that are too small or too big to see</td>
</tr>
<tr>
<td>2</td>
<td>investigate a system at different levels (individual agent level and whole system level)</td>
</tr>
<tr>
<td>3</td>
<td>test ideas that would be difficult to implement in the real world</td>
</tr>
<tr>
<td>4</td>
<td>generate and collect data</td>
</tr>
<tr>
<td>5</td>
<td>interpret and represent ideas using multiple meaning-making resources (e.g., code blocks, dynamic visualization)</td>
</tr>
</tbody>
</table>

**Integrating Computational Modeling Into Curriculum**

The four units, which address all 16 NGSS performance expectations for Grade 5, were developed in collaboration with teachers and went through iterative cycles of development, field testing, revision, and refinement. In each unit, students use StarLogo Nova, an agent-based, blocks-based programming environment, to develop a computational model and help answer the unit driving question. StarLogo Nova is free and publicly available.

In preparing to implement the curriculum, teachers participate in eight days of professional development, two days for each unit. In each unit, teachers group students intentionally. For example, students work in investigation groups of about four students. When developing computational models, each investigation group is split in half so two students work on one computer. Then, investigation groups reconvene to discuss the computational models.

Across the four units, students actively engage with the anchoring phenomenon before developing computational models. The computational modeling tasks follow a similar instructional progression in each unit. First, students engage in “storyboarding” to share ideas and plan how they will translate these ideas into code. Second, students interpret a starter model given by the teacher (via the curriculum), which provides a foundation of code blocks so students are not overwhelmed by programming demands. Third, students develop their computational models by modifying the code in the starter model and adding their own code. Fourth, students participate in design meetings in which they provide feedback and evaluate each other’s models. Finally, students revise their computational models based on feedback from their peers and teacher.

The computational models also serve as assessment artifacts. As formative assessments, while students develop their models, teachers attend to partner discourse to get a sense of students’ current thinking and ask probing questions provided in the curriculum to push that thinking forward. Teachers are provided with both general prompts (e.g., “What is this code telling the agent to do?”) and model-specific prompts (e.g., “What do you think would happen to the tiger salamanders if there were a decrease in algae?”). As a summative assessment of each unit, teachers examine the code in students’ computational models and triangulate the code with other artifacts (e.g., diagrammatic models, written explanations) to assess students’ science understanding.

Across the four units, computational modeling is used for different purposes to capitalize on various affordances. Next, we describe how these affordances of computational modeling (see Table 1) are evident in our curriculum with a focus on MLs.

**Unit 1: Explaining What Happens to Garbage**

In the physical science unit, students experience the phenomenon of garbage in their home, school, and community. Students observe the properties of their lunch garbage and identify the unit driving question, What happens to our garbage? They develop “landfill bottles” by putting garbage materials in open and closed bottles and observing what happens to the garbage over time. Over several weeks, when the weight of the open system (but not the closed) decreases and the bottle begins to smell, students develop diagrammatic models to show how smell particles remain inside the closed system but move freely out of the open system.

Computational modeling enters the unit to explain the mechanism by which solid food materials change to gas particles (Affordance 1 in Table 1). After students figure out that microbes (i.e., a type of decomposer) are present in the landfill bottles, students develop computational models to represent how microbes too small to see decompose solid fruit (e.g., a banana) and produce gas in the closed landfill bottle system (see Figure 1). By using computational models to represent the closed landfill bottle system, students can examine different levels within that system to figure out how agent-level interactions produce system-level behaviors (Affordance 2). For example, students create agents (e.g., banana solid, banana gas, microbes) and program the interactions between those agents. Students know from data collected in their physical landfill bottles that, at the whole system
level, the weight of the closed system does not change over time. Thus, students need to program microbes to interact with the banana solid in a way that conserves weight in the closed system (e.g., microbes, on collision with banana solid, delete the banana solid and create banana gas, as shown in Figure 2). Giving microbes different rules of behavior allows students to test ideas that would be difficult to implement in the real world (Affordance 3). As students run their models, data boxes and a graph (Affordance 4) track (a) the weight of the solid banana, (b) the weight of the banana gas, and (c) the total weight of the banana, which students then compare to the weight data from their physical landfill bottles. In the example shown in Figure 1, students developed a computational model in which the total weight of a banana remained the same (i.e., 500), even as the banana solid (i.e., 457) changed to banana gas (i.e., 43), which is consistent with the weight data from the physical landfill bottle (i.e., weight is conserved in both models). Ultimately, the microbe model affords all students, but especially MLs, the opportunity to interpret and represent ideas using multiple meaning-making resources (Affordance 5). For example, programming the code blocks shown in Figure 2 enables all students to communicate explicitly about the ratio of solid banana deleted to gaseous banana created (1:1).

Unit 2: Explaining Why an Organism Disappeared From an Ecosystem

In the life science unit, students experience the phenomenon of the disappearing tiger salamanders. With the help of an ecologist, students find out that tiger salamanders have disappeared from a local vernal pool in their community and identify the unit driving question, Why did the tiger salamanders disappear? They make observations of a local vernal pool ecosystem where tiger salamanders once lived but have since disappeared. As students obtain information about organisms in the ecosystem, they develop diagrammatic and physical models (i.e., each student acts out the role of an organism) to represent predator/prey relationships and the transfer of matter and energy through the ecosystem.

Computational modeling enters the unit to make sense of causal mechanisms within the vernal pool ecosystem (Affordance 1). Throughout the unit, students build a more
complex version of the ecosystem by adding organisms to their computational models and programming how those organisms interact. For example, students create agents (e.g., tiger salamander, garter snake, algae, zooplankton) and program the interactions between those agents (e.g., tiger salamanders, on collision with fairy shrimp, delete the fairy shrimp and gain energy, as shown in Figure 3). The computational models allow students to investigate the complex vernal pool ecosystem by moving back and forth between the individual agent level (e.g., what each organism eats) and whole system level (e.g., what happens to the health of the ecosystem as a whole) to make sense of both direct and indirect effects (Affordance 2). Students use their computational models to test different possible causes of the tiger salamanders’ disappearance, testing ideas that would be difficult to test in the real world (Affordance 3). Students manipulate organism populations by changing the slider values for garter snakes (i.e., predator of tiger salamanders) and fairy shrimp (i.e., prey of tiger salamanders) to test whether an increase or decrease in these organisms could have caused the tiger salamanders to disappear (see Figure 4). Each time students run the model, they generate data to track the number of organisms present in the ecosystem (Affordance 4). The computational models allow students to interpret and represent ideas using multiple meaning-making resources (Affordance 5).

Unit 3: Designing an Engineering Solution for Plastic Pollution

In the Earth sciences unit, students experience the phenomenon of tap and bottled water by engaging in a blind taste test to determine whether they can observe differences between tap and bottled water. Students identify the unit driving question, Why does it matter if I drink tap or bottled water? They trace the source of both tap and bottled water and figure out that, in their community, the tap and bottled water come from similar sources. They develop physical and diagrammatic models to show how water moves around. They also figure out that plastic water bottle pollution is harming the environment.
Computational modeling enters the unit in service of engineering. Students develop computational models to design and test solutions to the problem of plastic pollution in their community. Students design solutions that would be difficult in the real world due to constraints of time, money, and resources (Affordance 3). For example, one group modeled a social media campaign to encourage peers at their school to use the water fountain instead of plastic water bottles (see Figure 5). Students programmed people (yellow circles) to drop plastic water bottles (blue squares) at random intervals. When people collide with the water fountains (black circles), there is a 25% chance that the people will convert (red circles) to reusable bottles and stop dropping plastic bottles (see Figure 6). The computational model includes a graph that tracks the total number of plastic bottles used over time (Affordance 4). With MLs, computational models are particularly powerful in allowing them to use multiple meaning-making resources to communicate their ideas (Affordance 5). When they run their model, they get immediate feedback from the graph to determine if their design solution is efficacious in reducing the total number of plastic bottles used by the community. They use the feedback to refine their design solution.

**Unit 4: Explaining Why Falling Stars Fall**

In the space science unit, students experience the phenomenon of falling stars. Students observe the properties of meteorites, or “real” falling star pieces, and identify the driving question, *Why do falling stars fall?* They carry out a series of investigations (e.g., measuring the length of shadows at different times of the day) that produces evidence of Earth’s rotation, orbit, and gravity. Based on this evidence, students develop physical and diagrammatic models of the space system to explain why we see falling stars at night (i.e., Earth’s rotation), why we see different falling stars at different times of the year (i.e., Earth’s orbit), and how falling stars fall to Earth (i.e., Earth’s gravity).

Computational modeling enters the unit to make sense of causal mechanisms that are too big to see (Affordance 1). Students use their computational models to better understand the relationship between Earth’s movements (i.e., rotation and orbit) and the patterns caused by each movement. The
computational models allow students to examine different levels within the expansive space system (Affordance 2). Students toggle between an “Earth view” (the perspective of a single agent, Earth) and a “Space view” (the broader perspective of space to see how changes to Earth impact the space system as a whole). Specifically, this model allows students to test counterfactuals about the relationships of agents in the space system, an unrealistic feat in the real world (Affordance 3). Students test what would happen if Earth’s rotation, orbit, and gravity worked differently or did not exist (see Figure 7). For example, students modify their code so that a day on Earth lasts one hour (instead of 24) and then predict whether, in this one-day rotation, Earth would rotate faster, slower, or at the same speed (see Figure 8). Because code for Earth’s movements requires trigonometry beyond the scope for Grade 5, some of the more complicated code is provided for students. As students model and test different ideas, they collect and record data in an investigation sheet (Affordance 4). This model allows students, especially MLs, to leverage multiple meaning-making resources (Affordance 5). They program Earth to move in different ways and run the model to see what that movement looks like in the broader space system.

**Recommendations for Teachers**

Integrating computational modeling into NGSS instruction can often feel daunting. However, when field-testing our curriculum, teachers frequently reported that thinking about implementation was more overwhelming than actually doing it. We conclude this article by offering four recommendations for teachers interested in integrating computational modeling with MLs into science units.

First, start small. When first attempting integration, start by integrating computational modeling into a single unit.
For example, consider adding an engineering component that integrates computational modeling into an existing science unit. Many teachers are surprised at how savvy students are with technology, which makes implementation not only manageable but engaging for all students.

Second, utilize and modify existing resources. Many blocks-based programming environments are free and have curricular resources for teachers. For example, StarLogo Nova houses hundreds of free projects that can be modified to fit the needs of your students.

Third, leverage the affordances of computational models. Computational models can be leveraged to fit the needs of different science domains. For example, physical science often requires explaining causal mechanisms that are too small to see, while space science often requires explaining causal mechanisms that are too big to see.

Finally, and perhaps most importantly, embrace the messiness. Programming computational models requires students to think critically, as there is not a single “correct” answer. Embrace the moments when students are playing, exploring, and figuring out how to develop their models, as this is often when the most fruitful learning happens.

REFERENCES

ONLINE RESOURCES
StarLogo Nova
www.slnova.org
www.slnova.org/resources

Alison Haas (Ams728@nyu.edu) is the director of development and implementation for the SAIL Research Lab at New York University. Scott E. Grapin (sxg1544@miami.edu) is an assistant professor at the University of Miami in Coral Gables, Florida. Lorena Llosa (lorena.llosa@nyu.edu) and Okhee Lee (olee@nyu.edu) are professors at New York University.