Finding Abstraction-Comprehension Balance: A Study of Model-Based Integrated Science and Computational Thinking Application

Aakash Gautam[1], Whitney Bortz[1], Deborah Tatar[1], Kemper Lipscomb[2],
Stephanie Rivale[2], Christina Orsino[1], Felicia Etzkorn[1]
[1] Virginia Tech, [2] The University of Texas, Austin
{aakashg, whitney8, dtatar, corsino, fetzkorn}@vt.edu, {kemperlipscomb, srivale}@utexas.edu

Abstract: Computational thinking (CT) has received considerable attention over the past decade among educators and researchers. However, given the multiple goals of the educational system, and already demanding school curriculum, there is limited scope for adding computational thinking as a separate subject in schools.

This project integrates CT and Chemistry in activities for middle school. By "activity", we mean a NetLogo-based Chemistry environment which uses both the simulation and the structure of the underlying code model as resources for teaching, along with chemistry and computational thinking curriculum materials, and teacher professional development. This paper discusses tensions identified during the design process concerning the need to abstract information while making it comprehensible and instrumental for targeted pedagogical goals. We hope that our findings will illuminate some issues in designing an integrated activity and inform future research.

Objectives

The ability to think with computing mediums as tools is necessary for everyone, not just computer scientists (Wing, 2006, 2008; CST, 2010; Tatar et al., 2017; Kafura and Tatar, 2011). However, establishing computer science (CS) as a subject may not be the complete answer. First, K-12 curriculum is already overloaded. Second, CS itself may not motivate students from all backgrounds and interests. Third, one isolated CS course does not suffice to prepare students for a variety of fields. Fourth, there are not enough CS teachers. Fifth, CS does not necessarily look the same from the perspectives of different uses.

Our research addresses the problem of thinking with computer mediums through advocating and exploring an approach that integrates computational thinking and already existing school topics with the aim of deepening both. We have developed computational environments for middle- school students that allow them to explore the properties of chemistry and earth-science models. This research brings together four distinct groups of researchers: chemistry experts, computer scientists, science educators and researchers on computational thinking. Such diverse groups of people have differing views about design and implementation of activities.

In this paper, we describe some of the design conflicts and decisions we made based on compromises and satisficing (Simon, 1972) - finding "good-enough" solutions for multiple goals rather than optimizing on one particular goal. We believe that identifying some of the design tensions (Tatar, 2007) would help in finding overarching design configurations that can guide the development of effective integration of science and computational thinking.

Perspectives

Computing in K-12 gained prominence with Logo programming language in the 1980's (Papert, 1980). In 2006, Jeannette Wing's influential article (Wing, 2006) on the importance of computational thinking as a necessary attitude and skill brought wider interest among educators and researchers. Before Wing, a long line of thinkers had developed and used computer applications in formal and informal learning environments. In the 1960s, Alan Perlis argued for the need to teach the theory of computation to college students of all disciplines (Perlis, 1962) and similar arguments were made by Alan Kay with the idea of the Dynabook and the Smalltalk language (Kay & Goldberg, 1977). Papert brought the idea of computers as providing objects to think with to classrooms. The Logo programming language was designed with this in mind, and the extended in such products as Atari© Logo with animated sprites. Along similar lines, diSessa proposed the idea of "computational literacy" in which he argued for the use of computing as a medium to change the way people think and learn (diSessa, 2001). His approach found concrete expression through the Boxer language (diSessa and Abelson, 1986). Resnick and his colleagues have addressed these themes through both theory and the design of a series of environments and facilities with which children can both think and create: Lego Logo, Scratch, NetLogo and others (Resnick and Ocko, 1990; Resnick, Ocko and Papert, 1988; Martin and Resnick, 1993; Maloney, Resnick et al. 2010; Resnick, Maloney et al., 2009; Tisue and Wilensky, 2004; Bilkstein et al,. 2005; Wilensky, 1999; Wilensky and Stroup, 2002). These approaches have led to yet others: sewable circuitry (Buechley, 2010; Lovell and Buechley, 2010), Snap (Harvey, Garcia, et al, 2014), and now block languages such as BlockPy (Bart, Tilevich et al., 2017; Bart and Kafura, 2017). Similar calls to use computational medium to deepen learning were made by other researchers (e.g. Guzdial, 1994). While many have argued that computational thinking has a place in K-12 learning, an agreed definition does not exist in the literature (Grover & Pea, 2013). For this work, the team has adopted a working definition of computational thinking as "an intellectual skill rooted in the ability to conceive of meaningful, information-based representations that can be effectively manipulated by an automated agent (e.g., a computer)" (Kafura, Bart & Chowdhury, 2015).

Our project investigates a method of broadening access to CT by providing it in science learning contexts that all students take. The ultimate target, not reported here, for the replacement units developed so far was 8th Grade Integrated Science, but for purposes of refinement we tested parts and the whole sequence with younger students in informal and formal learning settings.

This CHEM+C method exposes students to how CT interacts with and deepens chemistry thinking. The method is to embed targeted teaching/learning of chemistry topics into editable, runnable computational models that will themselves be embedded in a pedagogy of structured scientific argumentation, featuring the production of evidence-based reasoning. Students will learn about chemical systems through interacting with models in a number of guises: as they are represented visually through simulations, especially those that combine macroscopic and microscopic features, as models occur in standard presentations of Chemistry, and as they are represented through code. Students learn about CT through interaction with models, by deriving their own representations and accounts, by comparing them, and by modifying and adapting the code to better represent aspects of the chemical systems. Students learn about chemical systems. Key concepts include collision theory, conservation of matter, transformation of energy, equilibria, and reaction rates.

In this paper, we focus on tensions in the design of one of the activities that we have developed. This activity aims to help students understand molecular behavior and properties. An important idea in early approaches to computer use was that students are not a blank slate, but come to the computer with ideas formed from their lives (Papert, 1980; Harel and Papert, 1990; diSessa, 2001). When they encounter new ideas, they attempt to make sense of them using the concepts that they have already used. When these prove inadequate, they may move towards "thinking like a scientist" (Papert, 1980, drawing on Jean Piaget). For example, based on everyday experience, some students have the reasonable but wrong ideas that atoms transform into new atoms during chemical reactions (Herrmann-Abell & DeBoer, 2011) and that molecules form from isolated atoms (Özmen, 2004). As a result, when learning a new scientific idea, they may fail to properly notice or understand the parts that don't align with their preconceptions (Arnaudin & Mintzes, 1985).

While designing this activity, we came across several conflicting expectations from domain experts: the chemistry experts, computer scientists, the science educators, and the computational thinking researchers. To design within the limited resources, and multiple choices and perspectives, we relied on a Design Tensions Framework: "*Design tensions conceptualize design not as problem solving but as goal balancing. They draw explicit attention to conflicts in system design that cannot be solved but only handled via compromise.*" (Tatar, 2007)

The Overarching Pedagogical Approach

The structure of each unit has five parts:

(1) To make the phenomena vivid and real, our curricular activity starts with an anchoring phenomenon, an interaction that invites inquiry or demands explanation. In this case, the anchoring phenomenon is a physical demonstration in the classroom of the electrolysis of water. Students put a battery into a solution of water and Epsom salts. By putting test tubes full of the solution over the anode and cathode of the battery, students can collect and observe the gasses emitted as a result of the reaction.

(2) Students are asked to discuss what is going on in small groups and in whole class discussion. They use whiteboards or large poster paper to draw answers to the question of what is happening. Public presentation of these allows students to make evidence-based arguments. They speculate on what the gasses are and where they come from. In this way, they construct static models of the reaction and, aided by the teacher, derive the reaction in the standard form of a chemical formula, $2H_2O \rightarrow 2H_2 + O_2$. Yet there is considerable ambiguity in the demonstration: do, for example, the gasses come out of the battery itself? The electricity is, of course, invisible.

(3) Subsequently, the students are introduced to an animated simulation in the NetLogo
programming environment (Wilensky, 1999) that shows molecules moving around and both forming and decomposing water (the Water Splitting and Formation simulation, shown in Figure 2). Students use this to explore freely. They may turn electricity on and off, adjust the starting amounts of water, oxygen and hydrogen, and change the temperature (see Appendix A: Figure 2).

(4) Eventually, they fill out a “design component chart” which asks them a number of questions including (1) “what is happening in the simulation?”; (2) “what about the model is scientifically accurate?” and (3) “what about the model is inaccurate?” This activity invites the students to reflect and also reveals evolution in student thought.

(5) Engagement with these different models invites thought about the nature of models themselves, an important part of computational thinking (Weintrop et al, 2016). But then we take it to the next level. Another kind of model is then introduced: the computer program that implements the simulation. Students change the program first through the Command Center and then through changes to the actual code.

Part 5 enlarges the exposure that students have to computational thinking in two ways. First, it exposes students to the language of computing. Of course, we have many computer languages, but arguably all of them are languages for modeling in the virtual world, something that is of interest in the non-virtual world. Constructs, for example, such as “if” are formalized. Second, students may use the power of computer language to change and explore the scientific model in more depth. This moves student towards having the freedom to create, but it also contrasts the ways we must think to use computers (that is, computational thinking) with both ordinary thought and the scientific thought that they are trying to master.

NetLogo itself is constructed on the Logo principle of “low floors and high ceilings” (Papert, 1980). Although the Water Splitting and Formation simulation that the students have been using is somewhat complex for a first program, we provide at least one “low floor” asking students to add a new kind of computer object to the code model (probably as Epsom salts) that will then appear visibly in the simulation. Since all the objects in this simulation are whole molecules, adding Epsom Salts can satisfy the students’ need to see representations of the concrete elements they saw in the demonstration without requiring that they code the actual behaviors of creating ions. Of course, if the students are interested in why the Epsom Salts are necessary for the reaction to take place, this is a good thing that that the teacher can use to open up further scientific discussion and that also raises really important questions about the simulation such as “Why, for example, are the objects molecules and not atoms?”, “What about parts of atoms?” and so forth.

Methods

The major method used in the design of the replacement units is Design-Based Research (Brown, 1992; Brown and Campione, 1996; Collins, 1992; DBRC, 2003; Barab and Squire, 2004) which focuses on the small, iterative adjustments that are crucial to making an educational technology and related materials “work” in the classroom in a way that teachers can utilize. However, because an explicit part of the design was mediating between the customs, expectations and requirements of different stakeholder groups, we were also influenced by the Design Tensions Framework (Tatar, 2007).

As part of the design process, we had multiple discussions with experts from the four different domains involved. These discussions involved identifying the absolutely required properties, the wished-for properties, the possible mechanisms for attaining those properties (through curriculum, teacher professional development, the design of the simulation, the design of the underlying code), and, eventually, the unexpected (positive or negative) entailments of particular decisions or combinations of decisions. We characterize some of the conflict by identifying the relationship between different groups' perspectives on "what is" and "what ought to be" as posited in the design tensions framework. Here, "what is" includes the state of understanding of the students, the materials they encountered, and the sense they were able to draw from those materials. "What ought to be" are the representations and meaning that at least some experts wanted to see explicitly represented in a form interpretable to them.

Acknowledging the tensions, and finding a compromise has helped us design our intervention. This involved a delicate dance in which we sought to achieve a balance across the various learning domains while prioritizing activities and approaches that addressed the research questions guiding the project.

Data Sources for This Analysis

The design partners expressed elements and properties they thought should be present in the model. We analyzed those requirements initially and in the face of student reaction and behavior.

We tested an initial version of the simulation for an hour with two groups of fifth-grade students with each group of around 10 students. The rationale for working with 5th grade students (ages 9- 10) was the criterion that if 5th grade students could in an hour of focused activity, figure out what the objects and activities in the simulation represented, then it should be easy for wide variety of older students, even those who had missed a lot of school, to perceive the intended meaning. We used the observations from this pilot intervention to identify further issues in the design of our activity. It also helped us in identifying design choices that worked well.

We then conducted a week-long intervention in two middle-school classrooms using the refined materials and approaches. Thirty-three 7th grade students in two different science classrooms in rural Floyd County, Virginia participated in a two-week long Chemistry plus CT intervention. In this paper, we present findings based on the pilot intervention and the classroom sessions.

Results

The driving question for the instructional activity was: "What is happening that we cannot see?" The overall chemistry themes involved collision theory, conservation of matter, and stoichiometry. However, the challenge when working with diverse disciplinary backgrounds is that differing ideas exist of how these objectives should be met through the materials and/or the instruction. The following discussion and examples demonstrate our iterative process of coming to a design and implementation that generated evidence of the desired student learning.

Understand that matter consists of atoms grouped together as molecules

One of the learning objective was to be able to differentiate between atoms and molecules, and understand that atoms are conserved whereas molecules are transformed during a reaction. This required a suitable implementation of visual elements to explain the concepts both in the simulation and code-based model.

What is - The initial simulation represented molecules that eventually reached an equilibrium. Molecules disappeared during reaction and other molecules appeared. A graph of the amount of each kind of molecule over time showed that they had reached equilibrium via the roughly parallel amounts. The mechanism (that is the repositioning of atoms into new conformations aided by energetic changes) was not clear.

What ought to be - Chemistry: Atoms should be individual agents in the model since the reaction is fundamentally the regrouping of atoms.

What ought to be – CS: Atoms as individual agents require large computing resources. Molecules should be represented with consistent colors and shapes.

What ought to be – Science Pedagogy: Atoms will be difficult to comprehend visually and will confuse students. Students discuss molecules in classrooms so they should begin their inquiry through molecular processes.

What ought to be – CT pedagogy: Students should be able to realize the transformation of molecules during a reaction by exploring the simulation and without a lot of direct "teaching."

Initial design decision: We decided to model molecules as agents while trying to convey regrouping by selecting specific shapes - hydrogen as gray squares, oxygen represented with red triangles, and water with blue triangle and square stacked together (called "house") (see Figure 1). When run, the simulation showed molecules moving around randomly and sometimes being transformed. A graph showed the relative amounts of H_2O, O_2, and H_2.

Testing the design decision: However, from our first fifth grade intervention, we found that students soon stopped looking at the actual simulation. Evidently, they could not understand transformation through the shapes. Many students said that the red triangles represented heat. Some thought that heat came from the electricity while others thought that heat was responsible for evaporating water. They did look at the graph, and the legend on the graph showed many the intended meaning of the colors. However, as the system went to equilibrium, the graph became less interesting.

Based on these observations, we concluded that the shapes were arbitrary abstractions that failed to convey sufficient information. In a second design iteration, we acknowledged the atomic structure of molecules by representing each as a group of atoms in which each atom was visible as a round object with some shading to indicate that it was a sphere rather than a flat circle. The center of each atom was labeled with its respective symbol - “H” for hydrogen and “O” for oxygen (see Figure 1). This was more recognizable and invited students to inquire about atoms and their relationship with molecules. Furthermore, to show the occurrence of a reaction, we added arrows in our simulation representing endothermic and exothermic reactions.

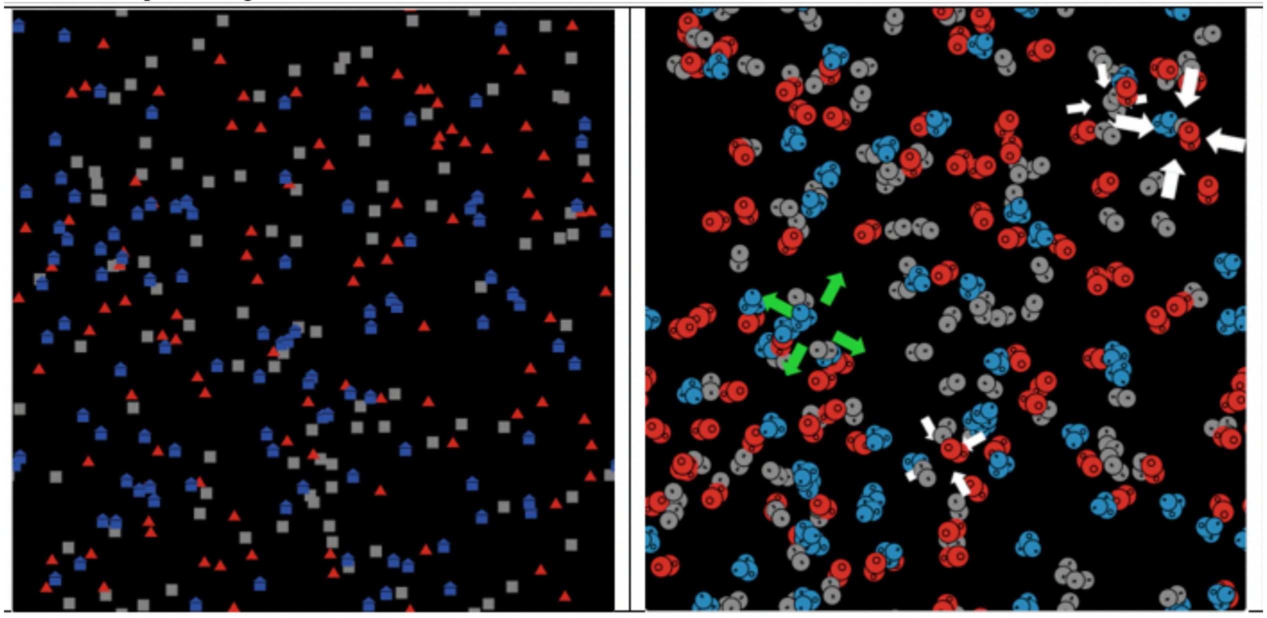


Figure 1: The left image shows the initial representation of molecules. Red triangles were oxygen, gray squares were hydrogen and the blue house represented water. The image on the right shows the changed representation following the pilot intervention.

Testing the design in the 7th Grade classrooms: On the first day of the intervention, students conducted demonstrations of the electrolysis of water. They then speculated the phenomena based on the physical observation, and held in-class discussions that broached the issue of water splitting and what the gas in the test tube was.

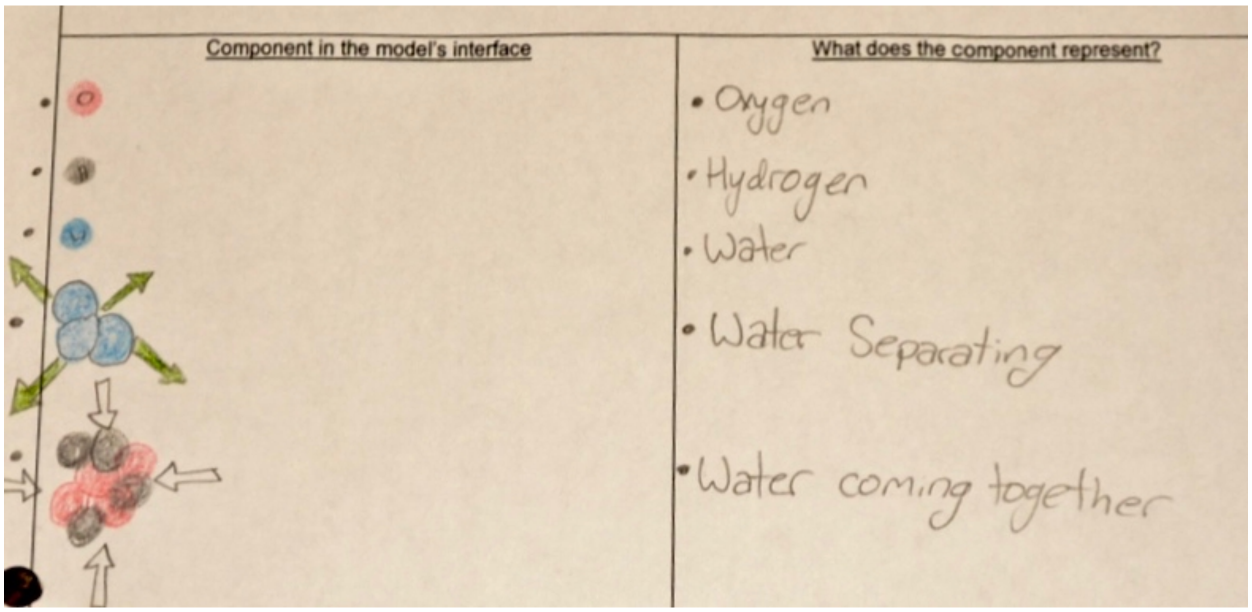
We then introduced the computer simulations. These 7th grade students attended to the simulation for much longer, particularly focusing on the arrows. They were able to produce explanations that mentioned the movement of molecule as important aspects of the microscopic world, and the presence of different molecules at the same place at the same time as necessary for the production of the arrows (e.g. the reactions). Figure 2 shows the work of a group who were able to interpret some elements of the simulation as we had intended and in way that tied the simulation to the demonstration. However, misconceptions are evidenced in this drawing. First, the simulation never shows single atoms, and while the students have successfully drawn the requirement to have multiple hydrogen and oxygen atoms as present for water to form, they have not made it clear that the hydrogens exist as molecules (not single atoms) and they have drawn a single water molecule as separating. An interesting twist is that the arrows, which we conceived of as showing energetic states, are (correctly) interpreted from the point of view of water.

That is, the exothermic reaction is water separating, and the endothermic is water coming together. This progressive presentation helped connect observed phenomena to that being shown in the simulation (see Figure 2).

Work with the simulation by itself does not complete the learning, but the simulation sets up the topic for subsequent whole class discussion:

> Instructor: *So, when you look at the red one, what is that a picture of?*
> Student 1: *Oxygen*
> Instructor: *Is it an oxygen atom or oxygen molecule?*
> Group of students: *Molecule*
> Instructor: *Why do you think it might be a molecule?*
> Student 2: *It's because a molecule is when there is two or more atoms.*
> Instructor: *You've got to have two oxygens together. How about this one? Gray ones, what are those?*
> Student 3: *Hydrogen*

We used the representations presented in simulation along with the instructions to discuss the differences between atoms and molecules. In addition, as part of the CT skill of thinking about representations, we pushed students to compare the simulation and model with real life. Using the abstract representations, we discussed some of the the advantages and limitations of the model. "*I was going to say the arrows were showing coming together and separating*", said a student during our class discussion and many remarked, "*I don't think in real life you would have those*".



Component in the model's interface	What does the component represent?
	• Oxygen
	• Hydrogen
	• Water
	• Water Separating
	• Water coming together

Figure 2: A sample worksheet from our classroom intervention where we see that the student has identified the objects presented in the simulation as well as the actions that occur in the simulation

Understand the model that controls the behavior of the simulation

A necessary CT skill is to understand how representations are manipulated by an automated agent which, in our case, was a computer executing a NetLogo program. A goal was to help students understand how the code-based model was implemented (and reflected in the simulation) while connecting it with the science they had learnt.

What is - Students are unfamiliar with programming practices, models, and code. Normally, we would follow the Logo style and priorities of introducing code bit-by-bit, initially introducing features that tie the notion of commands or objects to elements visible in the simulation and becoming progressively more abstract over time. Whether language is the primary form of manipulation or whether language is embedded in visual constructs (as in block languages), the computer language itself draws to the extent possible on constructs and syntax that are familiar from English. Ideally, the introduction of constructs would be gated by the learner's goals and ambitions, but even when a set series of projects is followed, the programs build from little and simple to bigger, more complex or bigger and more complex. However, following this path is challenging in the context of this project because this project leads with science and goal of science learning. For this reason, the models we present are not as simple and targeted as the kind of models we usually use to present in introductory computer science. Furthermore, we provide considerable pedagogical scaffolding, as outlined above.

What ought to be - Chemistry: Students should understand the chemical phenomena by understanding the operations of fundamental laws such as collision theory and conservation of matter. They should realize the role of Epsom salt in the reaction as an electrolyte and the need for having electrolytes to conduct electricity.

What ought to be – CS: There are three kinds of objects in the simulation (hydrogen, oxygen, and water) and there are many of each kind. Adding Epsom salt objects in the simulation would make it difficult to see the simulation clearly and could also make the simulation slower.

What ought to be – Science Pedagogy: The code should make the chemical model clear. Students should understand that molecules move around randomly and that reactions occur by chance when the conditions are right.

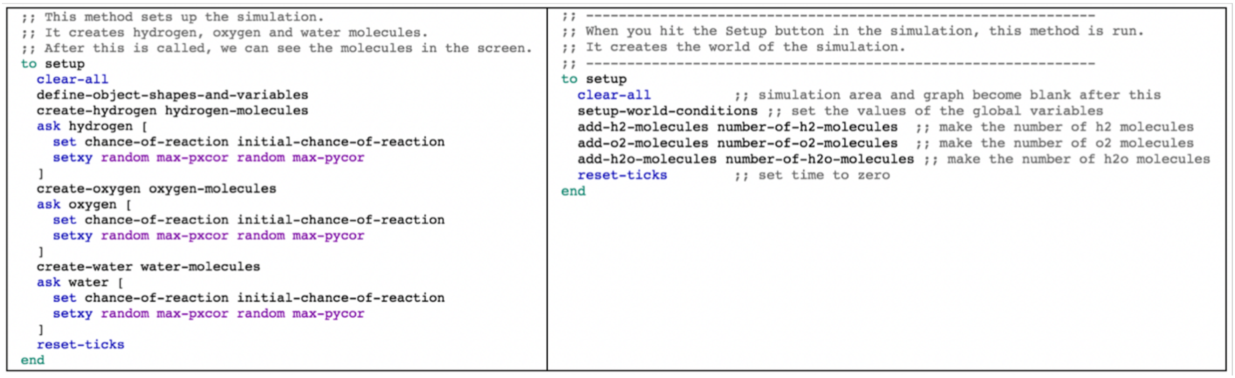
What ought to be – CT pedagogy: Initially, students will read code but only a few lines of it. Those lines will clarify the overall phenomena and they will explore individual methods to understand deeper.

Initial design decision: To simplify the implementation, the underlying code model did not implement atoms; only molecules. Therefore, the code was shorter and more reflective of what was seen in the simulation. We decided not to include Epsom salt, and instead help students realize the missing Epsom salt and add it as part of their programming challenge.

In the code, we assigned properties and actions to the objects where it was created. Our intention was for students to read a higher-level code and understand properties of the objects that would help in understanding the phenomena observed in the simulation. For example, from our initial design discussions, we concluded that students should infer that reactions occur by chance. We also wanted students to see that the phenomena would hold true irrespective of where the molecules lie in the microworld. We made change in our model accordingly (see Figure 3 Left).

However, having such structure of code made it harder for students to follow the top-level processes. It also meant that students were introduced to some primitives and processes that were necessary to implement the model but were hard to comprehend. For example, when two H2O molecules decomposed into an O2 and two H2s, the H2Os were seen to "die" (die is a NetLogo primitive), thus possibly detracting from student understanding transformation of molecules and conservation of matter.

So, where a programmer would typically have H2O die, here, for the purpose of not misleading their scientific understanding of what actually happens, we created a different and not typical approach using "regroup" method. This additional layer in the program was placed to show that molecules do not "die" but rather the atoms regroup to form new molecules. In many such cases, we arrived on the decision to add additional layers on to the code to help improve code comprehension.



```
;; This method sets up the simulation.
;; It creates hydrogen, oxygen and water molecules.
;; After this is called, we can see the molecules in the screen.
to setup
  clear-all
  define-object-shapes-and-variables
  create-hydrogen hydrogen-molecules
  ask hydrogen [
    set chance-of-reaction initial-chance-of-reaction
    setxy random max-pxcor random max-pycor
  ]
  create-oxygen oxygen-molecules
  ask oxygen [
    set chance-of-reaction initial-chance-of-reaction
    setxy random max-pxcor random max-pycor
  ]
  create-water water-molecules
  ask water [
    set chance-of-reaction initial-chance-of-reaction
    setxy random max-pxcor random max-pycor
  ]
  reset-ticks
end
```

```
;; ---------------------------------------------------------------
;; When you hit the Setup button in the simulation, this method is run.
;; It creates the world of the simulation.
;; ---------------------------------------------------------------
to setup
  clear-all           ;; simulation area and graph become blank after this
  setup-world-conditions ;; set the values of the global variables
  add-h2-molecules number-of-h2-molecules   ;; make the number of h2 molecules
  add-o2-molecules number-of-o2-molecules   ;; make the number of o2 molecules
  add-h2o-molecules number-of-h2o-molecules ;; make the number of h2o molecules
  reset-ticks         ;; set time to zero
end
```

Figure 3: A change made in the code before (left) and after (right) the pilot intervention. In this case, we favored abstraction of the model to help students comprehend the top-level processes in the model.

Testing the design decision: "There are 235 lines. How does a person write down 235 lines?!" exclaimed a student after seeing the (already very simplified) code. Addition of layers of code to hide the management of the objects in a computer model (like "regroup") had bloated our code base. However, by having smaller methods that delegate tasks to other methods helped students to understand the model in some depth without demanding that they understand everything all at once.

We observed that we could not cover the entire code in the class. In addition, some of the accepted commands in NetLogo (such as "rt" (right) and "fd" (forward)) were difficult to explain during instruction. So, we simplified the code-based model in two ways (see Figure 3):

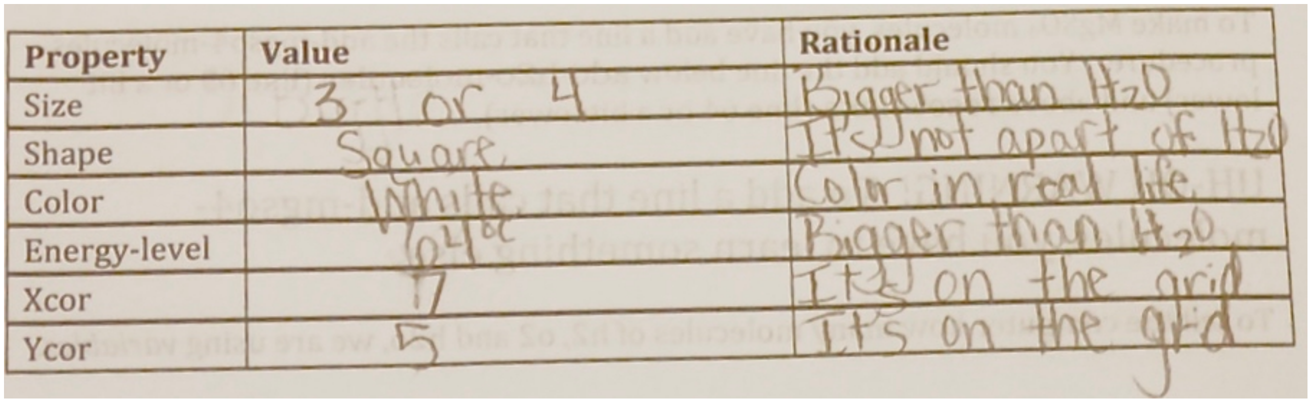
1) We created a hierarchy of processes with repetition of code if needed. Since we could not cover more than

reading the first level of code, we wrote the code such that the top-most-level method gave an overview of the science model.

2) We embedded our code with comments and method names were created such that it was easier to connect to the science topic.

Testing the design in the 7th Grade classrooms: In the classrooms, we introduced the computer simulation and model through multiple, progressive approaches. First, the students looked at the visual simulation. Then they worked on single-line commands to change the objects in the simulation in NetLogo command center. We discussed about statement evaluation and execution, and objects and their properties. Following the work on commands, we introduced students to a snippet of the code where we discussed types of objects, adding objects in the simulation, and executing methods.

The progressive introduction helped students to understand the model. One of our learning objective at the end of the 5-day intervention was to help students argue the weakness in the model and work upon one of the missing things in the model i.e., adding Epsom salt to the simulation ($MgSO_4$ molecule). They decided the properties to assign to the object and implemented it in the code (see Figure 4).



Property	Value	Rationale
Size	3 or 4	Bigger than H_2O
Shape	Square	It's not apart of H_2O
Color	White	Color in real life
Energy-level	24°C	Bigger than H_2O
Xcor	7	It's on the grid
Ycor	5	It's on the grid

Figure 4: Student-assigned properties for $MgSO_4$-molecule

The abstracted methods in the code (Figure 3 Right), was helpful for students to get started in adding objects. The students added lines in the code that were similar to the way other molecules were added such as "add-mgso4-molecules 5" which adds 5 objects to the microworld. Students had little prior experience with code but the structured, progressive presentation of the code supported students to write fairly sophisticated set of code to add an object to the model.

Conclusion

All models are wrong but some models are less wrong than others. Effective models do not strive towards similarity but are rather abstract such that a specific element can be used to explain a certain part of the phenomena (Giere, 2004). A model for instructional purposes needs to find a sweet-spot between abstraction of a phenomena and affordances to comprehend the underlying mechanism.

During the design process, pilot study and classroom intervention, we came across the need to balance between comprehension and abstraction of the scientific phenomena. However, abstraction could make it difficult for students to understand the science presented in the model or even invite alternative conceptions about the phenomena. On the other hand, introducing a detailed model could make it complex and overwhelm students. To find a balance between the level of abstraction and details to be added in the simulations and model, we have resolved to an iterative design process where we attend to the conflicts in requirements from an interdisciplinary team i.e., between different perspectives of "what is" and "what ought to be". Our design decisions were made by acknowledging the diverse perspectives, and handling those conflicts by finding trade-offs and balancing the goals.

References

Barab, S., & Squire, K. (2004). Design-Based Research: Putting a Stake in the Ground. *Journal of the Learning Sciences*, *13*(1), 1–14. https://doi.org/10.1207/s15327809jls1301_1

Bart, A. C., & Kafura, D. (2017, March). BlockPy Interactive Demo: Dual Text/Block Python Programming Environment for Guided Practice and Data Science. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 639-640). ACM.

Bart, A. C., Tilevich, E., Shaffer, C. A., & Kafura, D. (2015, October). Position paper: From interest to usefulness with BlockPy, a block-based, educational environment. In *Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE* (pp. 87-89). IEEE.

Blikstein, P., Abrahamson, D., & Wilensky, U. (2005, June). Netlogo: Where we are, where we're going. In *Proceedings of the annual meeting of Interaction Design and Children, Press*.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. *In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *Journal of the Learning Sciences*, *2*, 141-178.

Brown, A. L., & Campione, J. C. (1996). Psychological theory and the design of innovative learning environments: On procedures, principles, and systems. In L. Schauble & R. Glaser (Eds.), *Innovations in learning: New environments for education*. Mahwah, NJ: Lawrence Erlbaum Associates.

Buechley, L. (2010, June). LilyPad Arduino: rethinking the materials and cultures of educational technology. In *Proceedings of the 9th International Conference of the Learning Sciences-Volume 2* (pp. 127-128).

International Society of the Learning Sciences.

Collins, A. (1992). Toward a design science of education. In E. Scanlon & T. O'Shea (Eds.), *New directions in educational technology*. New York: Springer-Verlag.

CST (Computer Science and Telecommunications Board). (2010) *Report of a Workshop on the Scope and Nature of Computational Thinking.* National Academy Press, Washington D.C., 2010

DBRC (The Design-Based Research Collective). (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 5-8.

diSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy.* Mit Press. diSessa, A. A., & Abelson, H. (1986). Boxer: A reconstructible computational medium.

Communications of the ACM, *29*(9), 859-868.

Giere, R. N. (2004). How models are used to represent reality. *Philosophy of science*, 71(5), 742–752.

Griffiths, A. K., & Preston, K. R. (1992). Grade-12 students' misconceptions relating to fundamental characteristics of atoms and molecules. *Journal of Research in Science teaching,* 29(6), 611–628.

Grover, S., & Pea, R. D. (2013). Computational thinking in k–12 a review of the state of the field. *Educational Researcher*, 42(1), 38–43.

Guzdial, M. (1994). Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments*, 4(1), 001-044.

Harel, I. & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, 1, 1-32.

Harvey, B., Garcia, D. D., Barnes, T., Titterton, N., Miller, O., Armendariz, D., ... & Paley, J. (2014, March). Snap! (build your own blocks). In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 749-749). ACM.

Herrmann-Abell, C. F., & DeBoer, G. E. (2011). Using distractor-driven standards-based multiple-choice assessments and Rasch modeling to investigate hierarchies of chemistry misconceptions and detect structural problems with individual items. *Chemistry Education Research and Practice*, 12(2), 184-192.

Kafura, D., Bart, A. C., & Chowdhury, B. (2015, June). Design and Preliminary Results From a Computational Thinking Course. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 63-68). ACM.

Kafura, D. and Tatar, D. (2011). "Initial experience with a computational thinking course for computer science students," *Proceedings of the 42nd ACM technical symposium on Computer Science Education*, Dallas, TX, USA

Kay, A., & Goldberg, A. (1977). Personal dynamic media. *Computer*, 10(3), 31-41. Kind, V. (2004). Beyond appearances: Students' misconceptions about basic chemical ideas.

School of Education, Durham University, UK. Retrieved Sep, 25, 2009. Lovell, E., & Buechley, L. (2010, June). An e-sewing tutorial for DIY learning. In *Proceedings*

of the 9th International Conference on Interaction Design and Children (pp. 230-233). ACM.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.

Martin, F., & Resnick, M. (1993). Lego/Logo and electronic bricks: Creating a scienceland for children. In *Advanced educational technologies for mathematics and science* (pp. 61-89). Springer Berlin Heidelberg.

McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, *8*(5), 326-337.

NGSS. (2013). *Next generation science standards: For states, by states*. National Academies Press.

Özmen, H. (2004). Some Student Misconceptions in Chemistry: A Literature Review of Chemical Bonding. *Journal of Science Education and Technology*, 13(2), 147-159.

Perlis, A. (1962). The computer in the university. *Computers and the World of the Future*, 180- 219.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... &

Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67. Resnick, M., & Ocko, S. (1990). *LEGO/logo--learning through and about design*. Cambridge,

MA: Epistemology and Learning Group, MIT Media Laboratory. Resnick, M., Ocko, S., & Papert, S. (1988). LEGO, Logo, and design. *Children's Environments*

Quarterly, 14-18. Simon, H. A. (1972). Theories of bounded rationality. *Decision and organization*, 1(1), 161–176.

Tatar, D. (2007). The design tensions framework. *Human–Computer Interaction*, 22(4), 413– 451.

Tatar. D., Harrison, S., Stewart, M., Frisina, C., and Musaeus, P. (2017) Proto-computational Thinking: the Uncomfortable Underpinnings. In (P. Rich and C. Hodges, Eds.) *Emerging Research, Practice and Policy on Computational Thinking*

Tisue, S., & Wilensky, U. (2004, May). Netlogo: A simple environment for modeling complexity. In *International conference on complex systems* (Vol. 21, pp. 16-21).

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.

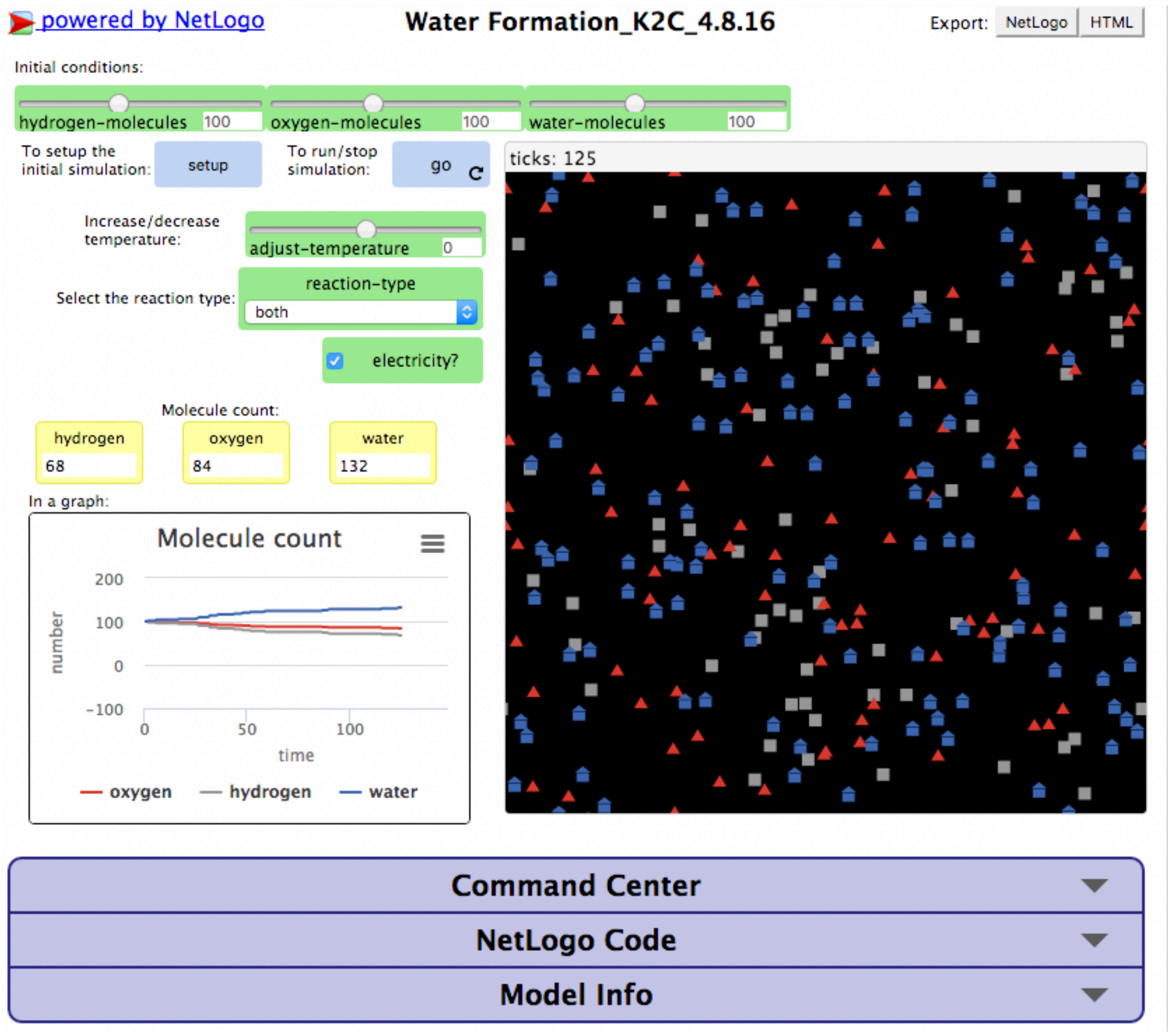
Wilensky, U. (1999). Netlogo. http://ccl.northwestern.edu/netlogo/. Northwestern University, Evanston, IL.. (Center for Connected Learning and Computer-Based Modeling)

Wilensky, U., & Stroup, W. (2002, April). Participatory Simulations: Envisioning the networked classroom as a way to support systems learning for all. In *Presented at the Annual meeting of the American Research Education Association, New Orleans, LA*.

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

Wing, J. Computational thinking and thinking about computing, (2008) *Philosophical Transactions of the Royal Society*, A, 2008, 366, pp. 3717-3725. DOI=10.1098/rsta.2008.0118

APPENDIX A: Simulation Interface


powered by NetLogo

Water Formation_K2C_4.8.16

Export: NetLogo HTML

Initial conditions:

hydrogen-molecules 100

oxygen-molecules 100

water-molecules 100

To setup the initial simulation: setup

To run/stop simulation: go

Increase/decrease temperature: adjust-temperature 0

Select the reaction type: reaction-type both

✓ electricity?

Molecule count:

hydrogen 68

oxygen 84

water 132

In a graph:

ticks: 125

Command Center

NetLogo Code

Model Info

Figure 1: Simulation interface before the fifth-grade pilot study

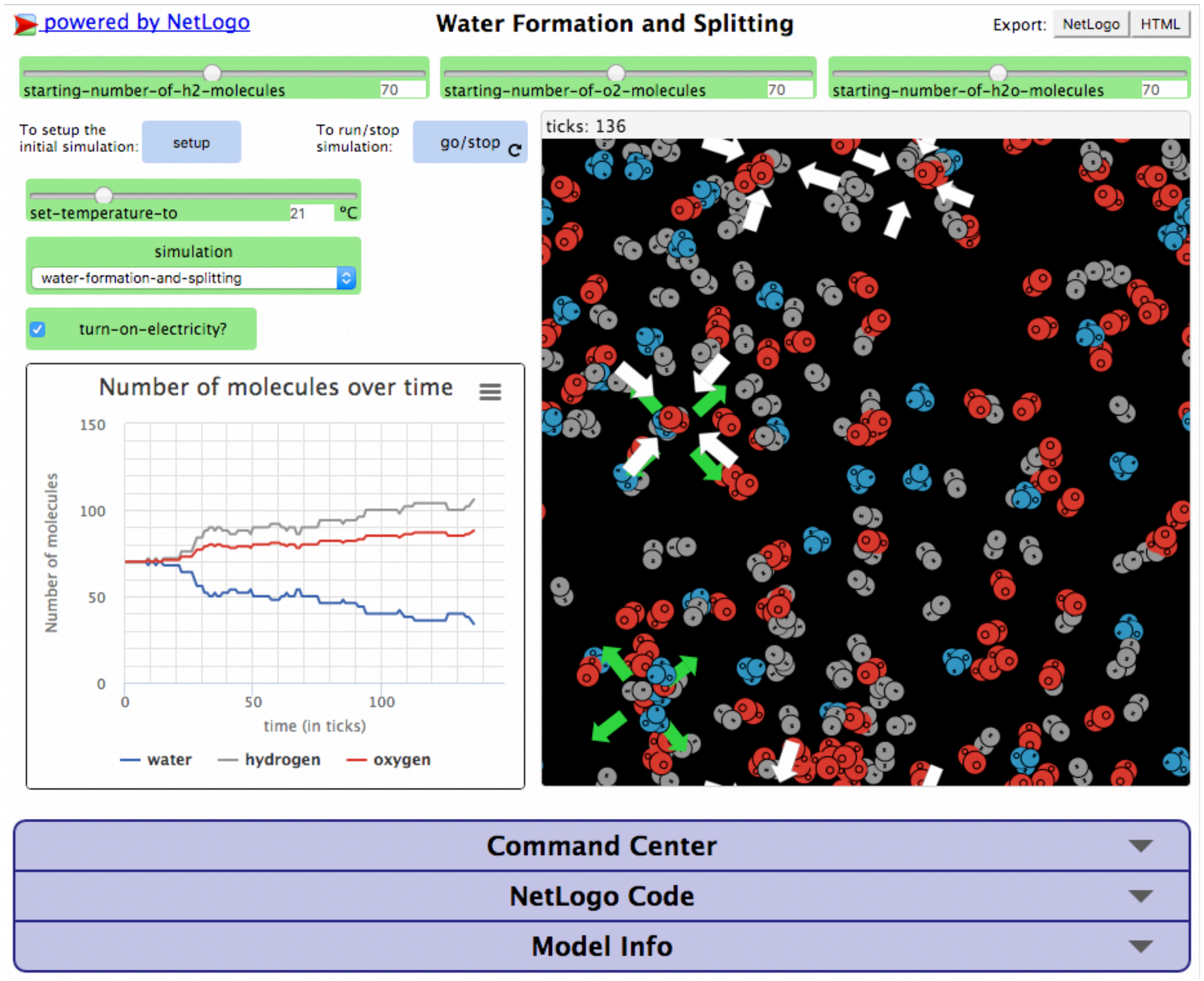

powered by NetLogo
Water Formation and Splitting
Export: NetLogo HTML
starting-number-of-h2-molecules 70
starting-number-of-o2-molecules 70
starting-number-of-h2o-molecules 70
To setup the initial simulation: setup
To run/stop simulation: go/stop
set-temperature-to 21 °C
simulation
water-formation-and-splitting
turn-on-electricity?
ticks: 136
Number of molecules over time
Number of molecules
time (in ticks)
water
hydrogen
oxygen
Command Center
NetLogo Code
Model Info

Figure 2: Simulation interface during the 7th grade classroom intervention