EARLY COMPUTING EDUCATION

INTEGRATING COMPUTATIONAL THINKING ACROSS THE K–8 CURRICULUM

Irene Lee, Fred Martin and Katie Apone

The excitement around K-12 Computing Education in the United States is rapidly increasing and K-8 holds great potential as the entry point for the integration of computing. We examine how young learners can gain early exposure and engage in rich computational experiences in K-8. These experiences can build students' computational thinking, understanding of CS concepts, programming skills and confidence as critical thinkers as well as provide experience with collecting and analyzing data. We discuss how three types of computational activities—digital storytelling, data collection and analysis, and computational science investigations—can be used to incorporate computational thinking (CT) across the curriculum.

In "Dancing with Robots" Frank Levy and Richard Murnane [12] predict the future of work in America and what it will take for the middle class to succeed. They looked into structural economic changes brought about by technology and saw that in the past three decades jobs requiring routine manual or routine cognitive skills have disappeared from the labor market. Automation, computer substitution and offshoring of human work has led to the steady decline of work centered on routine cognitive and manual tasks such as filing and assembly line work. Contemporaneously, jobs which require solving unstructured problems, communication, and non-routine manual work have grown as a proportion of the labor market. Under these circumstances, young people's preparation for the workforce must adapt. Two types of tasks that rely on uniquely human abilities are (1) the ability to integrate many kinds of information to solve unstructured problems and (2) acquiring, making sense of, and communicating information to others. Computational thinking is a key skill in the realm of solving unstructured problems, understanding and interpreting data, and communicating information to others using computers. This paper aims to help K-8 educators and the public at large understand how computational thinking can be integrated within existing curriculum through three genres of computing activities.

COMPUTATIONAL THINKING

Computational thinking (CT) is a term coined by Jeannette Wing [22] to describe a set of thinking skills, habits and approaches that are integral to solving complex problems using a computer and widely applicable in the information society. Thinking computationally draws on the concepts that are fundamental to computer science, and involves systematically and efficiently processing information and tasks. CT involves defining, understanding, and solving problems, reasoning at multiple levels of abstraction, understanding and applying automation, and analyzing the appropriateness of the abstractions made.

The International Society for Technology in Education (ISTE) and the Computer Science Teacher Association (CSTA) collaborated with leaders from higher education, industry and K-12 education to develop an operational definition of computational thinking that provides a framework and vocabulary that resonates with K-12 educators [4]. In the operational definition, computa-

tional thinking (CT) is described as a problem-solving process that includes the following.

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data.
- Representing data through abstractions (such as models and simulations).
- Automating solutions through algorithmic thinking (a series of ordered steps).
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.
- Generalizing and transferring this problem solving process to a wide variety of problems.

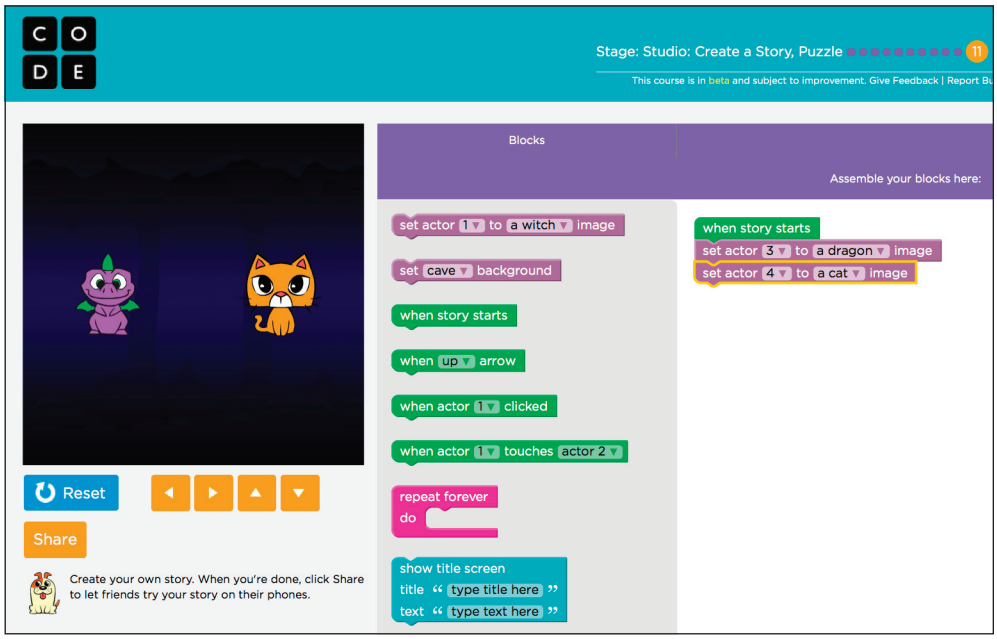

Figure 1. "Create a story" activity in Code.org K-5 curriculum.

Of particular relevance in K-8 are the understandings that CT is a way that humans think and that the ability to harness the power of computers can begin to be developed at a young age. Lee et al [11] presented examples of CT for youth in practice across three domains—modeling and simulation, robotics, and game design and development—that demonstrated that youth in K-8 were not only capable but also actively using computational thinking in out of school time activities. In this article, the authors will extend these examples to describe a set of activities and approaches that can be used integrate CT across the regular school day in core courses.

APPROACHES TO DEVELOPING YOUTHS' COMPUTATIONAL THINKING

Here we present three approaches that guide the development of youths' computational thinking. Each approach entails making shift in agency, tool use, and use of abstraction. Students progress from completing challenges posed by others to taking an active role as developers of their own designs and investigations that make use of computational tools. They learn to use computational tools and techniques in their fullness and apply the tool to solving new problems that they pose for themselves, and they leverage their experiences analyzing data that they produced before analyzing large data sets amassed by others.

"Puzzles to Open Sandbox" Approach Used in Digital Storytelling

In this approach, the learner takes the role of the apprentice and learns various facets of expert knowledge in small independent tasks or puzzles over a period of time. Only after all of the puzzles have been mastered is the apprentice allowed open access to the full studio or workbench (referred to as an "Open Sandbox" in creative computing communities) to create a project of the apprentice's choosing.

Code.org's K-5 curriculum [1] presents a series of activities to the user as puzzles of increasing difficulty within each unit. The culminating activity in each unit involves creating a story or game using any or all of the commands learned and a set of characters encountered in earlier puzzles. In this progression, the learner gains experience with simple commands while developing solutions to puzzles that were developed by someone else. Eventually, the learner as creator of a story or game poses problems that he/she will solve using computational tools (Figure 1). This type of scaffolded activity can serve as an easy on-ramp to creating artifacts in open sandbox environments with full graphical editing capabilities such as Scratch [18].

Note that in this scenario the development of computational thinking is not explicitly taught but rather evolves through one's impetus to create. Abstraction in the context of creating animated stories and games occurs when students create "costumes" or "sprites" as graphical representations of characters. Simple representations are used to represent specific people or things. Cartoonists take advantage of the brain's propensity for abstraction. "The more realistic a character appears, the more we think of them as a different person, whereas when we see a more simplified face, we see ourselves reflected in the character [15]." Furthermore, in storytelling or designing a narrative to drive a game's progression, abstraction is used in partitioning the experience into segments (chapters in stories, levels in games). Games are abstracted into a set of scenes (or levels) containing characters. Early use of abstraction involves deciding on important scenes and characters to retain, or drop, and how to get an idea across with a minimum set of scenes and/or characters. The use of abstraction to simplify a story or game to make it transferable to computational media, the use of automation or the commands to drive the unfolding of the story or behavior in the game, and the performance of simple analysis ("Do

the elements incorporated help get the idea across?" or "Is the game fun to play?") is in essence computational thinking in the creation of digital stories.

Relevant standards addressed in this approach:

- CSTA & ISTE's "Operational Definition of CT" [4] identifies key practices as formulating problems in a way that enables us to use a computer and other tools to help solve them, and automating solutions through algorithmic thinking (a series of ordered steps).
- Computer Science Teachers Association's K-12 Standards [19] addressed —in the Computational Thinking strand—are: Use the basic steps in algorithmic problem-solving to design solutions (2-1); and Describe and analyze a sequence of instructions being followed (2-6). In the Computing Practice and Programming strand, the standards addressed are: Use a variety of multimedia tools and peripherals to support personal productivity and learning throughout the curriculum (2-2); Design, develop, publish and present products using technology resources that demonstrate and communicate curricular concepts (2-3); and Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions (2-5).

Use-Modify-Create Progression Used in Computational Science Investigations

In the article "Computational Thinking for Youth in Practice" Lee *et al.* [11] described a three-stage progression for engaging youth in CT within rich computational environments. This progression, called Use-Modify-Create, describes a pattern of engagement (see Diagram 1) that was seen—in various NSF-funded Innovative Technology Experiences for Students and Teachers (ITEST) projects—to support and deepen youths' acquisition of CT. The Use-Modify-Create progression is based on the premise that scaffolding increasingly deep interactions will promote the acquisition and development of CT.

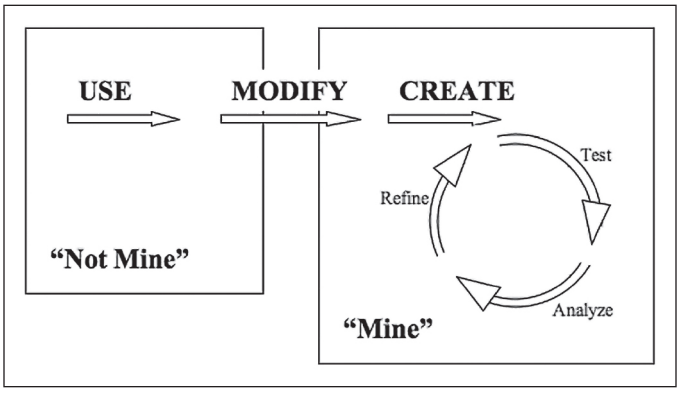

Diagram 1: Use-Modify-Create Learning Progression, from Lee et al. 2011

In the "use" stage, students are consumers of someone else's creation. For example, in the context of a computational science investigation, students run experiments using pre-existing computer models as experimental test beds. Over time they begin to "modify" the model, with increasing levels of sophistication. For example, a student may initially want to change the color of a character or some other purely visual attribute. Later the student may want to change the character's behavior in a way that entails developing new pieces of code. In this "modify" phase of the progression, an understanding of at least a subset of the abstraction and automation contained within a model is necessary. Through a series of modifications and iterative refinements, new skills and understandings are developed as what was once someone else's becomes one's own. Furthermore, as youth gain skills and confidence, they can be encouraged to develop ideas for new computational projects of their own design that address issues of their choosing. Within this "create" stage, all three key aspects of computational thinking—abstraction, automation and analysis—come into play.

Relevant standards addressed in this approach include:

- Standards related to modeling and simulation that appear in the Computer Science Teachers Association's K-12 Standards [19]. In the Computational Thinking strand, the standards addressed include: Interact with content-specific models and simulations to support learning and research (2-9); Evaluate the kinds of problems that can be solved using modeling and simulation (2-10); Analyze the degree to which a computer model accurately represents the real world (2-11); and Use modeling and simulation to represent and understand natural phenomena (3A-8). In the Computing Practice and Programming strand, the standards addressed include: Implement a problem solution in a programming environment using looping behavior, conditional statements, logic, expressions, variables and functions (2-5); and Collect and analyze data that are output from multiple runs of a computer program (2-9).
- Scientific and Engineering Practices in Achieve's Next Generation Science Standards (NGSS) [16] that include: Developing and using models (Practice #2); Planning and carrying out investigations (Practice #3); Analyzing and interpreting data (Practice #4); Using Mathematical and computational thinking (Practice #5); and Constructing explanations and designing solutions (Practice #6).

Transference of modeling and simulation as a problem-solving tool was seen in a small pilot study [10]. When asked to help their teacher study a community problem, 80% of students previously exposed to modeling and simulation in an afterschool club context suggested constructing a model of the scenario and its use as an experimental test bed to investigate and potentially solve a new community problem. It is important to note that the respondent group consisted of students in low-, mid- and high- participation rate groups—in other words, applying modeling and simulation to investigate and potentially solve local issues was a "sticky" concept. Furthermore, 50% of the student respondents could elaborate on the abstractions and automations required in the model they had in mind as an experimental test bed. This application of a computational tool to a new problem and formulation, or identification, of new problems that can be understood or investigated using computer modeling and simulation is clear evidence of computational thinking in K-8.

From Data Generation to Data Analysis Progression Used in Data Explorations

There are deep connections between CT and data, and, consequently, an important dimension of computational thinking involves students' work with data. In this third approach, learners gain first-hand knowledge with data due to their role in generating it. A strong connection can be forged between a real-world experience and the resulting data set. Ultimately, patterns in time and space can be understood in different frameworks—e.g., as a personal or socially shared memory, as captured data sets, as visualizations, and as characterized by statistics. Student data explorations can progress from (1) generating data or collecting data following activity-based instructions; to (2) organizing and analyzing data; and finally, to (3) posing unique questions and seeking appropriate data sets to analyze in seeking an answer to the question. Having a personal relationship with the data prior to working with other's data potentially engenders deeper understanding of plotted data, and possibly greater engagement with one's own data.

Students need to shift from end-users to creators and innovators capable of using computational tools and techniques to make new tools or artifacts.

Relevant standards documents refer not only to computational representations of data—e.g., bits and bytes—but the more expansive processes of gathering, representing, and using data.

- The Computer Science Teachers Association's K-12 standards discuss data in both the computational thinking (CT) and computing practice and programming (CPP) strands [19]. Specific standards include using digital tools to gather and manipulate data (CSTA CPP 1:6-10); exploring ways that data may represent text, sounds, pictures, and numbers (CSTA CT 2-7); using visual representations (CSTA CT 2-8); and analyzing data from multiple runs of a program or simulation (CSTA CPP 2-9).
- The International Society for Technology Education also discusses data in its computational thinking standards. The 2011 "Operational Definition of CT" identifies key practices with data, including "logically organizing and analyzing data" and "representing data through abstractions [4]." In the 2014 Standards for Computer Science Educators [9], the topic of data is discussed in the Knowledge of Content section, stating that learners should "effectively use, manipulate, and explain various external data stores: various types (text, images, sound, etc.), various locations (local, server, cloud), etc."
- In the high school realm, the College Board's CS Principles Big Idea #3 broadly recognizes that "data and information facilitate the creation of knowledge [3]."

Shifts Required in Computational Thinking

On the way to "applying computational tools and techniques to express themselves creatively or solve problems" student must learn to see themselves as capable of coming up with solutions. In other words, they need to shift from end-users to creators and innovators capable of using computational tools and techniques to make new tools or artifacts. This shift in agency can be instigated through project-based assignments in which students are given more autonomy in selecting the topic and deciding on the scope of their project. Another shift occurs in the set of tools used or capabilities of a tool used. In the storytelling scenario, we described using one environment, Code.org's K-5 online curriculum, for the puzzle based learning progression, followed by another, Scratch, for motivating student-driven projects with unique graphical representations. Progressively making new commands available over time within a single tool or environment was pioneered long ago by environments such as DrJava [5]. Finally, we see a shift in the use of abstraction across these three approaches. In the first approach, "puzzles to open sandbox," students start with abstractions in visual representation and progress to partitioning the storyline into segments. In the second approach, "Use-Modify-Create," they use abstraction when they select components and represent processes from the real-world in a computer model. In the third approach, "Data generation to data analysis," they use abstraction when representing real-world elements as dots on maps, or data points in graphs and charts, then use mathematical abstraction as they describe a set using statistical characteristics.

INTEGRATING COMPUTATIONAL THINKING ACROSS K-8

Three genres of computational activities (digital stories, computer models and simulations, and data explorations) can be used to incorporate CT across the K-8 curriculum. In this section, we present three examples of work with middle school learners: (1) using Scratch for digital storytelling in language arts and history classes; (2) using StarLogo for computational science investigations in the context of science classes; and (3) using iSENSE for data collection and analysis for learning about time, rate, and distance.

Digital Storytelling

Scratch [18] has been used by students to create animated stories and reports to fulfill class assignments in various subject areas in K-8. The example below shows how a student used Scratch to create a digital book report on E. B. White's story "Charlotte's Web" [14]. In this retelling of the story, three main scenes were retained and the story progresses through the dialog of the key characters. The author remarks in the notes and credits window "This is for a book report for school. I slightly changed the story to keep the main points but make the extent of the video not too long. I made some minor spelling changes because my teacher asked me to."

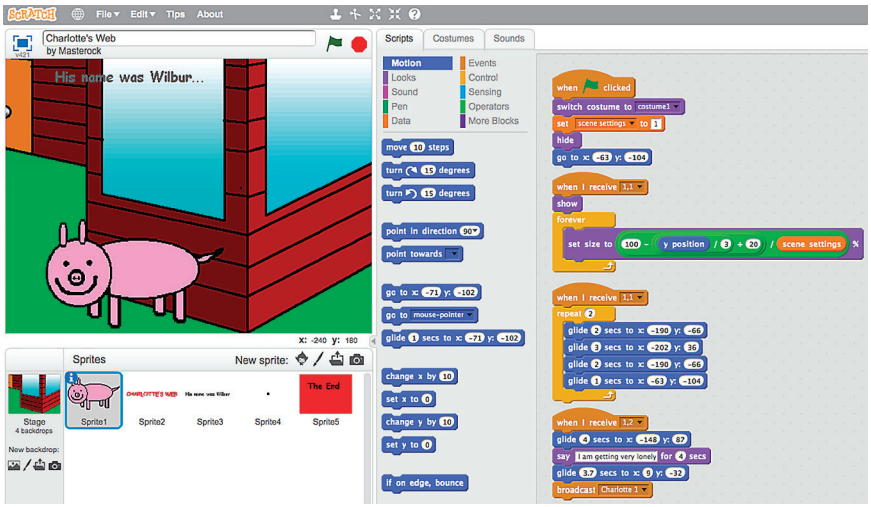

Figure 2. Screen capture from "Charlotte's Web" project in Scratch.

"Why Government? Short animation for my History class" is another example of a class assignment shared by a student [6]. The author remarked "Made this for my history class @ school thanks to Mr. H for letting me use SCRATCH!" In this example, loops and event handling were used to advance the narrative over a sound track.

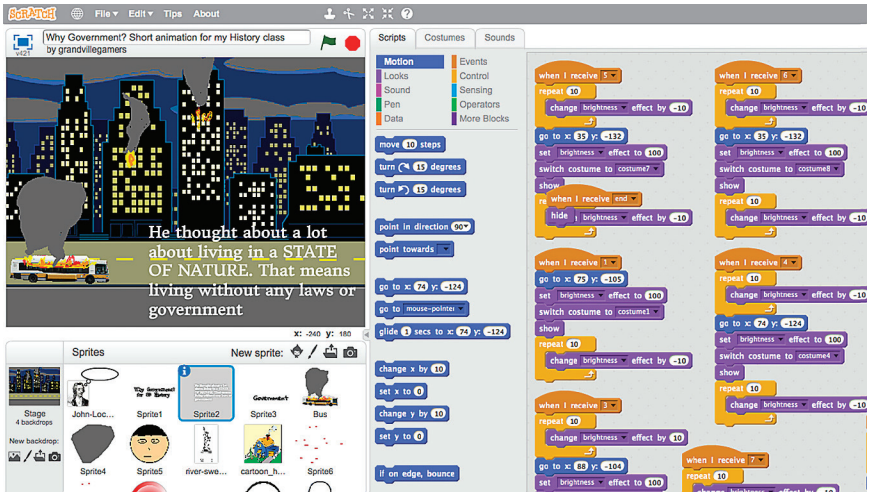

Figure 3. Screen capture from "Why Government? Short animation for my History class" in the Scratch environment.

In these examples, students used abstraction to simplify a story and make it portable to computational media. They used automation when assembling the commands that drive the unfolding of the story and they performed simple analysis when determining whether the elements they incorporated make the story interesting and/or informative to others. The combination of ownership, agency and accomplishment is evident in these projects.

Computational Science Investigations

Mathematical and scientific concepts are better suited for investigation using computer models and simulations than illustrated using animated stories. Agent-based modeling environments, specifically StarLogo TNG [20] and StarLogo Nova [21] have been used with 6-8th grade students in science classes to delve deeply into science concepts through modeling and simulation. In the context of a life science unit on ecosystems, the Use-Modify-Create progression provides a scaffolded approach to engage students in CT within modeling and simulation environments. After an initial description of the model is presented to the student, the progression can guide the student to deeper levels of understanding of the model and the phenomena being modeled. For example, given a simple ecosystem model, the student viewing the underlying code sees that the world is made up of fish and plankton that move through the pond with a little bit of randomness in their motion. Fish lose energy when they move. When a fish encounters a plankton, the plankton gets eaten and the fish gains energy. When the fish's energy has reached a threshold, it gives birth to a new fish and gives the new fish some of its energy.

In the "use" stage, students run experiments on this pre-built computer model. In this example, students are asked to find an initial number of fish that leads to a pond ecosystem where both fish and plankton can persist over time. By changing initial number of fish through a user interface widget and running repeated experiments, students might discover that there are three possible outcomes: fish die out but the plankton survive; plankton get all eaten then the fish die out; and that the populations in the ecosystem persist in dynamic equilibrium. (See Figure 4.)

Over time, students ask questions and modify the model with increasing levels of sophistication. For example, students initially want to see if changing the color of the plankton impacts the system's behavior. Later the students add a predator and see its impact on the simple ecosystem (Figure 4). This addition entails using CT when developing a new agent and algorithms for its behavior. Subsequent experiments uncover a new pattern in the system, with the addition of a top predator, sometimes the fish population never grows above a certain threshold and the cycles of population booms and busts cease to exist. In this way, through a series of modifications and iterative refinements, new skills and new scientific understandings are developed as what was once someone else's model becomes one's own. A Project GUTS ecosystem unit incorporating this activity has been implemented and used successfully with diverse populations of middle school students both in the context of afterschool clubs and within regular school day mandatory classes over the past five years.

Recently, a set of StarLogo Nova modeling and simulation modules have also been developed by Project GUTS [17] for used in Earth, Life, and Physical Science contexts [2].

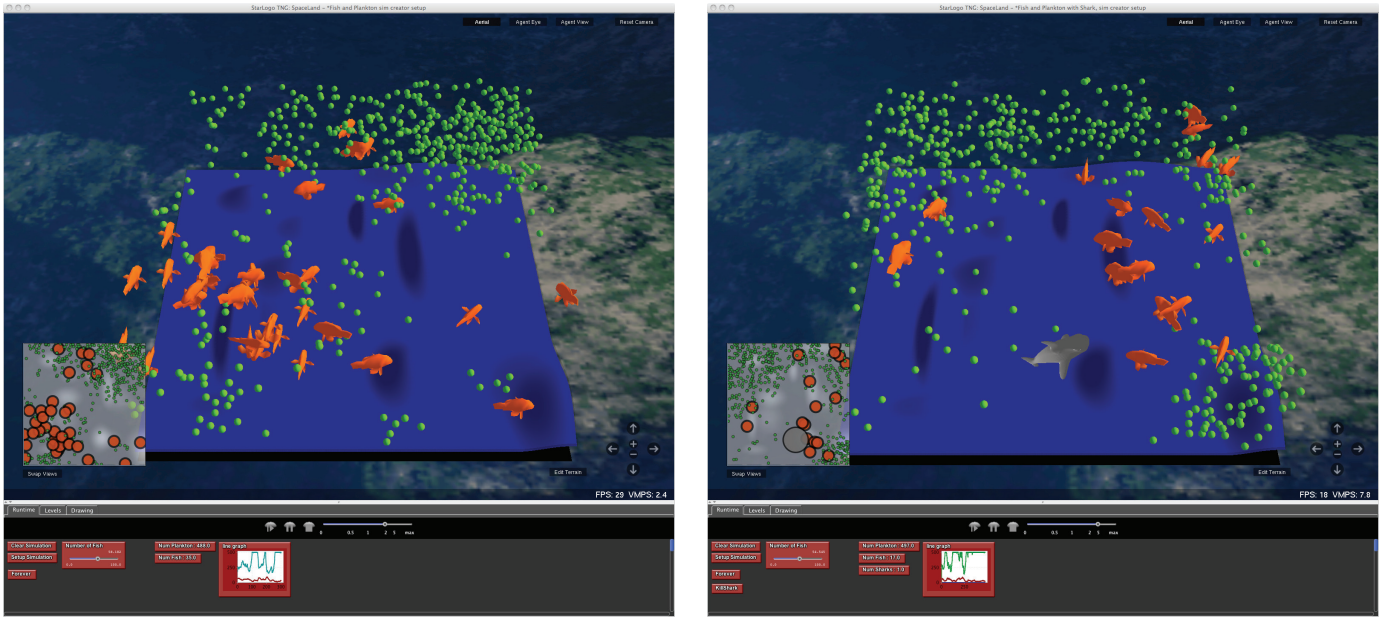

Figure 4. StarLogo TNG ecosystem simulation with and without predator.

Computational Thinking with Data

iSENSE [8] is a set of web-based and mobile tools that enables sharing, collaboration, and visualization of real-world data sets [13]. At the iSENSE web site, teachers (and students) can author "projects," which are structured repositories of data, usually for a specific investigation or purpose. Then students can contribute individual data sets (each of which may contain single or multiple rows of data). Once data sets are present, students and teachers can select them and visualize them, using a variety of tools, including a map view, time series view, and summarizing bar chart view.

Data for analysis on iSENSE may come from a variety of sources. Users may directly type data into a web page, or upload tabular data in Excel or CSV form. Data saved from classroom experiments using probeware may be uploaded. Also, data sets published on the web may be downloaded and then uploaded to iSENSE for sharing and visualization. Custom mobile apps are available that use a device's sensors—its GPS and accelerometer—to record data and directly upload it to iSENSE. One of these apps is the Data Walk, which is available for Android devices. This app records the user's position, velocity, and instantaneous acceleration at an adjustable rate (e.g. every second, or every 10 seconds). After a short period of data capture, the data set is uploaded to the iSENSE web site for inspection and sharing [7].

In collaboration with two Boston-area teachers (one high school teacher and one middle school teacher), the iSENSE team developed a classroom unit called the "Human Tickertape." The activity is modeled after earlier paper tape experiments created by physics teachers, in which an apparatus marks a paper tape at regular timed intervals. As the paper tape is unfurled (e.g. by an accelerating mass), students can see the distance between the dots grow, introducing them to velocity and acceleration. In the Human Tickertape version, students carry a mobile device running the iSENSE data walk app. It's an outdoor activity, and, as students go on a walk (or run), the data walk app records their position and velocity. Afterward, the data are uploaded to iSENSE, and students view themselves and each other on a map, and interpret their data using other visualizations.

Figure 5 shows an example of the resulting data. The data sets are from students of a middle school science teacher who carried out the activity using the tennis courts behind her school. Students were organized in teams and given directions for collecting a 30-second sample of data—e.g., walk slowly for 30 seconds or run for 30 seconds. The maps show data from two teams of students,

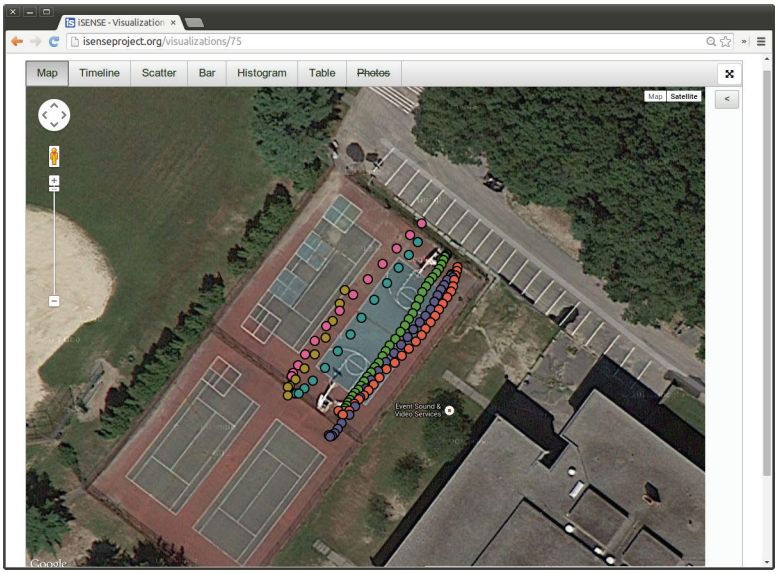

Figure 5. Map visualization of six students' data from Human Tickertape activity.

each with three participants. The teams did their work simultaneously, but in different parts of the school's tennis courts. When the students viewed their data, they saw that the group using the upper-left portion of the tennis courts (farther from the school) was running, while the other group of students walked. In the upper group, the distance between the map markers is larger, indicating that the three students in this group were moving faster. Students were able to use their own body memory of performing the activity while interpreting the visualization.

Then students explored other views of the same data. For instance, the timeline view shows the six data walks using a velocity-vs.-time graph (Figure 6). Students were able to see that three of them were walking (reaching a maximum velocity between 1 and 2 meters per second) while the other three were running (reaching about 3.5 meters per second). The three data sets from the "running group" show the quick acceleration to running speed terminated by a hard stop. The three data sets from the "walking group" show an approximately constant speed for the recording period.

Figure 7 presents yet another view of the same data, summarizing each student's walk as a single bar chart of average velocity. This example in iSENSE highlights several important computational thinking practices with data. Students are using computational tools to "gather and manipulate data," "use visual representations," and analyze "multiple runs" of data. Furthermore, because the iSENSE system puts all of the students' data into the same project container, it allows them to share and compare data in a facile way that's afforded specifically by the computational medium. It's also worth reflecting upon the idea of abstraction as it exists in these data explorations. Typically we discuss abstraction in the context of programming—for example, procedural abstraction or object-oriented abstraction. In this fashion, abstraction is about hiding information, yet simultaneously revealing the essentials of the matter.

In the progression of iSENSE visualizations, we can also see this principle in action. The map view abstracted the students' real world motion to set of points on the map, showing students where they were, and also representing velocity as the interval between the locations. The velocity timeline abstracted away the position information, but highlighted the acceleration and deceleration in each student's walk. The velocity bar graph collapsed the timeline into a single point of information—but more easily allowed comparisons among students' walks. While this example came from a science class, these approaches are also valuable in mathematics. Teachers working with iSENSE have used it in math classes to help students understand probability (e.g., having many students contribute dice rolls or other data from other random events to a project, and learning about the role of a large sample size) and proportional reasoning (e.g., comparing the length of one's forearm and tibia for people of different heights).

Figure 6. Timeline visualization of students' data walks, showing velocity vs. time. Some of the data are overlaid because students were simultaneously recording their walks on two different devices; the graph shows all of their data together, regardless of which device initially recorded it.

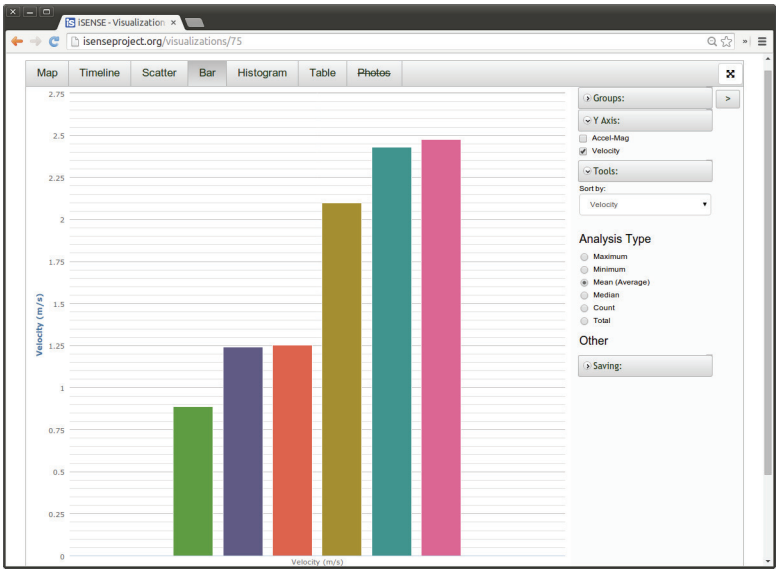

Figure 7. The six data walks, presented as bar graphs of average velocity per walk. The data sets are color coded to match across the visualization tools.

CONCLUSION

Based on our experiences with youth learning CT, we offer concrete examples of activities that integrate into regular school day classes in K-8 to support the development of computational thinking. The tools and approaches presented in this article can be used in conjunction with project work across a variety of content areas. While implementing CT across the curriculum during the regular school day is a compelling vision, there remain substantial challenges to its implementation in some schools. These barriers include lack of space in the existing curriculum for computationally rich project work, lack of opportunities for teachers to learn CT as

part of their professional development, and lack of access to necessary infrastructure. Nonetheless, there are other teachers who are currently, or may become, well positioned to integrate CT in their classrooms. It is our hope that by describing some feasible approaches, these teachers, and potentially many others, may benefit from these practical approaches to embedding CT in K-8 that are both grounded in experience and linked to relevant standards. Ir

ACKNOWLEDGMENTS

Co-author Martin wishes to thank public school teachers Steve Cogger and Barbara Keating for their work on the Human Tickertape activity. iSENSE Project material is based upon work supported by the National Science Foundation under grant numbers IIS-1123972 and IIS-1123998. Co-author Lee wishes to thank Project GUTS club leaders and facilitators for their dedication to bringing computational science experiences to middle school students, and the MIT StarLogo Development Group for their ongoing support and partnership. The modeling and simulation material presented in this article is based upon work supported by the National Science Foundation under grant numbers DRL-0639637 and DRL-1031421.

REFERENCES

[1] Code.org K-5 Curriculum; http://learn.code.org/s/course1; http://learn.code.org/s/course2; http://learn.code.org/s/course3. Accessed 2014 July 30.
[2] Code.org / Project GUTS Middle School Computer Science in Science Curriculum; http://code.org/curriculum/mss. Accessed 2014 July 30.
[3] College Board. *AP Computer Science: Principles Course Annotations.* 2010; http://www.collegeboard.com/prod_downloads/computerscience/1_2010_AP_CS_Principles_Annotations_0929.pdf. Accessed 2014 July 30.
[4] CSTA & ISTE. *Operational Definition of Computational Thinking for K–12 Education.* 2011; http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf. Accessed 2014 July 30.
[5] DrJava. www.drjava.org; Accessed 2014 July 30.
[6] grandvillegamers. "Why Government? Short animation for my History class;" http://scratch.mit.edu/projects/12171399/. Accessed 2014 July 30.
[7] iSENSE Data Walk mobile application; https://play.google.com/store/apps/details?id=edu.uml.cs.isense.datawalk_v2 . Accessed 2014 July 30.
[8] iSENSE project website; http://isenseproject.org. Accessed 2014 July 30.
[9] ISTE Standards for Computer Science Educators. 2014; http://www.iste.org/docs/pdfs/2014_ISTE_Standards-CSE_PDF.pdf. Accessed 2014 July 30.
[10] Lee, I. "Assessing Youth's Computational Thinking in the context of Modeling & Simulation." Presented at *American Educational Research Associates* (AERA) Conference Proceedings. (New Orleans, Louisiana, 2011).
[11] Lee, I., et al. "Computational Thinking for Youth in Practice." *ACM Inroads*, 2, 1(2011): 32-37.
[12] Levy, F., and Murnane, R. "Dancing with Robots: Human Skills for Computerized Work;" http://www.thirdway.org/publications/714. Accessed 2014 July 30.
[13] Martin, F., et al. "iSENSE: A Web Environment and Hardware Platform for Data Sharing and Citizen Science." Presented at the *Association for the Advancement of Artificial Intelligence 2010 Spring Symposium.* (Stanford, CA, 2010).
[14] Masterock. "Charlotte's Web." (Scratch project); http://scratch.mit.edu/projects/112192/. Accessed 2014 July 30.
[15] McCloud, S. *Understanding Comics: [The Invisible Art].* (New York: HarperPerennial, 1994).
[16] NGSS Lead States. *Next Generation Science Standards: For states, by states.* (Washington, DC: National Academies Press. 2013).
[17] Project GUTS website; http://projectguts.org. Accessed 2014 July 30.
[18] Scratch website; http://scratch.mit.edu. Accessed 2014 July 30.
[19] Seehorn D., et al. *CSTA K-12 Computer Science Standards.* (New York: ACM publications, 2011).
[20] StarLogo Nova website; http://slnova.org. Accessed 2014 July 30.
[21] StarLogo TNG website; http://education.mit.edu/projects/starlogo-tng. Accessed 2014 July 30.
[22] Wing, J. M. "Computational Thinking," *Communications of the ACM*, 49, 3 (2006): 33-35.

IRENE LEE
Santa Fe Institute, 1399 Hyde Park Road,
Santa Fe, New Mexico 87505 USA
lee@santafe.edu

FRED MARTIN
University of Massachusetts Lowell, 1 University Avenue,
Lowell, Massachusetts 01854 USA
fredm@cs.uml.edu

KATIE APONE
Code.org, 1301 5th Avenue, Suite 1225,
Seattle, Washington 98101 USA
katie@code.org

Categories and Subject Descriptors: K.3.2 [Computers and Education]: Computer and Information Science Education – *Computer science education, curricula, literacy.*
General Terms: Human Factors, Performance, Design, Experimentation.
Keywords: Computer science education, computational thinking, abstraction, automation, analysis.

DOI: 10.1145/2684721.2684736