Computational Thinking for Physics: Programming Models of Physics Phenomenon in Elementary School

Hilary Dwyer*, Bryce Boe†, Charlotte Hill†, Diana Franklin†, & Danielle Harlow*

*Gevirtz Graduate School of Education, University of California, Santa Barbara, CA 93106-9490
†Department of Computer Science, University of California, Santa Barbara, CA 93106-5110

Abstract. Computational thinking, an approach to problem solving, is a key practice of science education rarely integrated into instruction in an authentic way. A second key practice, creating models of physical phenomenon, has been recognized as an important strategy for facilitating students' deeper understandings of both science concepts and the practices of science. We are creating an interdisciplinary computational thinking curriculum for grades 4-6 that combines the development of computational thinking with content in other disciplines such as science. Here we present an example project where students can iteratively develop a model to explain the momentum and acceleration of an object, coupled with sophisticated computational thinking concepts to simulate that model. In addition, we present two findings from related research on fourth graders' pre-instructional knowledge related to computational thinking: 1) Students recognized the need for but struggled to produce specific instructions, and 2) Students understood that small errors could change outcomes.

Keywords: Physics Education Research, Computational Thinking, Elementary School Education.
PACS: 1.40.-d, 1.40.eg, 1.40.Fk, 1.40.G-, 1.50.H-

INTRODUCTION

The Next Generation Science Standards [1] integrates science content with eight key practices of science and engineering. The intention is K-12 students will be expected to not only learn *about* these practices, but also learn science content *through* these practices. Two practices key to this project are (1) Mathematical and Computational Thinking and (2) Developing and Using Models.

The Computer Science Teachers Association (CSTA) defines computational thinking as "an approach to solving problems that can be implemented with a computer. … Computational thinking includes "abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts" [2]. Computational thinking, however, is not limited to work on a computer. It is a method of problem solving useful across disciplines, including physics.

In this project, we are developing a set of activities for children in grades 4-6 that will support students' science learning by developing and using models *while* developing computational thinking. Along with this, we are developing an empirically-based learning progression for computational thinking.

We have long recognized that students come into instruction with many ideas about science from their everyday interactions with the real world [3]. For example, we know that young children articulate that things slow down because "force runs out" [4], an idea counter to the accepted scientific explanation but consistent with their everyday observations. To help children develop understandings aligned with scientific ideas, we need to be cognizant of children's existing interpretations of their observations.

Thus, to help children develop computational thinking, we must be aware of their existing ideas and related experiences. Unfortunately, we know very little about children's notions and skills related to computational thinking [5]. Thus, as a first step toward developing curriculum, we are investigating children's notions about computational thinking developed prior to formal instruction in this area.

Here, we present an example activity that engages students in developing and using a model of a physics phenomenon and discuss computational thinking. We then describe our research design and present initial findings about fourth grade students' pre-instructional ideas related to three aspects of computational thinking.

MODELING, COMPUTATIONAL THINKING, AND PROGRAMMING

As an example activity, consider students engaged in dropping objects of different masses onto a seesaw that then projects another object up. After making observations of such real life objects in a lab setting,

2013 PERC Proceedings, edited by Engelhardt, Churukian, and Jones; Peer-reviewed, doi:10.1119/perc.2013.pr.021

students could develop a computer program (or simulation) that models the behavior of the launched object in response to the mass of the dropped object. To create a program which mimics the real life observations, a student would need to consider whether and how the mass of the falling object was related to the rate at which the object fell, the maximum height of the launched object, as well as whether the deceleration of the launched object as it traveled upwards is the same as the acceleration of the object as it falls back down, and other factors. The student could then write a program that would animate objects to match their observations.

While programming activities have been used to assess students' understanding of science ideas [6] and help students model science phenomenon [7,8], the difficulty of programming can limit its usefulness [9]. Several graphical programming interfaces have been developed that allow novices to more easily engage in authentic programming and computational thinking activities. Scratch [10] is one such graphical programming interface that lowers the cognitive barriers to programming by removing the possibility of making syntax errors. When programming in Scratch, users select blocks and drag them onto the programming area, creating scripts associated with agents (called sprites) and the background.

Returning to the example activity described above, Figure 1 shows three screen shots at sequential time points of a possible animation using the Scratch programming environment showing a rock falling on a seesaw and projecting a person up. Three rocks of different masses can be dropped on the seesaw. The left column shows sample scripts (programs) that a student might write to animate the man. This script shows that the man would travel upwards to different heights (different values of y) depending on the rock. However, as this program is written, the man moves at a constant velocity. Gravity is not accounted for. In this way the animation models some aspects of observations but is limited in others.

In such an activity, students must first develop a conceptual model of the physical phenomenon and identify all relevant variables and the relationship among these variables. They then must translate these ideas into a computer simulation using precise language. Observing the results of their simulation allows them to assess how well their model matches reality and iterate their model until it is a reasonable representation. This process involves two key concepts in computational thinking: abstraction (identifying only relevant variables) and programming (using sequential execution and message passing).

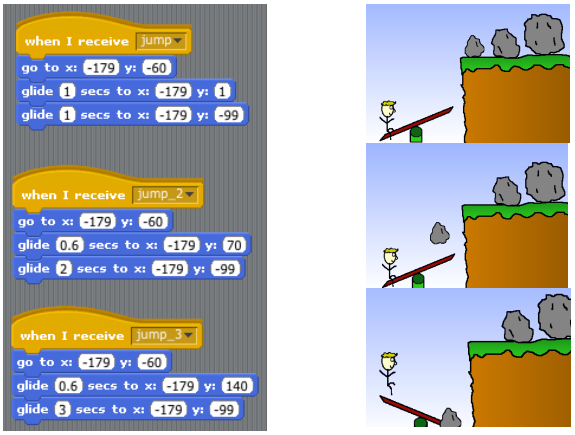

FIGURE 1. Left shows sample script for one sprite (the projected man) of an animation. The right shows three sequential screenshots of the stage of an animation.

RESEARCH DESIGN

The results we report here are part of a larger project designed to develop a learning progression about computational thinking for grades 4-6. This learning progression will inform our curriculum development. A learning progression consists of a lower anchor describing learners' pre-instructional ideas, an upper anchor point describing goals, and several intermediate levels.

Hypothesized Learning Progression. We began with a hypothesized learning progression of how ideas would develop over time with appropriate curriculum. This learning progression will be iteratively revised as the project progresses. Our first stage was to test the lower anchor point, i.e., those ideas we expected fourth graders to develop prior to formal instruction.

Our hypothesized learning progression included six strands aligned with aspects of computational thinking described by the CSTA: Abstractions, Knowledge about algorithms, Algorithms, Data, Knowledge about how computers work, and Programming. For each strand we proposed three to eight smaller ideas of incrementally higher sophistication. These were based on the CSTA standards [2]. The lowest level for the Algorithms (Al) and Knowledge of how computers work (KC), were,

> Al-1: Develop one algorithm - Create step-by-step instructions for an age-appropriate task.
>
> KC-1: Computers have a very limited vocabulary, so you have to learn how to tell them what to do.

These ideas are both important not only for the strands they are listed in, but also for the programming strand. The lowest idea in programming is

> P-1: Create step-by-step instructions for an age-appropriate task for someone who is younger than you who has limited vocabulary.

P-1 builds on Al-1 and KC-1 because it includes both the idea of step-by step instructions (Al-1) *and* the idea of limited vocabulary (KC-1).

Data Collection and Analysis. We conducted fifteen focus group interviews with fourth graders (N=55) at four elementary schools in California during May 2013. To obtain a representative sample, we selected participants from schools with a wide range of academic performance and backgrounds among students. We interviewed all students whose parents returned signed consent forms. The percentage of English Language Learners (ELL) ranged from 22% to 81% and students receiving free or reduced lunch ranged from 29% to 100%.

Each focus group interview lasted approximately 30 minutes and was filmed. The interviews included three topics (knowledge about computers, complex decisions, and sequential procedures), and ended with one of four activities designed to elicit ideas related to aspects of computational thinking.

Here we focus on the six focus group interviews in which the students (N=23) participated in the *drawing* activity. The drawing activity was based on an activity in CS Unplugged called *Marching Orders* [11], where students were given a picture and then asked to give instructions to another student, who was to draw the picture based on the directions.

We adapted the activity in the following ways. First, we did the activity in three rounds, each progressively more difficult. In the first round, the picture was simple: a circle, square, and triangle vertically arranged and touching. In the second, the picture was a square with lines bisecting different sections. In the final round, the picture included shapes and wavy lines, haphazardly organized. We also adapted the activity so that students participated in different ways during each round. In the first two rounds, one student gave directions to the other students. In the third round, students worked together and created directions for the interviewer, who would draw a picture based on their description.

The discussion from this activity in the six groups was transcribed, and we assigned pseudonyms to all participants and schools. We analyzed these segments for discourse that supported or challenged our proposed model. Text was coded by its connection to the hypothesized learning progressions.

FINDINGS

Finding 1: Fourth graders recognized the need for specific instructions but struggled to produce them.

As described above, we hypothesized that fourth graders could create step-by-step instructions. This idea is related to two areas of computational thinking: algorithms and programming. Participants recognized when others' instructions were vague and provided suggestions to amend the instructions, but they struggled to produce specific instructions themselves. In the drawing activity, many students described shapes and explained directions with their hands or metaphors; however, they did not include details such as location, position, size, or pattern required to enable someone else to reproduce the drawing.

For example, in the first round of the drawing activity, students were asked to give directions for a simple picture: a circle, square, and triangle where each shape abuts another vertically. José described the picture as a series of shapes, but did not specify size or position of the shapes. This information was important to include because the triangle could be oriented in a number of ways and the shapes could be spaced apart.

> José: There's a circle on the top.
> Interviewer: So we're going to draw a circle on top.
> José: There's a triangle – there's a square in the middle. There's a triangle at the bottom.

This type of description was common. Fourth graders easily identified shapes and described how the shapes were aligned with respect to each other, but often left out crucial information such as size or position. For instance, José did not describe whether the shapes were touching, the triangle's orientation, or the size of the shapes.

In terms of receiving directions, the fourth graders did replicate pictures from the directions of their peers in all three rounds. When asked, they also recognized how the directions could be improved to more accurately replicate the picture. For instance, Victor and Angel shared that José could have specified that the shapes "connected."

> Interviewer: What are some things that José could have said that might have made your pictures look more like [the original] picture?
> Victor: To connect them, connect the shapes.
> Interviewer: How would you have said it then?
> Victor: There's a circle, a square, and a triangle.
> Angel: [overlap] Between the middle and the tops.

Here, Victor described that had José included more information such as connecting the shapes, he could have drawn the picture more accurately.

That fourth graders lacked specificity when giving direction was an important preliminary finding. Recognizing the need for and critique of step-by-step instructions appeared to be an anchor point below our lowest anchor point in two strands of our hypothesized learning progression: algorithms and programming (Al-1 and P-1).

Finding 2: Fourth graders recognized that small errors could change outcomes.

We hypothesized that fourth graders would know that computers have a very limited vocabulary. Preliminary results suggested that while our participants were *not* aware of computers having a limited vocabulary, they did relate instances where small deviances from what was "allowed" by a program resulted in errors. From this, we inferred that fourth graders knew that they must communicate in specific ways for a computer (or program) to work properly.

For example, Jess recognized that in a typing program used at her school, the difference between "j" and "J" resulted in different outcomes:

> Interviewer: So the whole point [of the drawing activity we just did] is that when we're playing with computers and we're typing, the computer needs really specific directions right?
> Jess: Right, like I – I put like – in Mavis Beacon even if like put j-k-j and it was supposed to be J-capital J.

Jess connected a statement about how computers need specific directions to mistakes she had made with a typing program.

In a second example, Henry came to a similar conclusion. After the interviewer explained that the purpose of the drawing activity was to demonstrate that computers need specific directions, Henry noted that even small errors such as "one wrong number" might change the outcome.

> Interviewer: So the reason that we're doing this is because when you're working with computers, you have to give really precise directions right?
> Henry: Oh! Computer programmer?
> Interviewer: And it's actually really, really hard –
> Henry: You do like one wrong number – you do 1 instead of 0 the whole thing [is wrong]

Just as Jess recognized the importance of distinguishing "J" and "j" for the typing program, Henry recognized that for computer programmers if they use "1" instead of "0", the computer may not do what the user expects.

These two examples suggested that some fourth graders enter the classroom with a knowledge that people need to use specific language to communicate with computers; however, they did not suggest that computers have a limited vocabulary. This was a subtle but important difference. While the students did not volunteer that computers had their own language, they did recognize that there were ineffective ways to communicate. Thus, preliminary evidence suggested that students understand that small errors (a single character) could change the outcome, an idea below the lowest anchor point of the strand Understanding how computers work in our hypothesized learning progression (CW-1).

DISCUSSION AND IMPLICATIONS

Children develop ideas relevant to learning computational thinking through their rich everyday experiences engaging in problem solving and decision making as well as interacting with computers and media. Understanding their existing ideas will help us build meaningful and relevant curriculum that integrates computational thinking with other content areas. We think that using computational thinking and programming as tools for students to develop, use, and iterate models of science phenomenon they observe is a particularly powerful method of learning physics.

Our preliminary findings described students' computational thinking prior to instruction. Both findings presented here allowed us to describe anchor points lower than those in our proposed learning progression. This will contribute to our first revision of the learning progression. Also, our findings have implications for how we design the curriculum. Our activities will be developed with this understanding of students' prior knowledge.

As our project progresses we will collect data to understand how elementary students' computational thinking develops over several years of instruction and how such thinking co-develops with science concepts, and ideas about building and using science models.

ACKNOWLEDGMENTS

This research was supported by NSF grants CNS-1240985 and CNS-0940491. We thank the participating teachers and students.

REFERENCES

1. Next Generation Science Standards, available online at nextgenscience.org
2. Computer Science Task Force, CSTA K-12 Computer Science Standards, 2011.
3. R. Driver *Children's Ideas in Science* (Open University Press, Milton Keynes, UK, 1985).
4. M. McCloskey, "Naive theories of motion." *Mental models* edited by D. Gentner and A. Stevens (Lawrence Earlbaum Associates, Inc, Melville, J, 1983): 299-324
5. S. Grover & R. Pea. *Educational Researcher*, 42, 38-42 (2013).
6. A. Baytak and S. Land, *Educational Technology Research and Development*, 59(6), 765-782 (2011).
7. A. Dickes and P. Sengupta, *Research in Science Education*, 43(3), 921-953 (2012)
8. B. Sherin, A. diSessa, and D. Hammer, *Interactive Learning Environments*, 3(2), 91-118, (1993).
9. M. Guzdial, *Interactive Learning Environments*, 4(1), 1-44 (1994).
10. Scratch.mit.edu
11. Computer Science Unplugged http://csunplugged.org/