Adding computational thinking to your science lesson: what should it look like?

Jennifer Albert
The Citadel
171 Moultrie St.
Charleston, SC 29409
(843) 953-7121
jalbert@citadel.edu

ABSTRACT
In a promotional video recently released by code.org (2013), Will.i.am states "great coders are today's rock stars". However, introducing computational thinking into K-12 curricula has been a long and slow process. This paper describes efforts to develop computational thinking activities that can be easily implemented in any science classroom. Studies have shown that a set of conditions must be met for computational thinking tools to be used in K-12 education and that when they are used, there is a wide spectrum in the level of computational thinking that the tool enables. This study extends this work by examining how middle school students translate their science fair projects in Scratch and what evidence of computational thinking is present. Overall, it was found that most students simply created a presentation of their project without much complexity. They did not use more complex features such as loops or conditionals. Eight students created interactive projects that required user participation and used more advanced computational concepts. Finally, recommendations are given for next steps in the creation of a series of activities that would scaffold student learning as they applied computational thinking concepts to a science concept. These activities will help science teacher educators better prepare science teachers to address computational thinking in their classrooms.

Keywords
Computational Thinking, Next Generation Science Standards, Middle School Science Education

INTRODUCTION

Introducing computational thinking into K-12 curricula has been a long and slow process. However, like in many classes where technology usage has been introduced to teach both technical literacy and the subject matter, there have been a few instances of science classes using introductory programming environments like Scratch to get students to create projects to demonstrate knowledge of scientific concepts. Ideally, this would encourage students to learn concepts such as loops, variables, and abstraction, but in at least one instance, only a bit more than half of the students went any farther than simply making animations in the environment. The authors of the Next Generation Science Standards [1] must have thought these concepts important when making mathematical and computational thinking one of the key science and engineering practices. For example, HS-ETS1-4 states "use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem"[1]. However, not all modes of computational thinking are created equal or are feasible for the typical high school science classroom.

In this paper, we describe what we discovered by analyzing a selection of 29 programs written by students and uploaded to the Scratch online community for a science project. Nearly 14 of the

Paper presented at the annual meeting of the 2016 *Association for Science Teacher Education Conference*, Reno, Nevada.

submissions were hard-coded animations, and the others either involved user interaction or some kind of randomization that demonstrated that the student went beyond the surface of the environment. We make the claim that for students to use Scratch as more than just an animation tool and incorporate the computational thinking concepts built into the environment, scaffolding is suggested. Students use the functionality in the language that corresponds to constructs in programming languages with few scaffolds but not in a reliable manner. Finding a method for extrapolating this effort in a consistent manner may be possible through increased scaffolds. Finally, we make recommendations for how to incorporate this scaffolding into science activities to accommodate the learning of both the course content and the computational thinking skills.

1. PRIOR WORK

1.1 Frameworks

Scratch is a graphical programming environment in which students write programs to create animations. In Scratch, rather than the user having to key their program in text, they use drag-and-drop blocks shaped like puzzle pieces. This makes Scratch much more user-friendly for its target audience of young children since it is impossible to make a program that is syntactically incorrect as the syntax is enforced by the shapes of the blocks.

Brennan and Resnick [2] developed a framework for teaching and evaluating computational thinking by enumerating seven core concepts of programming: sequences, loops, events, parallelism, conditionals, operators, and data. These correspond to several features in the Scratch tool (p. 3-6):

1. Sequences: a particular activity or action that is expressed as a series of individual steps that can be executed in order by computer

Figure 1. Scratch event block

2. Loops: Mechanisms for running identical sequences multiple times

3. Events: "One thing causing another thing to occur" (see Figure 1 example)

4. Parallelism: Sequences of instructions happening simultaneously

5. Conditionals: Making decisions based on certain conditions, which can support multiple outcomes

6. Operators: Enable the programmer to preform numeric and string manipulations through mathematical, logical, and string expressions

7. Data: Storing, retrieving, and updating values such as variables and lists

In order to enable the effective development of computational thinking in the classroom, Repenning, Webb, and Ioannidou [3] created a checklist of six conditions that must be met by the tool used. They include low threshold, high ceiling, scaffolds Flow, enables transfer, supports equity, and systemic and sustainable. Several of these conditions were the initial reasons for the creation of Scratch, which is a drag and drop, block based, programming language designed for use in elementary school through college (especially ages 8-16) in multiple disciplines. It also contains a community of submitted works currently containing over 5 million projects and resources specifically for educators wanting to incorporate Scratch [4]. For these reasons, Scratch was selected as the data source for this pilot study.

1.2 Scaffolding

Scaffolding is a vital part of any complex activity (e.g. [5]). Scratch contains several scaffolding elements to assist learners in their program development. First, the block shapes either fit together like puzzle pieces or fill specific openings within other blocks. The blocks are also color-coded and organized into tabs based on their function (i.e. event blocks are orange). However, these scaffolds are simply tools and do not guide learners as they work to complete a specific task or activity. According to

Azevedo et al. [6] in a survey of studies on scaffolding self-regulated learning and metacognition, "support can be in the form of pre-stocked questions, static questions, dynamic support that is tailored to the student and context, or computer-based tools that guide students in their thinking, etc." (p. 370).

In a recent review of computational thinking in K-12 by Grover and Pea [7], several gaps in the literature were identified. Of these, the "use of appropriate scaffolds for successful transfer" (p.42) was specifically identified as an area of research that has not been advanced far beyond studies of the LOGO debugging curriculum [8]. In discussing the next steps for this research, it is important to keep in mind that student not only need to construct "conceptual understanding, [they] need to acquire new disciplinary strategies to guide reasoning in the domain" (p. 277, [9]). As we analyze current science programs for computational thinking, it will be important to address not only the computational thinking aspects but also those of the domain.

2. METHODS

2.1 Hypotheses

Students who create a Scratch project that represents their science fair project in a Digital Technologies class will mostly do so as an animation. In other words, they will simply display aspects of the project for the user to view and not make the project interactive.

In an unscaffolded/uscripted open-coding environment, there is evidence of computational thinking concepts regardless of the scaffolding provided to the students as they complete an activity.

2.2 Context and Tool

As stated above, the Scratch community is home to over 5 million shared projects posted by users. Using the search feature, "sciencefair" was entered as a keyword. Of the returned results, Scratch Studio - Year 7 **Science Fair** Projects 2013 was selected because it contained a set of 32 projects (29 of which were able to be analyzed) and the following description giving context to the projects: "Year 7 conducted awesome Science Fair investigations in Science. Then, in Digital Technologies they made these models of their experiments and investigations. Great work guys!" All students were in Year 7 (age 11) in New Zealand, which would be equivalent to 6th grade in the US. The projects were then descriptively coded [13] to determine general characteristics of the projects. To better analyze the computational thinking concepts in the student projects, a text file of each project was loaded into Scrape [10]. The Scrape Local tool allows a quantitative analysis of the blocks used in the projects to be analyzed by individual project or as a group of projects.

A block is elements of code in Scratch. They are color-coded based on their function (i.e. orange is an event block). A sprite is a character or object to which the code is assigned to enact the code. For instance, if the block above was assigned to Sprite 7, when the user presses the down arrow on the keyboard, it will cause something to happen to Sprite 7 (i.e. it will move down).

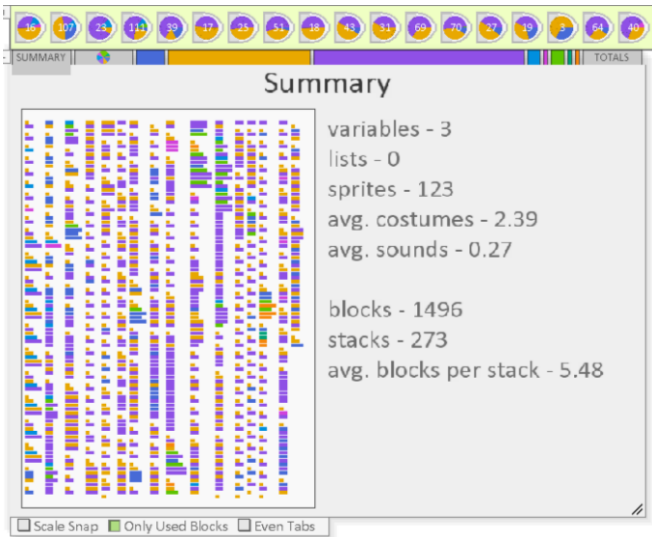
Table 1. Summary of computational thinking concepts in students projects

Concept from Brennan & Resnick	% Student Usage
Events	100%
Sequence	96.5%
Parallelism	58.6%
Loops	34.5%
Conditionals	20.7%
Operators	20.7%

2.3 Analysis

All 32 projects were in the current Scratch 2.0 format. Scrape will only analyze Scratch 1.4 projects. Therefore, all projects were convert from version 2.0 to 1.4. However, 3 projects would not convert properly and were therefore removed from

the sample. The 29 remaining projects were then converted to text files and uploaded into the Scrape Local tool. The files were first combined for a single analysis (see top of Figure 2). Figure 2 shows the "Summary" tab and "Pie Chart" tab. Projects were then analyzed individually and percentages for each block type were entered into a spreadsheet (see Table 2). Finally, the individual projects were coded as to whether they required user interaction (beyond click the green flag) and the code was assessed for computational thinking concepts according to the Brennan and Resnick [2] framework (see Table 1 for summary).



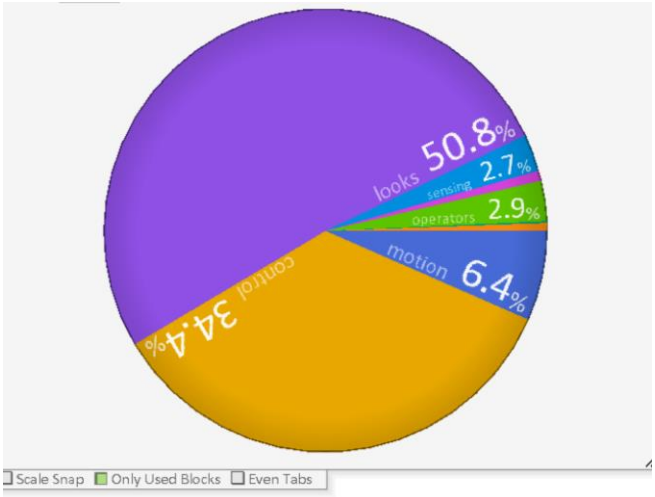


Figure 2. Summary and pie chart for all 29 coded projects using Scrape Local tool

2.4 Case Studies

Twenty-nine projects make up the larger study. Four case projects were selected as exemplars of the most extensive use of computational thinking concepts. The projects are qualitatively coded for the computational thinking and scientific inquiry concepts present.

3. FINDINGS

3.1 Student Transfer

We found that 14 of the 29 projects only allowed the user to view the project and 15 projects required user interaction (pressing keys to enable certain actions). The projects that were view only showed a rocket launch, 5 plants and the effect of sun, the life cycle of an eel, feeding chickens and others. The interactive projects showed the effect of different solutions on flowers, the phases of the moon, solutions that attract a bee, a volcanic explosion and others. Most (21 projects) of the text was achieved through broadcast, which is one of the first skills taught in basic programming. Switching costumes was a popular way (17 projects) to move or change objects in the project. However, this too is a simplistic skill in programming. Only 9 projects made use of more advanced concepts such as if/then (conditional), repeat (loop), and forever.

3.2 Evidence of computational thinking

The Scrape tool shows a summary of the quantitative characteristics of the set of projects. In all, 1,496 blocks and 123 sprites were used among the 29 projects (Figure 1). Looking at the pie chart, 50.8% of all blocks used were for "looks". These include blocks that "say" or "think" something, switch costumes, or hide.

Control blocks were 34.4% of the total blocks used. This category includes the customary starting block "when green flag clicked". In all, 85.2% of the total blocks used were basic functions within the projects. Only 14.8% of the blocks used consisted of more complex functions.

The projects were also analyzed individually using the Scrape tool. Table 2 shows a snapshot of this analysis. Most project followed the overall trend of primarily Orange (control) and Purple (looks) blocks but 9 deviated using more sophisticated blocks (ex. Project #5). Also, in accordance with the framework developed by Brennan & Resnick [2], sequence and events was used in almost all projects but loops, conditionals, and operators were not frequently used, demonstrating higher-level concepts. However, no student made use of data.

3.2.1 Case Study 1

In case study 1, Project 5, both computational thinking and scientific inquiry are explicitly displayed. The program shows a rainbow colored smiley face that begins by asking your name and asking you to guess its name. Then it displays text documenting the hypothesis, experimental procedure for dissolving the same amount of sugar in water, vinegar and lemonade and then timed how quickly each dissolved. Finally, it gives the result that vinegar dissolved the quickest (approximately 20 minutes), then water and lastly the lemonade. There is also a short demo of squares (representing sugar) in cups marked V, W, and L showing how many squares disappeared over the period of time. This program demonstrates all but the two initials steps (question and research) that are typically part of the scientific inquiry process.

Conditionals and operators, as well as sensing blocks (ask and answer), are used in the program to ask and answer questions posed related to name and the results of the experiment. Parallelism is used throughout the project but particularly at the end when you are asked to push "1". Three different sprites, representing cups of water, vinegar and lemonade, have events that occur simultaneously.

3.2.2 Case Study 2

In case study 2, Project 24, the program shows a gray hand with white fingernails. A "cotton wool" and "nail polish" are also shown on the screen. The directions instruct the user on how to move the cotton wool and nail polish and state the "this experiment is all about how nail polish remover takes it off and what happens." Essentially, the user is controlling the experiment. The program is missing all other aspects of scientific inquiry, but very accurately simulates how the student may have carried out the experiment.

Each nail uses a loop, conditional, and sensing block to detect which sprite (cotton wool or nail polish) is touching. There is also a loop, conditional and sensing in the cotton wool that will play a sound when touching. Parallelism exists with all sprites except the nail polish as they all have some event that occurs upon clicking the green flag.

3.2.3 Case Study 3

In case study 3, Project 25, starts with a question, "does color affect the taste of food?". The program then proceeds to show an experiment in which three people taste the same food colored both red and blue. The results of the experiment show that the taste testers did not taste a difference based on color. The program then shares research on the topic of color, discussing that red coloring can cause hyperactivity and that in some studies, subjects given orange juice and orange colored water responded that they taste the same.

Conditionals, operators and the ask/answer blocks were again used when the program asked a question and responded to an answer. Parallelism, conditionals, loops and operators were used with a sprite that jumped across the screen to demonstrate that red can make a person hyper.

Table 2. Snapshot of data from Science Fair 2013 Scratch projects

Proj #	# of Scripts	# of Sprites	Concepts	# of Blocks	%Orange -Control	%Purple - Looks	%Blue -Motion
1	1	2	none	3	66.6		33.3
2	14	11	broadcast, switch costumes (**Sequence**)	69	36.2	63.7	
3	22	4	switch costumes (**Sequence**)	43	50	45.4	4.5
4	7	7	broadcast, switch costumes (**Sequence**)	64	26.5	64	9.3
5	16	6	Broadcast, switch costumes, **if/then** (**Sequence, Conditional**)	222	26.5	63.5	

3.2.4 Case Study 4

In case study 4, Project 27, starts with a hypothesis stating that mixing milk with concrete will take longer to dry than juice or the traditional water and concrete. The sprite then describes the experiment and gives the result that the milk took 3 days longer to dry and the water and concrete dried the fastest. The sprite also describes the composition of concrete (research). This project demonstrates all of the typical parts of scientific inquiry except an analysis of the experiment.

The program uses background changes to display most of the content. Parallelism occurs with all 5 sprites and the background as the green flag activates them all. There is a bouncing ball throughout the program that uses conditionals, loops, and operators but does not contribute to the program in any meaningful way.

Figure 3. Example of an "Hour of Code" style activity

4. CONCLUSIONS

From these analyses, we can conclude that while more advanced computational thinking concepts are evident in these science fair projects, a more scaffolded project is needed if we want students to use more complex concepts at a more regular interval. Case study 1 shows a program that demonstrates most of the scientific inquiry process typically required in a science fair project. It also makes use of computational thinking concepts to complete some of these tasks. Case studies 2-4 also have elements of scientific inquiry but, again, only use computational thinking concepts for asking and answering questions and the movement of sprites that are present for aesthetic effect. It is evident that students have a working understanding of Scratch blocks and how to construct complex programs. However, students were not successful in applying those skills to scientific inquiry. It is possible that this was simply their first attempt and they need more practice in domain specific programming tasks. Another explanation could be there they need a set of activities to scaffold their transfer of knowledge of computational thinking to the science domain. Given our recent experience with Code.org's "Hour of Code" [11] and recommendations from research regarding scaffolding activities, we plan to create a series of activities in the style of "Hour of Code" to scaffold the transfer of student knowledge from the computing domain to the science domain.

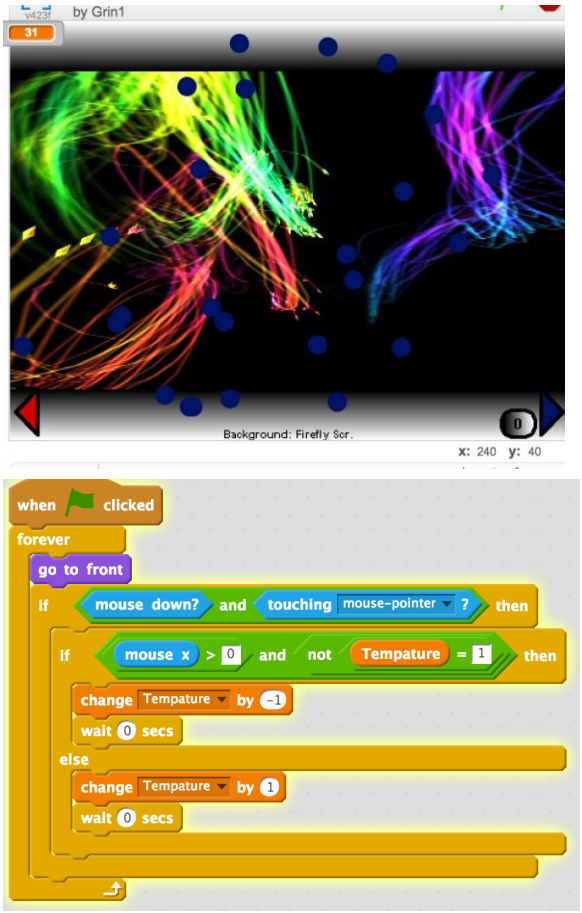


Figure 4. Sample Scratch project and code

5. NEXT STEPS

We now have a better understanding of how a group of middle school students spontaneously apply computational thinking concepts to a science fair project activity using Scratch. Our next step is to develop a set of "Hour of Code" science lessons in which the environment is better controlled and scaffolded to assess the students current and developing computational thinking practices and help them to better apply them to the science domain. Figure 3 shows the basic format of the activities created by our team [12] as an introductory activity for university non-majors. This is the second activity. The first activity had the same objective, say "hello world" but the students were only given 3 blocks. This version has the students ask "what is your name" first and then have the program respond, "hello world". Also note that all blocks have already been filled-in with the relevant text and there are step-by-step instructions along the left-hand side. The blocks are limited to only those needed for the activity and the instructions explain the purpose behind each step (e.g. drag and drop the green *command* block labeled "clear" that will clear the screen). This satisfies Dabbagh & Kitsantas' [16] first steps of scaffolding where novice learners are supported by limiting the complexity of the task. As the activities proceed, more blocks are added and the task becomes more complex which fades the limitations. Finally, the proposed structure of the activities also includes a "show me the answer" button. Worked examples

have shown to help students who are struggling to solve a problem, for example in intelligent tutoring systems [14].

As stated above, Scratch contains blocks that fit together and are color-coded according to their function. However, there are 9 types of blocks each with several individual blocks, which would be overwhelming to a novice programmer. Therefore, the scaffolding suggested above would limit the complexities of the entire program and allow students to learn the basic functions of the programming language without being overwhelmed by the sheer number of options.

In order to address the linkage missing between computational thinking concepts and scientific inquiry concepts, as in the case studies, a program designed to address both needs to be the end activity. Figure 4 shows a sample Scratch program that uses conditionals, loops, operators and data (missing in all previously studied projects). The program shown in in Figure 4 demonstrates molecular behavior (shown as small blue balls) as it relates to temperature. This program not only demonstrates complex computational thinking concepts, but also higher level scientific conceptual understanding (see nature of matter learning progression [15] for levels of understanding). Again, this is a complex task that would need to come at the end of a series of exercises but demonstrates the type of program that would reflect a student's understanding of both types of key concepts and the ability to transfer knowledge between the domains of computing and science.

6. REFERENCES

[1] *Next Generation Science Standards.* Achieve, Inc. on behalf of the twenty-six states and partners that collaborated on the NGSS, Washington, DC, 2013.

[2] K. Brennan and M. Resnick. 2012. New frameworks for studying and assessing the development of computations thinking. In *Proceedings of the American Educational Research Association* (Vancouver, BC, Canada, April 13 - 17, 2012).

[3] A. Repenning, D. Webb, and A. Ioannidou. Scalable game design and the development of a checklist for getting computational thinking into public schools. *In Proceedings of the 41st ACM Technical Symposium on Computer Science Education,* pages 265-269. ACM, 2010.

[4] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (November 2009), 60-67. DOI=10.1145/1592761.1592779 http://doi.acm.org/10.1145/1592761.1592779

[5] M.T.H. Chi, S. Siler, H. Jeong, T. Yamauchi, & R. Hausmann. 2001. Learning from human tutoring. *Cognitive Science* 25, 471–534.

[6] R. Azevedo and A. F. Hadwin. 2005. Scaffolding self-regulated learning and metacognition – Implications for the design of computer-based scaffolds. *Instructional Science* 33, 367-379.

[7] S. Grover and R. Pea. 2013. Computational thinking in K-12: A review of the state of the field. *Educational Researcher* 42, 38-43.

[8] D. Klahr and S. M. Carver. 1988. Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology* 20, 362-404.

[9] B. J. Reiser. 2004. Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *The Journal of the Learning Sciences* 13, 273-304.

[10] U. Wolz, C. Hallberg, and B. Taylor. 2011. Scrape: A tool for visualizing the code of Scratch programs. *In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education,* Dallas, TX.

[11] Code.org. 2014. Hour of Code. Retrieved September 5, 2014 from http://code.org/

[12] B. W. Peddycord III. 2014. BJC Hello World. Retrieved August 5, 2014 from http://isharacomix.org/snap-exercises/lessons/hello/#1

[13] J. Saldaña. 2012. *The coding manual for qualitative researchers* (No. 14). Sage.

[14] I. Roll, V. Aleven, B. M. McLaren, and K. R. Koedinger. 2011. Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction* 21, 267-280.

[15] S. Y. Stevens, C. Delgado, and J.S. Krajcik. 2010. Developing a hypothetical multi-dimensional learning progression for the nature of matter. *Journal of Research in Science Teaching* 47, 687-715.

[16] Dabbagh, N., & Kitsantas, A. (2005). Using web-based pedagogical tools as scaffolds for self-regulated learning. *Instructional Science*, *33*(5-6), 513-540.