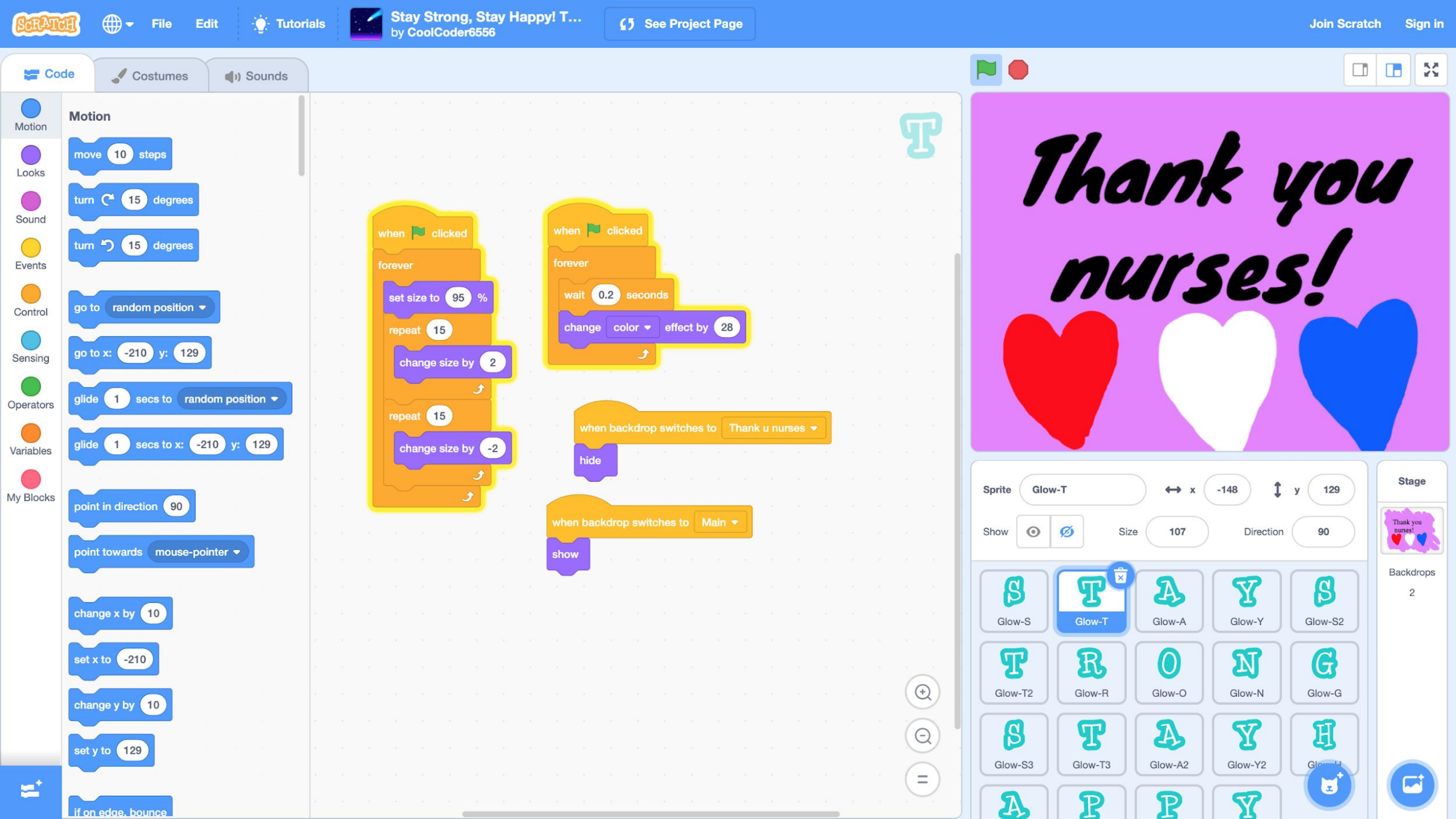
Three questions about CT in PK–5

Karen Brennan
Harvard Graduate School of Education
Wednesday, May 6, 2020
karen_brennan@gse.harvard.edu

10 years ago...



Scratch
File
Edit
Tutorials
Stay Strong, Stay Happy! T...
by CoolCoder6556
See Project Page
Join Scratch
Sign in
Code
Costumes
Sounds
Motion
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
move 10 steps
turn 15 degrees
turn 15 degrees
go to random position
go to x: -210 y: 129
glide 1 secs to random position
glide 1 secs to x: -210 y: 129
point in direction 90
point towards mouse-pointer
change x by 10
set x to -210
change y by 10
set y to 129
if on edge, bounce
when clicked
forever
set size to 95 %
repeat 15
change size by 2
repeat 15
change size by -2
when clicked
forever
wait 0.2 seconds
change color effect by 28
when backdrop switches to Thank u nurses
hide
when backdrop switches to Main
show
Thank you nurses!
Sprite
Glow-T
x -148
y 129
Show
Size 107
Direction 90
Stage
Backdrops
2
Glow-S
Glow-T
Glow-A
Glow-Y
Glow-S2
Glow-T2
Glow-R
Glow-O
Glow-N
Glow-G
Glow-S3
Glow-T3
Glow-A2
Glow-Y2

Viewpoint | Jeannette M. Wing

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

LISA HANEY

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

Table 1. Strengths and limitations of assessment approaches.

	Concepts	**Practices**	**Perspectives**
Approach #1: Project Analysis	presence of blocks indicates conceptual encounters	N/A	N/A (possibly by extending analysis to include other website data, like comments)
Approach #2: Artifact-Based Interviews	nuances of conceptual understanding, but with limited set of projects	yes, based on own authentic design experiences, but subject to limitations of memory	maybe, but hard to ask directly
Approach #3: Design Scenarios	nuances and range of conceptual understanding, but externally selected projects	yes, in real-time and in a novel situation, but externally selected projects	maybe, but hard to ask directly

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design.* Paper presented at annual American Educational Research Association meeting, Vancouver, Canada.

Now, in 2020...

Why CT?

What is CT?

How can we support CT?

Why CT?

What is CT?

How can we support CT?

Santo, R., Vogel, S., & Ching, D. (2019). *CS for What? Diverse Visions of Computer Science Education in Practice*. New York, NY: CSforALL. Retrieved from https://www.csforall.org/visions/

Why CT?

What is CT?

How can we support CT?

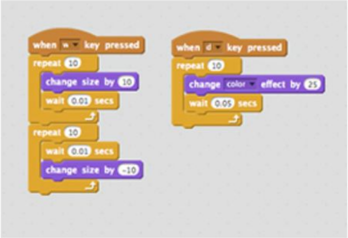
COMPUTATIONAL THINKING WITH SCRATCH

DEVELOPING FLUENCY WITH COMPUTATIONAL CONCEPTS, PRACTICES, AND PERSPECTIVES

DEFINING | ASSESSING | SUPPORTING

WHAT IS COMPUTATIONAL THINKING?

Over the past five years, we have developed a computational thinking framework based upon our studies of interactive media designers. The context of our research is Scratch — a programming environment that enables young people to create their own interactive stories, games, and simulations, and then share those creations in an online community with other young programmers from around the world. By studying activity in the Scratch online community and in Scratch workshops, we have developed a definition of computational thinking that involves three key dimensions: (1) computational concepts, (2) computational practices, and (3) computational perspectives. Observation and interviews have been instrumental in helping us understand the longitudinal development of creators, with participation and project portfolios spanning weeks to several years. Workshops have been an important context for understanding the practices of the creator-in-action.



CONCEPTS

As young people design interactive media with Scratch, they engage with a set of computational **concepts** that are common in many programming languages. We have identified seven concepts, which are highly useful in a wide range of Scratch projects, and which transfer to other programming (and non-programming) contexts:

- **sequence:** identifying a series of steps for a task
- **loops:** running the same sequence multiple

PRACTICES

From our interviews with and observations of young designers, it was evident that framing computational thinking solely around concepts insufficiently represented other elements of designers' learning and participation. The next step in articulating our computational thinking framework was to describe the processes of construction, the design **practices** we saw kids engaging in while creating their projects. Although the young people we interviewed had adopted a variety of strategies and practices for developing interactive media, we observed

PERSPECTIVES

In our conversations with Scratchers, we heard young designers describe evolving understandings of themselves, their relationships to others, and the technological world around them. This was a surprising and fascinating dimension of participation with Scratch — a dimension not captured by our framing of concepts and practices. As the final step in articulating our computational thinking framework, we added the dimension of **perspectives** to describe the shifts in perspective that we observed in young people working with Scratch, which included three

Operational Definition of Computational Thinking
for K–12 Education

The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) have collaborated with leaders from higher education, industry, and K–12 education to develop an operational definition of computational thinking. The operational definition provides a framework and vocabulary for computational thinking that will resonate with all K–12 educators. ISTE and CSTA gathered feedback by survey from nearly 700 computer science teachers, researchers, and practitioners who indicated overwhelming support for the operational definition.

Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- 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

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

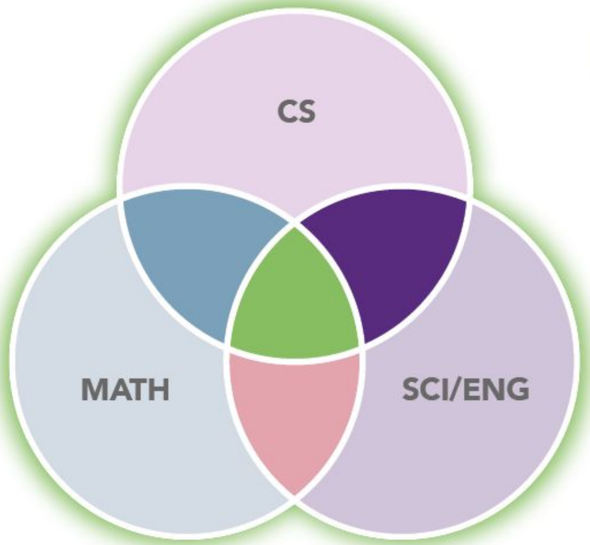
- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution

 This material is based upon work supported by the National Science Foundation under Grant No. CNS-1030054.

ISTE & CSTA. (2014). *Computation Thinking for All*. Retrieved from https://www.iste.org/explore/Solutions/Computational-thinking-for-all.

computational thinking

~~computational~~ thinking



CS + Math

- **Develop and use abstractions**
  M2. Reason abstractly and quantitatively
  M7. Look for and make use of structure
  M8. Look for and express regularity in repeated reasoning
  CS4. Developing and Using Abstractions
- **Use tools when collaborating**
  M5. Use appropriate tools strategically
  CS2. Collaborating Around Computing
- **Communicate precisely**
  M6. Attend to precision
  CS7. Communicating About Computing

CS + Sci/Eng

- **Communicate with data**
  S4. Analyze and interpret data
  CS7. Communicating About Computing
- **Create artifacts**
  S3. Plan and carry out investigations
  S6. Construct explanations and design solutions
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
  CS6. Testing and Refining Computational Artifacts

CS + Math + Sci/Eng

- **Model**
  S2. Develop and use models
  M4. Model with mathematics
  CS4. Developing and Using Abstractions
  CS6. Testing and Refining Computational Artifacts
- **Define problems**
  S1. Ask questions and define problems
  M1. Make sense of problems and persevere in solving them
  CS3. Recognizing and Defining Computational Problems

K–12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org.

~~computational~~ thinking

computational thinking

computational ~~thinking~~

participation

literacy

computational ~~thinking~~

making

action

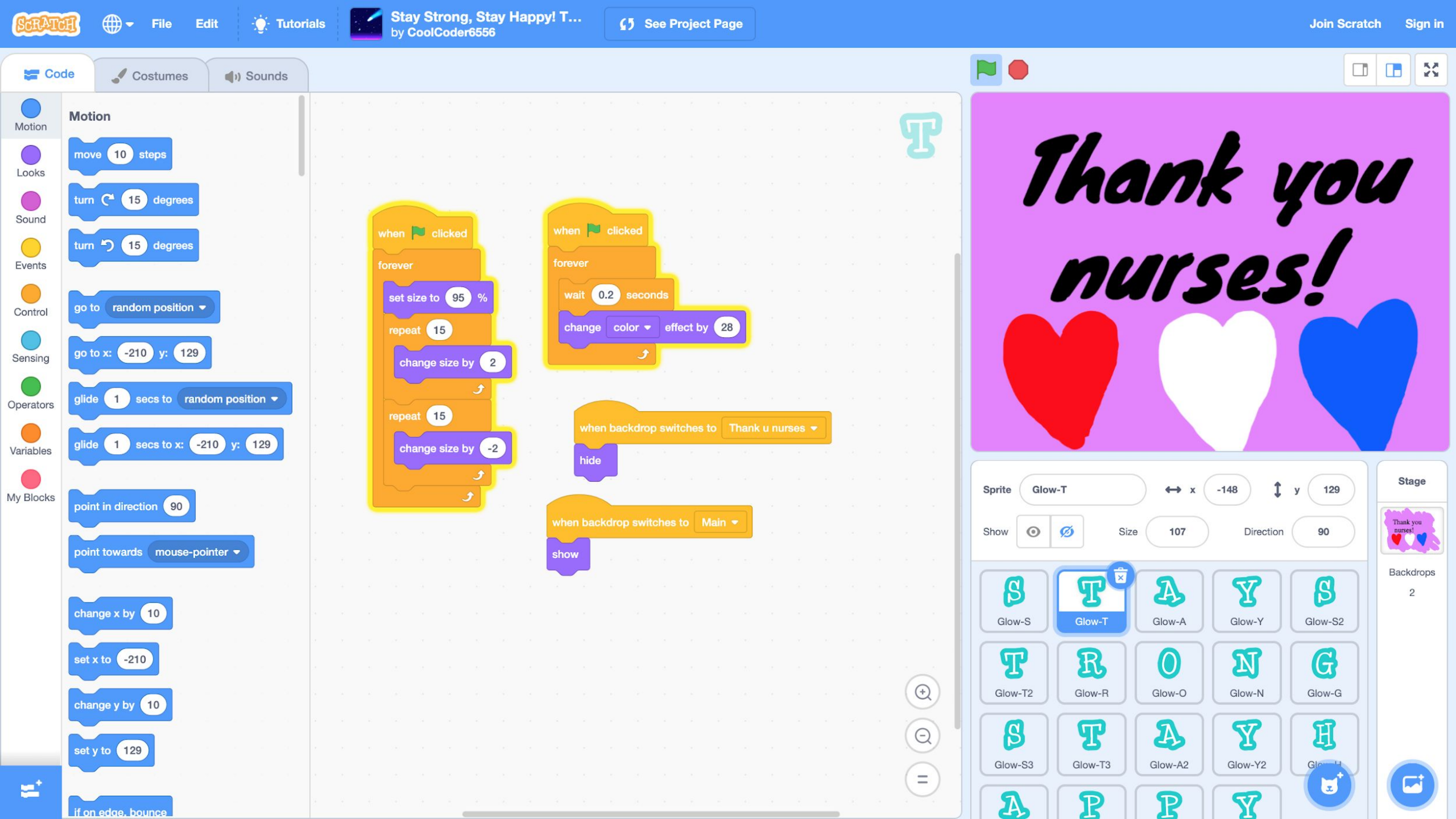
Kafai, 2016
Lee & Soep, 2016
Rode et al., 2015
Tissenbaum, Sheldon, & Abelson, 2019

Why CT?

What is CT?

How can we support CT?

Blikstein, P. (2018). *Pre-College Computer Science Education:*
A Survey of the Field. Mountain View, CA: Google LLC.



Join Scratch
Sign in
File
Edit
Tutorials
Stay Strong, Stay Happy! T...
by CoolCoder6556
See Project Page
Code
Costumes
Sounds
Motion
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
Motion
move 10 steps
turn 15 degrees
turn 15 degrees
go to random position
go to x: -210 y: 129
glide 1 secs to random position
glide 1 secs to x: -210 y: 129
point in direction 90
point towards mouse-pointer
change x by 10
set x to -210
change y by 10
set y to 129
if on edge, bounce
when clicked
forever
set size to 95 %
repeat 15
change size by 2
repeat 15
change size by -2
when clicked
forever
wait 0.2 seconds
change color effect by 28
when backdrop switches to Thank u nurses
hide
when backdrop switches to Main
show
Thank you nurses!
Sprite
Glow-T
x
-148
y
129
Show
Size
107
Direction
90
Stage
Backdrops
2
Glow-S
Glow-T
Glow-A
Glow-Y
Glow-S2
Glow-T2
Glow-R
Glow-O
Glow-N
Glow-G
Glow-S3
Glow-T3
Glow-A2
Glow-Y2

Lesson 5

Peanut Butter & Jelly Algorithms

In this unplugged lesson, students will construct algorithms to first create a peanut butter and jelly sandwich, and then to make their own sandwich.

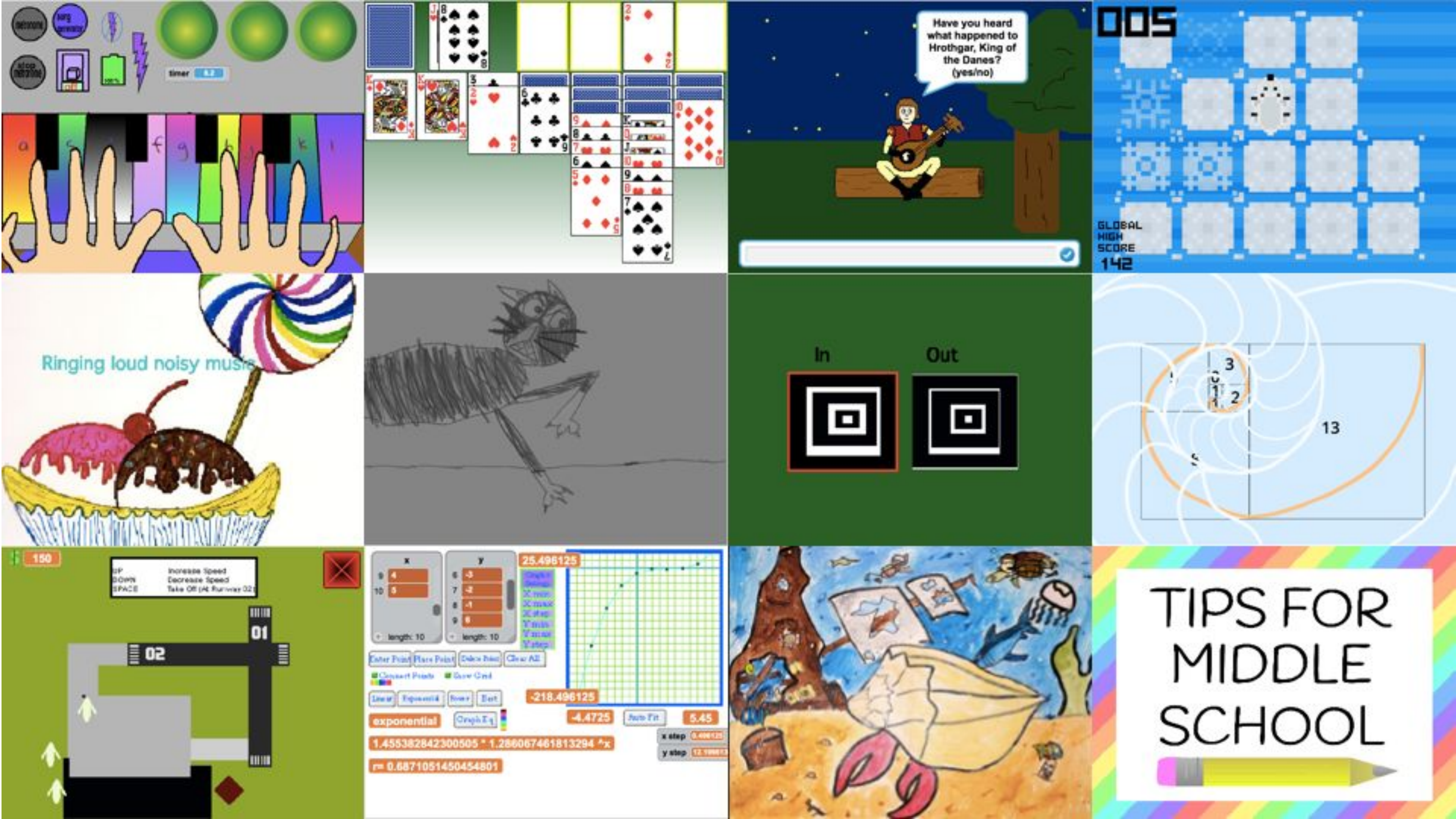
View Lesson Plan

Agenda

1. Video: **Peanut Butter & Jelly** (1:58)

Materials

- PB&J Algorithm cards **English** / **Spanish**


Have you heard what happened to Hrothgar, King of the Danes? (yes/no)
005
GLOBAL HIGH SCORE 142
Ringing loud noisy music
In
Out
3
2
13
150
UP Increase Speed
DOWN Decrease Speed
SPACE Take Off (At Runway 02)
01
02
25.496125
-218.496125
-4.4725
5.45
exponential
1.4553828423000505 * 1.286067461813294 ^x
r= 0.6871051450454801
TIPS FOR MIDDLE SCHOOL

Research on Learning in Formal and Informal Settings

STEM + Computing K-12 Education (STEM+C)

STEM+C Program FAQs

Additional guidance for the STEM+C Program may be found in the FAQs. Please review the information in the FAQs; if you have further questions, contact program personnel.

CONTACTS

Name	Email	Phone	Room
Arlene M. de Strulle	adestrul@nsf.gov	(703) 292-8620	
Chia Shen	cshen@nsf.gov	(703) 292-8447	

J Sci Educ Technol (2016) 25:127–147
DOI 10.1007/s10956-015-9581-5

CrossMark

Defining Computational Thinking for Mathematics and Science Classrooms

David Weintrop[1,2] · **Elham Beheshti**[3] · **Michael Horn**[1,2,3] · **Kai Orton**[1,2] · **Kemi Jona**[2,3] · **Laura Trouille**[5,6] · **Uri Wilensky**[1,2,3,4]

Published online: 8 October 2015

Abstract Science and mathematics are becoming computational endeavors. This fact is reflected in the recently released Next Generation Science Standards and the decision to include "computational thinking" as a core scientific practice. With this addition, and the increased presence of computation in mathematics and scientific contexts, a new urgency has come to the challenge of defining computational thinking and providing a theoretical grounding for what form it should take in school science and mathematics classrooms. This paper presents a response to this challenge by proposing a definition of computational thinking for mathematics and science in the form of a taxonomy consisting of four main categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. In formulating this taxonomy, we draw on the existing computational thinking literature, interviews with mathematicians and scientists, and exemplary computational thinking instructional materials. This work was undertaken as part of a larger effort to infuse computational thinking into high school science and mathematics curricular materials. In this paper, we argue for the approach of embedding computational thinking in mathematics and science contexts, present the taxonomy, and discuss how we envision the taxonomy being used to bring current educational efforts in line with the increasingly computational nature of modern science and mathematics.

Keywords Computational thinking · High school mathematics and science education · STEM education · Scientific practices · Systems thinking · Modeling and simulation · Computational problem solving

✉ David Weintrop
dweintrop@u.northwestern.edu

1 Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL 60208, USA

2 Learning Sciences, Northwestern University, Evanston, IL 60208, USA

3 Computer Science, Northwestern University, Evanston, IL 60208, USA

4 Northwestern Institute on Complex Systems, Evanston, IL 60208, USA

5 The Adler Planetarium, Chicago, IL 60605, USA

6 Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA), Northwestern University, Evanston, IL 60208, USA

Introduction

> By 2020, one of every two jobs in the "STEM" fields will be in computing
> (ACM pathways report 2013)

The release of the Next Generation Science Standards (NGSS) places a new emphasis on authentic investigation in the classroom, including eight distinct scientific practices (NGSS Lead States 2013). While some of these practices are familiar to veteran teachers, such as "asking questions and defining problems," others are less well understood. In particular, the practice of "using mathematics and computational thinking" reflects the growing importance of computation and digital technologies across the scientific disciplines. Similar educational outcomes can be found in mathematics standards, such as the Common Core guidelines, which state that students should be able "to use technological tools to explore and deepen their

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.

Why CT?

What is CT?

How can we support CT?