Journal of Technology-Integrated Lessons and Teaching

2025 Volume 4, Issue 1

ASSOCIATION FOR EDUCATIONAL COMMUNICATIONS & TECHNOLOGY

Acknowledgements

The Journal of Technology-Integrated Lessons and Teaching (JTILT) is an online, peer-reviewed, open-access journal published semi-annually by the University of Wyoming in partnership with the Teacher Education Division of the Association for Educational Communications and Technology (AECT). The journal publishes original, technology-rich lessons, activities, professional development, and micro-credentials for PK-16+ professionals and their international equivalents.

The views expressed in JTILT are not necessarily those of the publisher or the partnering association.

Front Cover

The front cover is a composite image created from the following images:

Marble. By danielkirsch. Pixabay License. Cropped and rotated from the original.
Puzzle Pieces. By stux. Pixabay License. Cropped from the original
Green Hexagon. By Craig Shepherd. Cropped and rotated from the original.
Gears and Cogs. By Mustangjoe. Pixabay License. Cropped and rotated from the original.
Shell cutaway. By Peggy_Marco. Pixabay License. Cropped and rotated from the original.
Boy with marker. BySchoolPRPro. Pixabay License. Cropped, flipped horizontally, and rotated from original.
Woman holding leaf. By youzuowei1230. Pixabay License. Cropped from the original.

Usage Rights and Permissions

Appropriate Credit without Modification

Appropriate Credit with Modification

If users make modifications, additions, or transformations to the resources provided, they must:

- Identify the original authors of the resources.
- License their derivative work under a CC BY-NC-SA 4.0 license and include a link to the license.
- Indicate what modifications were made.
- Refrain from suggesting that the original licensor endorsed their modifications or use.

Additional information about providing appropriate credit is located at CC Wiki's Best Practices for Attribution.

Journal Editors

Editorial Board

Journal of Technology-Integrated Lessons and Teaching Contents

Volume 4 Issue 1 June 2025

Front Matter

Learning Representations

Technology-Rich Lesson Competition

Purpose and Details

The Technology-Rich Lesson Competition recognizes educators for original, PK-12 (or equivalent) lessons and activities. Submissions must include original lessons or activities that focus on stated goals and objectives. Technology should be integral to the submission and help to improve identified student learning outcomes. Submissions should include the following sections:

- A **concise title** that accurately describes the lesson or activity.
- An **overview paragraph** that indicates the subject, topic, duration, what learners will do, what technology-rich experience(s) will take place, and what assessments will occur.
- A **materials** section that identifies ALL materials needed for implementation.
- A **context and setting section** that describes the intended setting (e.g., urban, rural, public, private), grade-level (or equivalent), classroom characteristics, learner characteristics—including prior content and technical knowledge, instructor characteristics, fit within the larger curriculum, development rationale, and so forth (refer to the first page of the JTILT lesson plan template for details).
- A **setup section** describing how to set up and organize the learning experience.
- **Goals and learning objectives** and their alignment with professional standards (as applicable).
- The **lesson and instructional strategies** are described in sufficient detail so that others can implement them as-is or adapt them to meet individual needs.

Submission Information

- **Submissions are due October 1, 2025.**
- Submissions should not exceed five pages, excluding resources (e.g., presentations, work samples, assessments, and rubrics that are included as separate documents).
- Submissions must be original, solely created by the submitter(s), and not published elsewhere (beyond personal websites, blogs, or equivalent personal publication locations).
- Submission resources must be created by the submitter(s), open-access, or include Copyright permissions.
- Submit your lesson and resources on the JTILT website. Login or create a new account. Then select the New Submission button. When the Submit an Article form appears, select JTILT Competitions from the Section dropdown. Complete the form and upload your materials.

Eligibility

- Submitters need not be current members of the Association of Educational Communications and Technology (AECT) nor the Teacher Education Division (TED)
- Current AECT TED board members and JTILT editorial board members are ineligible for this competition.

Award

- Two winning submissions will each receive $100 to be split among the authors.
- Winners will be announced in the TED membership meeting during the AECT Annual Convention held October 20-24, 2025, at Planet Hollywood in Las Vegas, Nevada.
- Winning submissions will be published in JTILT.

Journal of Technology-Integrated Lessons and Teaching
(2024) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9633

Introduction: Computational Thinking and Computer Science Special Issue

Craig E. Shepherd and Cecil R. Short, Editors-in-Chief

As computers become more functional and ubiquitous, societies are placing greater emphasis on programming and development skills. Computer science credentials and degree programs have long existed in higher education. Many high schools have also offered computer science courses like coding, computer graphics, game development, and cybersecurity. However, the desire to push computer science training to younger audiences is increasing. Currently, a dozen states require computer science instruction as a prerequisite for high school graduation (Barack, 2025). Many others provide opportunities for computer science experiences within PK-12 curricula. However, computer science topics may intimidate educators and students. Coding languages like Python, C#, and Javascript feel cryptic and take time to learn. Block options like Scratch provide easier entries into coding but still require sustained effort and attention.

Yet, computer science is broader than coding. The processes involved in computer science often involve components of problem solving labeled as computational thinking. These processes involve approaches like problem recognition, decomposition, pattern recognition, abstraction, and algorithmic thinking. These processes can also be introduced in myriad grade levels and subject areas. This special issue includes nine articles that describe various low and high-tech approaches to teach computational thinking and computer science.

The first three articles focus on the use of unplugged and high-tech tools to foster computational thinking. “Puzzling our way into computational thinking” by David Mulder leverages puzzle pieces to help practicing P-8 teachers consider the processes needed to solve a puzzle, recognize patterns, and deconstruct the challenge. “Unplugged and plugged: An introduction to coding for elementary school children” by Jessica Kerr and Theodore Kopcha leverages low-tech tools in a 4-5th grade afterschool program to help students transition to block-based coding. “Integrating CT in science methods: Advancing practice and pedagogy” by Ugur Kale and Yuahnua Wang moves the focus to preservice teacher preparation where unplugged activities and block coding help students develop computational thinking lessons implemented during practicum experiences.

Computational thinking is relevant to various subjects. “The plot thickens: Literacy through computational thinking” by Bataul Alkhateeb and Eiman Abushihab and “Artificial intelligence-enhanced digital storytelling: Empowering young creators in a summer STEM camp” by Bulent Dogan and Amani Itani describe how computational thinking aligns with secondary and elementary English Language Arts curricula. “Transformations throwdown: Extending mathematics knowledge with Assemblr” by Brian Johnson et al. aligns computational thinking with middle-school mathematics standards. “Scratch day: Hands-on computational thinking activities for youth and adult learners” by Ayanna Perkins et al. used a day-long event to help learners of all ages gain awareness and experience with block coding.

The remaining two articles “Scratch Encore: Creating and sustaining culturally responsive computer science education” by Rasha Alkhateeb et al. and “Integrating CT/CS into a teacher education program: A year-long PD” by Irene Bal et al. focus on sustained curricula. All articles in this issue focus on in-person instruction. However, the articles about sustained curricula also included hybrid and online activities or guidance about how to make activities suitable for online and hybrid delivery.

These articles are by no means exhaustive on the subject. However, we hope they will provide ideas, perspectives, and activities to inspire educators who are considering how to leverage computational thinking and computer science concepts in their classrooms.

Get Involved

The JTILT editors and editorial board continue to examine how they can disseminate materials that are immediately useful, simplify processes, and increase their relevance to all teaching practitioners. As JTILT continues into its fourth year, we hope to see greater emphases on international and underrepresented voices. We welcome your voice! Reach out to the editorial team to get involved. Share your ideas and suggestions. Submit an article. Volunteer to review. JTILT is growing! We openly invite you to become part of that growth.

References

Barack, L. (2025, May 14). *Significant momentum for computer science curriculum*. K-12DIVE. https://www.k12dive.com/news/ significant-momentum-computer-science-curriculum-ai/748103/

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9205

Puzzling Our Way into Computational Thinking

David J. Mulder, Dordt University

Overview

This unplugged activity is a brief, practical introduction to computational thinking that uses an accessible and decidedly low-tech approach: solving a jigsaw puzzle. The skills needed to collaboratively solve a jigsaw puzzle illustrate the key concepts of computational thinking in a straightforward way that makes the basics of decomposition, pattern recognition, abstraction, and algorithmic thinking come to life. The learning representation included here was taught as part of a professional development workshop for PK-12 teachers but could easily be adapted to use with learners from upper elementary grades through middle school, high school, or university.

Topics: Collaborative Learning, Computational Thinking, Hands-on Learning

Time: 30 minutes

Materials

- Children's jigsaw puzzles (48-piece puzzles work well) – one for each group of 3-5 participants
- Zip-top storage bags
- Whiteboard and markers
- You are not a computer presentation

Setup

A classroom setup with tables works best. Arrange each table for 3-5 participants. Each participant should be able to see the whiteboard where ideas will be shared. To prepare the puzzles, remove each puzzle from its box and place the pieces in a zip-top storage bag. Keep the boxes for storage, but groups should receive pieces of their puzzle without the box or any picture of how the puzzle should appear.

Context-at-a-Glance

Setting
A professional development workshop for PreK-8 teachers at a non-public school located in the upper Midwestern United States.

Modality
Face-to-face

Class Structure
This lesson was implemented as part of a series of workshops related to computational thinking and educational robotics. This lesson was part of the first workshop in the series.

Learner Characteristics
Twenty-four PreK-8 educators participated in the workshop. Their experience ranged from first-year teachers to veteran educators with over 30 years of classroom experience. Only a handful of participants had any previous experience with coding or computational thinking.

Instructor Characteristics
A university professor with over 25 years' experience teaching in PreK-12 and higher education. He previously taught middle school and high school computer science courses, and currently teaches educational robotics and other STEM education courses. He provides STEM professional development opportunities to PreK-12 educators.

Development Rationale
This activity was developed to introduce computational thinking in a way that is accessible, meaningful, and relevant by using non-digital materials that were already familiar to the participants.

Design Framework
Social Constructivism (see Brau, 2022; Palinscar, 1998)

Context and Setting

Emphasis on computer science education and computational thinking skills is on the rise in K-12 education (Klein, 2023, Vegas & Fowler, 2020). As such, I am regularly invited to provide professional development (PD) opportunities for teachers in PK-12 schools, particularly related to topics in STEM education and educational technology, which are my areas of expertise. My education and experience have prepared me to do this sort of training, and I thoroughly enjoy it! I spent 11 years serving as a middle school math and science teacher, 3 years as a technology director and K-8 computer science teacher, and 13 years in higher education, where I teach courses in educational technology, STEM education, and educational foundations.

Through grant funding, my department was recently able to purchase some materials to support our students' learning about computer science education and computational thinking. Having access to these resources got me thinking about outreach opportunities in our area as well. I contacted local school districts to share that I am willing and able to provide PD on these topics for their teachers. The email I sent to administrators offering my services explained what computational thinking is, the possibilities of using computational thinking strategies across the curriculum, and the enjoyable aspects of moving from "novice to knower" with regard to learning about educational robotics—a good reminder for teachers of how it feels to be brand-new at learning something. These workshops were designed to help remind teachers how much courage it can take as a learner to try something new; something teachers sometimes forget, because we are experts in our content.

As part of this outreach, I was invited by a local non-public PK-8 school to provide a series of PD experiences to help introduce their faculty to some topics in computer science education that could have broad, cross-curricular appeal. Working with their Director of Learning in the planning process, we decided to focus on computational thinking, an introduction to block-based coding, and educational robotics. This lesson was a 30-minute segment of the full PD that focused on understanding computational thinking skills.

Each session of the full PD was scheduled for an hour-long block at the end of a Wednesday school day when the students had an early release so the staff could focus on PD. Twenty-four PK-8 teachers and paraeducators participated, representing a variety of grade levels and content areas. The participants in these PD opportunities had a range of experience as teachers. Some were in their first year, and several had over thirty years of experience as professional educators. None of the participants were currently teaching computer science as a primary part of their content responsibilities, though two of the teachers were tasked for teaching computer applications as part of their assigned curricula. Additionally, one of the paraeducators had some experience with block-based coding and educational robotics through an enrichment activity she helps facilitate.

The lesson described here came about serendipitously as I was preparing for our first session in this series of PD opportunities. I was reflecting on how I could best introduce the concepts of computational thinking as I was walking through the student lounge adjacent to our Education department offices. We always have a jigsaw puzzle out on one of the tables for students (and faculty) to work on collaboratively when they need a break from their academic work. Seeing the puzzle illustrated in Figure 1, I was suddenly struck with an idea for introducing computational thinking through an unplugged, readily accessible approach.

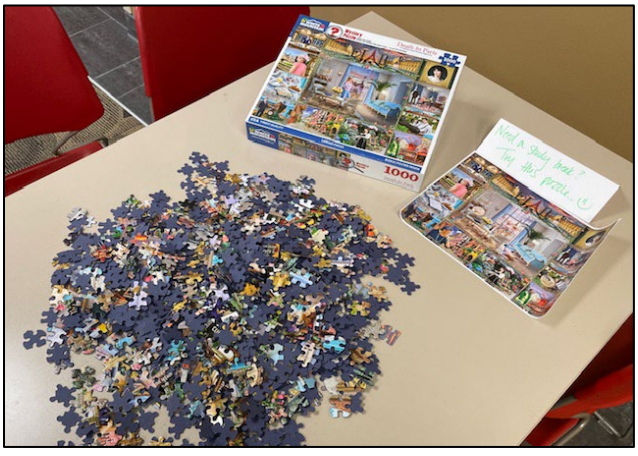


Figure 1. A photo of a communal jigsaw puzzle in our student lounge. The sign next to the puzzle box says, "Need a study break? Try this puzzle."

This lesson, described in the following section, illustrates the introduction to a series of PD workshops conducted with this same group of participants. The first workshop, including this

activity, introduced the idea of computational thinking, and gave a variety of opportunities for the participants to make connections to their grade levels and content areas. The second workshop in the series introduced block-based coding using Sphero BOLT+ robots, and participants learned the basics of how to use pattern recognition and algorithmic thinking particularly to address beginner-level coding challenges. The third and final workshop used Sphero BOLT+ robots to reinforce the things participants had previously learned and utilize all four computational thinking skills to achieve more demanding coding challenges.

Learning Representation

During this lesson, italic text identifies questions or prompts for the learners.

Introduction

Computational thinking is a structured approach to attacking complex problems (Wing, 2006). While generally considered a subset of computer science, computational thinking can be broadly applied to many different disciplines (Hunsaker, 2020; Lodi & Martini, 2021). In a seminal article from 2006, Wing suggests that "Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers" (p. 35). Instead of thinking like a computer, computational thinking is a key part of the creative work that humans do to attack authentic problems—the sort of "wicked problems" that crop up in almost every discipline.

While there are different definitions of what skills should be thought of as part of computational thinking (Cansu & Cansu, 2019), four cognitive skills are included in depictions of computational thinking across the literature (Campbell & Heller, 2019; Cansu & Cansu, 2019; Hunsaker, 2020; Lodi & Martini, 2021; McNicholl, 2019; Yin et al., 2020). These four key computational thinking skills are:

1. Decomposition
2. Pattern recognition
3. Algorithmic thinking
4. Abstraction (Campbell & Heller, 2019)

In this learning segment, the instructor leveraged the strengths of social constructivism as an approach for learning, prioritizing active engagement with other learners and facilitated discourse about the learning experiences (Palincsar, 1998). The idea of computational thinking was introduced through a hands-on activity, and then a discussion was facilitated with the learners to help them understand what these four component skills look like in practice. This approach illustrates the tenets of social constructivism, drawing on the strength of the group along the lines of Vygotsky's approach of learning by doing and by interacting with others in support of the learning process (Brau, 2022). The role of the instructor in such a social constructivist setting shifts to that of facilitator, asking questions, providing support, and dialoguing with learners throughout the learning process (Brau, 2022).

Opening the Session

The session began by inviting participants to sit down in groups of about five people at each table. They were given the opening prompt, "*When you hear the phrase 'computational thinking,' what ideas, concepts, and pictures come to mind? Think out loud with your tablemates about what 'computational thinking' might look like…"* Participants discussed their ideas, for about two minutes, after which they were invited to share the things that had come up in their table conversations. Their associations for "computational thinking" included things like computers, programming, and artificial intelligence. The instructor affirmed their ideas, and suggested that by the end of this session, they would have a clear definition for what computational thinking is.

As the participants moved into the activity for this session, the instructor shared three learning targets that would guide their work together:

1. *I can define and give examples of computational thinking.*
2. *I can commit to playfulness, collaboration, and celebration of learning.*
3. *I can take reasonable risks that will help me practice new skills.*

Hands-on, Minds-on Collaborative Investigation

At this point, the instructor gave each group a 48-piece jigsaw puzzle in a clear zip-top bag.

Participants did not receive the box or any indication of what the finished puzzle should look like.

They then received this instruction: "*Let's play! Grab your jigsaw puzzle and work with your group to complete it as quickly as possible.*" Figure 2 illustrates how a group began completing the puzzle.

Figure 2. A group early in their work putting together the puzzle. This group rapidly sorted pieces by color, which was a helpful strategy for them.

In each group, someone opened the bag, poured out the pieces onto the table, and the hands flew into action. Several groups joked about not receiving a picture of what they were trying to compose as they worked. But they quickly began flipping the pieces face-up, sorting them out, and snapping the pieces into place. There was a flurry of activity and lots of conversation and encouragement as they worked. Figure 3 illustrates the quick progress a group made as they collaborated on putting the puzzle pieces together.

Figure 3. A group making quick progress. This group looked for edge pieces, and had a few members work on the "sky" while others worked on the "grass."

Most groups had their puzzles completed in about three minutes, as seen in Figure 4. Two groups, however, needed a little more time, as they had lenticular puzzles (the kind where the picture shifts as you move your field of vision from side-to-side.) These ended up being much more challenging to solve as compared to the more basic puzzles. All the groups were finished with their puzzles in five minutes or less.

As groups finished putting the pieces together, they were encouraged to think through *exactly* what they did to solve the puzzles, as that would be the next topic of discussion.

Figure 4. A group finishing their puzzle. This group quickly recognized the characters in the puzzle, which they noted helped give them context to figure out what the finished puzzle should look like.

Processing the Puzzling

After all the groups had their puzzles completed, they were invited to get specific about the strategies they used for solving their puzzles quickly. Some ideas that emerged included:

- Finding the corner pieces and edge pieces and sorting them out from the non-edge pieces.
- Flipping all of the pieces face-up so the colored sides were showing.
- Sorting the pieces by color.
- Using clues from the pieces to make sense of what the picture of their puzzle was.
- Using the shapes of the pieces to determine where they actually fit into the puzzle.
- Starting from the outside and working towards the inside of the puzzle.

- Intuitively subdividing the work so different people focused on different parts of the task.

As they shared their responses, the suggestions were affirmed and written down on the whiteboard. When they finished suggesting strategies for solving their puzzles, the participants had a few minutes to examine the list and see if they had other strategies to add. Hearing none, the instructor asked a clarifying question to the group: *"How did you know if you were successful in this task?"*

This question took a little time for them to answer. The immediate response for most groups was something like, "We just knew we were done." But this was probed a bit further: *"What does 'done' mean for this task?"* Eventually the participants agreed that they knew they were successful when they had used their pieces to complete a picture, and they had no leftover pieces. This was added to the whiteboard as well, which can be seen in Figure 5.

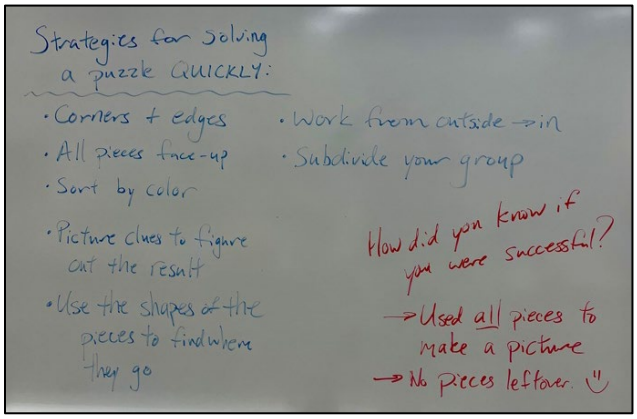


Figure 5. A photo of the whiteboard, including the ideas the group shared about their strategies for quickly solving their puzzles, and how they would know if they were successful.

The idea of this final question was to help them keep the focus on the main goal of the whole activity: solving an actual problem. This is central to the idea of computational thinking: using creative and critical thinking strategies to solve authentic problems by "drawing on concepts fundamental to computer science" (Wing, 2006, p. 33).

So…What is Computational Thinking?

At this point, the content took a turn to some direct instruction. The instructor explained to the participants that what they had been doing as they solved their puzzles was, in fact, applying computational thinking skills.

The instructor reiterated that the goal of computational thinking is not only to solve computer science problems, but to use structured, strategic thinking to attack authentic problems, including putting together a jigsaw puzzle. The four computational skills posited by McNicholl (2019) was also explained. The definitions presented to the participants for these four computational skills were:

1. Decomposition: Breaking the problem down into parts
2. Pattern Recognition: Seeing similarities and differences, and making connections
3. Algorithmic Thinking: Devising step-by-step processes to solve the problem
4. Abstraction: Identifying what actually needs to be done to solve the problem and filtering out noise

After this explanation, participants were encouraged to see if they could find examples of all four of these thinking skills in the way they approached solving their puzzles.

Groups immediately named pattern recognition as a key skill for solving the puzzle, which did not surprise the instructor, as so much of solving a jigsaw puzzle depends on finding patterns in colors and shapes of the pieces.

Decomposition was also quickly identified. Participants named the ways they had worked to break down the big task of "solve the puzzle" into component parts. This led most groups to immediately explain the algorithms, the steps, they used to solve the puzzle. Based on the strategies used for solving their puzzles, which were still listed on the whiteboard, most of the groups' algorithms were similar:

1. Flip all the pieces to be picture-side up.
2. Sort the pieces by color and/or shape.
3. Put the edge pieces together.
4. Fill in the middle pieces to complete the puzzle.

Abstraction, identifying what actually needs to be done to solve the problem and filtering out noise, was the most difficult of these four computational thinking skills for the participants to name, but eventually the instructor helped them list things like:

- Instead of trying to solve the whole problem all at once, focus on just one part of the problem, like finding the edge pieces
- Don't worry about sorting out every single piece before you start putting pieces together.

In this particular example of using computational thinking, the main goal of "use all the pieces to make a picture" was a helpful general rule, and that helped the participants conceptualize what abstraction is about: generalizing the particular. This sort of instructor facilitation is an important part of social constructivist learning theory (Brau, 2022).

A question several participants wanted answered was, "It seems like we're saying some parts of solving the puzzle are examples of several of these components of computational thinking at the same time. Is that right?" My response to them was that this is true; a component task like "start with the edge pieces" could be part of their algorithm for solving the problem, and it was *also* part of the way they decomposed the broader problem, and *also* one of the patterns they recognized in the pieces. That answer satisfied the adult learners in this group.

Wrapping Up the Lesson

To close out this lesson, the instructor reiterated that computational thinking is a skillset for solving complex problems, and while computational thinking is an essential part of computer science, it has broad application to many other disciplines. The participants were also encouraged to not think of computational thinking as "thinking like a computer," but rather a way of:

- Developing thinking skills to analyze complex problems.
- Learning to clearly articulate a plan.
- Designing a "good enough" plan to attack the problem.
- Refining that "good enough" plan through iterations.

The instructor also affirmed that these thinking skills are used by programmers as they are coding solutions to problems, which is the general intent of computational thinking within computer science. It was suggested that in future sessions the groups might work on making explicit connections between computational thinking skills and their own content areas. Hunsaker (2020) has a great many ideas for how computational thinking can be applied across a wide variety of disciplines, from more "obvious" options like math, engineering, and science as well as perhaps less-obvious content areas, including language arts, foreign language, social studies, music, family and consumer science, and physical education.

To close out the lesson, the learning targets for the session were revisited, and participants were asked to rate themselves on a 3-point scale (thumbs-up, thumbs-sideways, thumbs-down) of how they thought they did on each of the targets. Thumbs-up was the predominant response all around!

Critical Reflection

Overall, I was very pleased with how this PD experience unfolded for the participants. They came away with a practical, first-hand experience that helped them understand the fundamental principles of computational thinking at an intuitive level. The participants were able to apply the things they learned in this first session into other activities in later sessions as they learned about coding and educational robotics. We also continued to refer back to this shared puzzle experience as a way of making sense of the other things we did later on in this series of PD workshops. We continued to use the language of computational thinking skills (decomposition, pattern recognition, algorithms, and abstraction) throughout our work together.

Since using this approach of introducing computational thinking with puzzles, I have used this activity with several other groups of educators and found similar results. I have also used it with a group of 16 middle school aged students at a summer camp to help introduce the idea of computational thinking as part of an experiential learning opportunity of programming robots. I did not share that experience of puzzling with middle school students as the focus of this lesson because a colleague was the one actually leading that session, and I just received permission to try out this activity in the first 20 minutes with the middle schoolers to see how it went with a group of young adolescents. I did not substantially change anything about the way I presented the activity to the young adolescents from the way I presented it to the adult learners who participated in the activity described above. I used the same instructional moves to engage them in

completing the puzzles and facilitating a brief discussion with them afterward. I was encouraged how well it worked with the middle schoolers, and I think it merits more exploration. I'm convinced based on these experiences with both adults and young adolescents that this activity could work well in a wide variety of contexts and different age groups. After successfully implementing this learning representation multiple times, I continue to reflect upon the importance of connecting this experience to further learning about computational thinking. Instructors using it in a similar setting might be able to use it as described, and those adapting it to use with other age-groups or in specific content areas might be able to use it similarly. But in either case, having a clear connection to future instruction would be valuable. This learning representation will not do all the work of teaching learners about computational thinking.

That said, I do believe this jigsaw puzzle strategy is a useful way to begin learning about computational thinking, and to understand the key vocabulary related to computational thinking skills: decomposition, pattern recognition, algorithmic thinking, and abstraction. The unplugged activity described here should be considered just that, an intentionally low-tech introduction to computational thinking that should be robustly supported and expanded by a variety of other experiences with computing and programming.

References

Brau, B. (2022). Constructivism. In R. Kimmons (Ed.), *Education research: Across multiple paradigms* (Section 3.3). EdTech Books. https://edtechbooks.org/education_research/constructivismy

Campbell, L. O., & Heller, S. (2019). Building computational thinking: Design and making in teacher education. In J. Leonard, A. C. Burrows, & R. Kitchen (Eds.), *Recruiting, Preparing, and Retaining STEM Teachers for a Global Generation* (pp. 163-189). Brill.

Cansu, F. K., & Cansu, S. K. (2019). An overview of computational thinking. *International Journal of Computer Science Education in Schools*, *3*(1), 17-30. https://doi.org/10.21585/ijcses.v3i1.53

Hunsaker, E. (2020). Computational thinking. In A. Ottenbreit-Leftwich, & R. Kimmons (Eds.), *The K-12 Educational Technology Handbook* (Section 2.3). https://edtechbooks.org/k12handbook/computational_thinking

Klein, A. (2023, November 1). *Computer science courses are on the rise—But girls are still half as likely to take it.* Education Week. https://www.edweek.org/teaching-learning/computer-science-courses-are-on-the-rise-but-girls-are-still-half-as-likely-to-take-it/2023/11

Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, *30*(4), 883-908. https://doi.org/10.1007/s11191-021-00202-5

McNicholl, R. (2019). Computational thinking using code.org. *Hello World, 4*, 36-37. https://issuu.com/raspberry314/docs/helloworld04

Palincsar, A. S. (1998). Social constructivist perspectives on teaching and learning. *Annual review of psychology*, *49*(1), 345-375. https://psycnet.apa.org/doi/10.1146/annurev.psych.49.1.345

Vegas, E. & Fowler, B. (2020, August 4). *What do we know about the expansion of K-12 computer science education?* The Brookings Institution. https://www.brookings.edu/articles/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Yin, Y., Hadad, R., Tang, X., & Lin, Q. (2020). Improving and assessing computational thinking in maker activities: The integration with physics and engineering learning. *Journal of Science Education and Technology, 29*(2), 189-214. https://doi.org/10.1007/s10956-019-09794-8

About the Author

David J. Mulder serves as Professor of Education at Dordt University, where he teaches courses in educational foundations, STEM education, and educational technology in both the undergraduate Teacher Preparation Program and the Master of Education program. His research interests include technology integration, social presence in online learning, and digital citizenship. He can be contacted about this lesson at david.mulder@dordt.edu.

Acknowledgement

The development of this learning representation and the materials included herein are based upon work supported by the National Science Foundation under Grant No. (DUE-2243334). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9293

Unplugged to Plugged: An Introduction to Coding for Elementary School Children

Jessica Kerr and Theodore J. Kopcha, University of Georgia

Overview

This course introduced 4th and 5th graders to coding concepts like sequencing, loops, and decomposition through an unplugged-to-plugged learning sequence. Over four after-school learning sessions, students explored programming first through their bodies (i.e., unplugged), then in the block-based programming environment, Scratch (i.e., plugged). The goal was for learners to transition from concrete forms of programming to an abstract understanding needed for block-based programming. To achieve this goal, the plugged activities were intentionally designed around the concept of concreteness fading, where the unplugged programming challenge mirrored the plugged Scratch environment, allowing students to move from a concrete to more abstract understanding of computer science. The activities in the course drew from ready-to-use materials for computer science education (e.g., Code.org) as well as free, block-based coding websites (e.g., Scratch).

Topics: Block-Based Coding, Computational Thinking, Mathematics, Plugged Programming, Unplugged Programming

Time: Four one-hour class sessions

Materials

Materials for each class session are presented in the Learning Representation sections. General materials:

- Programming Skills assessment
- Presentation files (see daily Materials lists)
- 16oz (or similar) stackable cups
- Decks of playing cards
- Paper and markers (for planning)
- Computers with internet access
- Paper and markers (for planning)
- Scratch.org classroom account

Context-at-a-Glance

Setting
An after-school programming club held at an urban elementary school in the Southern United States.

Modality
Face-to-face

Class Structure
Four one-hour sessions. Students worked in self-selected pairs at large desks to complete programming activities.

Organizational Norms
The school district has a goal of strengthening computer science education at the elementary level.

Learner Characteristics
Ten students participated on a voluntary basis. At this school, 74% of the students were considered minority, with 27% (math) and 22% (reading) at or above the proficient level.

Instructor Characteristics
The instructor holds a Master's and Specialist's degree in Instructional Technology, with expertise in delivering hands-on, open-ended STEM learning.

Development Rationale
Unplugged approaches to coding can help elementary children apply concepts such as sequencing computer code, decomposing a larger challenge into manageable sub-tasks, and using loops and conditionals in a plugged environment.

Design Framework
The unplugged-to-plugged sequence draws on the idea of concreteness fading, in which children transition from a concrete, bodily understanding of concepts to more abstract representations (see Bruner, 1966 in Fyfe et al., 2014).

Setup

Setup for each session took 10-15 minutes. Prior to each session, the instructor prepared the materials needed for the planned activity. For the unplugged activity, this entailed gathering 10 plastic cups or a deck of playing cards, a couple sheets of paper, and a marker for each pair of students. For the plugged activity, students needed a computer, a couple sheets of paper, and a marker per pair. Along with gathering materials, the instructor pulled up the presentation slides necessary to teach that day's concept. Slides for each lesson are provided in the Materials section of each activity description below.

Since sessions were held face-to-face, the learning environment was designed to encourage discussions and collaboration on activities. To begin each session, the instructor had students sit on the rug. The instructor presented a mini lesson and activity for the day. Students worked on activities in groups of 2-3. To best support collaboration, each group worked at a table or pod of desks. At the conclusion of the work time, students were brought back to the rug in order to participate in a debrief and class discussion.

Standards

The course activities aligned with the national standards put forth by the Computer Science Teacher Association (CSTA, 2017) for grades 3-5, including:

- Break down problems into smaller, manageable subproblems (Standard 1B-AP-11).
- Create programs that include sequences, events, loops, conditionals, and variables (1B-AP-10).
- Identify and fix errors in a program or algorithm (1B-AP-15).

Context and Setting

The after-school club was offered to 4th and 5th graders in Spring 2024. Each one-hour session was held after school, once per week over a four-week period. The club was meant to supplement STEM learning activities at the school which, at the time of offering, did not include a focus on computer programming. While the district offered computer programming opportunities at the middle and high school, this club was among the first to be offered at the elementary level. It was therefore meant to attract students interested in computer programming while also preparing them for more advanced programming courses in the future. This context supported the use of an unplugged-to-plugged sequence in that elementary school children benefit greatly from having a concrete introduction to basic computing concepts before moving into block-based programming environments (Batni & Junaini, 2024).

Learner Characteristics

Of the 10 students who participated in the club, seven were boys and three were girls. For the majority of the students, it was their first time learning to program a computer. Participant demographics closely mirrored school demographics with minority students representing 50% of after-school participants. Students represented a variety of learning abilities with roughly 30% being advanced and another 30% having documented learning needs.

Instructor Characteristics

At the time of the after-school program, the instructor had 9 years of experience teaching at the elementary level. The instructor holds a Master's and Specialist's degree in Instructional Technology, with expertise in delivering hands-on, open-ended STEM learning activities.

Development Rationale

Computer science education has become increasingly important in elementary schools. This importance stems from the idea that early exposure to coding can help children view STEM careers like computer science as both attainable and desirable (Sullivan & Bers, 2018). However, many computer programming concepts can be challenging for young children because they are abstract and require new forms of thinking and reasoning (Li et al., 2022). Unplugged coding activities offer a solution to this problem. Unplugged activities are those that use tangible, non-digital objects to introduce computing concepts to children (Lee & Junoh, 2019, Zhang et al., 2024). The goal is to introduce children to coding skills through hands-on, concrete experiences that can serve as a foundation for more abstract

approaches to programming in the future (Caeli & Yadav, 2019, Huang & Looi, 2021, Wong, 2022). Research on unplugged activities suggests that they are particularly effective for teaching computer programming to elementary-aged children (Zhang et al., 2024).

With this in mind, this course was intentionally built using an unplugged-to-plugged approach to teaching coding. The overall goal of the course was for learners to transition from the concrete understanding of programming to a more abstract understanding that is needed for advanced programming. To achieve this goal, plugged activities were intentionally designed such that the surface features of the programming challenge mirrored those of the unplugged environment. The goal to mirroring the features of the challenge was to help the learners apply the coding concepts learned in a concrete way (i.e., during the unplugged activity) in more abstract ways in the plugged environment. The unplugged activities in the course drew from ready-to-use materials for computer science education provided by Code.org; plugged activities were created by the lead author.

Design Framework

The framework behind the design draws on Bruner's (1966) concept of concreteness fading (see Fyfe et al., 2014). The basic idea behind concreteness fading is that children learn better when they are first introduced to concepts in a concrete, physical way. This is often accomplished using objects that, when handled and manipulated, engage the learner in the desired concepts. Those objects can be slowly withdrawn and replaced with more abstract representations of the concept. Fyfe et al. (2014) offered the simple example of the number two, which could be represented first through two apples, then two dots, which are more symbolic, and finally with the written numeral (2).

The concept of concreteness fading is considered an important part of learning in the unplugged-to-plugged approach in teaching coding concepts. The general idea is that a child can more easily understand abstract coding concepts when they use everyday objects, including their bodies, to visualize the results of abstract coding concepts (Batni & Junaini, 2024; Kwon et al., 2022). In the unplugged environment, a child might learn about a variable by filling and emptying a box or container with a different number of objects that achieve a goal. Later, the concrete idea of filling and emptying the box could be used to introduce a variable as a container that can be filled. This would support more abstract programming concepts in which values are temporarily assigned to or held within a variable (e.g., x = 2).

Code.org contains a variety of unplugged activities, each of which addresses a different set of skills and concepts associated with coding. Each of those activities have the potential to serve as a concrete introduction to coding that could later be used to teach more abstract forms of programming.

Learning Representation

One activity was conducted in each of the four sessions. The following learning representation section is organized in four activities. Overall, across the four sessions, objectives were created for students to meet. For the entire course, learners will be able to:

- Create sequences, conditionals, and events that could command a human and/or computer to accomplish a given task.
- Break down problems into more manageable sub-tasks.
- Identify and fix errors in a sequence or program.

Each activity had a specific focus that supported these larger goals:

- Activity 1: Unplugged sequencing and looping
- Activity 2: Unplugged conditionals
- Activity 3: Plugged sequencing
- Activity 4: Applying sequencing, loops, and conditionals in a personal project

Activity 1: Cup-Stack Activity

The first activity was a modified version of the Code.org (n.d.a) lesson, *My Loopy Robotic Friends Jr*. The activity challenged students to use basic commands (e.g., up/down, left/right) to create a sequence that would arrange plastic cups into a specific design. To complete this challenge, the students needed to create a sequence of commands that another person could follow that would achieve their goal. This introduced the concepts of

sequencing and looping to the learners. The instructor modified the original Code.org lesson in several ways. First, the instructor only used two of the 8 cup-stack designs contained in the lesson to fit the activity in one hour. Additionally, the instructor streamlined the presentation provided by Code.org to align with the two cup-stack designs.

Materials

- My Loopy Robotic Friends Jr. by Code.org (n.d.a)
- Code.org presentation modified for this activity
- Cup stacking patterns (see Figures 1 and 2)
- 16oz plastic cups
- Markers and paper

Introduction (10 minutes)

The activity began with introducing the first of two challenges to the whole group. The first challenge was to arrange two cups with a space between the two (Figure 1). This task was meant as an introduction to the basic commands associated with the task. The instructor then spent five minutes walking the entire group through a set of initial commands associated with the task, such as picking up a cup, putting down a cup, and moving a cup left or right. Students were instructed to document their code, using arrows to denote their coding commands (e.g., an up arrow for an upward movement, a left/right arrow for side movements). The instructor demonstrated how those commands were associated with the movement of a cup. The instructor also encouraged students to modify these commands or create new ones based on their own individual solutions to the problem.

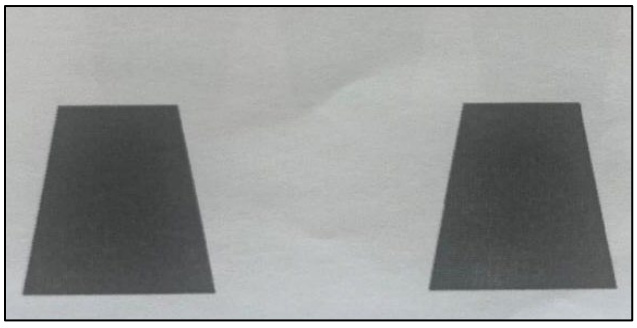


Figure 1. First cup-stack challenge. Note: Image reproduced with attribution under Creative Commons BY-NC-SA 4.0 license.

First Challenge (15 minutes)

The learners then worked in pairs to plan a solution to the first challenge. They were tasked with writing out their code using markers on a sheet of paper. When they finished, the teacher read the written code aloud to the pair as the students enacted it. If the code did not work, the students were tasked with fixing it and re-checking the revised solution with the instructor.

Second Challenge (15 minutes)

The second challenge was more complex than the first. The design was an arrangement of five cups, with three on the base layer and two on top of those three, as if forming a pyramid without the top cup. When students completed the first challenge, they worked on the second challenge until they created a solution or time ran out.

Students used the skills they acquired in the first challenge to help them code a solution for the second challenge. However, as students were coding, they ran into the challenge of determining how to write the code necessary to create the top layer of the design. Students realized that to create a pyramid shape, the top cup had to balance between the two cups below. Students had to work together to figure out how to adjust the "side" unit used in the previous challenge. This resulted in students building off of the given commands to create their own "small side" or half steps.

The instructor walked around the room listening to students' conversations and asking guiding questions as needed. When they finished, the teacher read the written code aloud to the pair as the students enacted it. If the code did not work, the students were tasked with fixing it and re-checking the revised solution with the instructor. As students finished, they were challenged to think of how they might shorten their code, encouraging them to use loop commands. Though loop commands were not explicitly taught during the introduction, the instructor introduced the coding concept to pairs who were ready for the challenge.

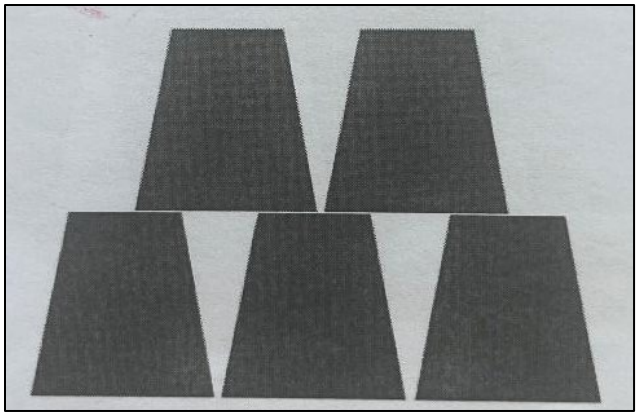


Figure 2. Second cup-stack challenge. *Note: Image reproduced with attribution under Creative Commons BY-NC-SA 4.0 license.*

Whole Group Debriefing (15 minutes)

The instructor then led the whole group through a debriefing. Particular emphasis was placed on the challenges students faced writing code for the second design challenge. The instructor began by asking students to share what they found easy and challenging about the activity. For example, students found it challenging to include all of the necessary steps to recreate the cup stack design. Many students discussed the need to test and adjust their code multiple times.

The instructor then guided the conversation to focus on how students wrote code for the top layer in the second challenge. The instructor purposefully highlighted students' use of "small side" or half side steps in order to move the cup a shorter distance. This emphasis aligned with the idea of concreteness fading in that it drew the students' attention to the way that the physical width of the cup could be manipulated to help measure distance and plan a sequence of moves.

Activity 2: Conditionals Activity

The second activity was a variation of the Code.org (n.d.b) lesson, *Conditionals with Cards,* which challenged students to invent a set of rules for a game that awarded points when specific conditions were met. This activity used a deck of cards as the primary concrete object. Students were tasked with developing different conditionals like *if-else* and *if-else if-else* that used the features of a card to award points when specific conditions were met. The goal was to run the conditionals through the entire deck of cards and see who could accumulate the most points. For example, students might create a conditional that awards a point to one team when a red card is drawn and the other when a black card is drawn. They would then take turns drawing cards and assigning a score based on their conditional statement.

Materials

- Conditionals with Cards by Code.org (n.d.b)
- Code.org presentation modified for this activity
- Several complete decks of cards
- Markers and paper

Introduction (10 minutes)

The activity began with introducing the basic ideas of an *if-else* conditional. The instructor used the example from the Code.org (n.d.b) lesson, which commanded the class to clap *if* a King is drawn, *else* everyone says "Awww!" After practicing this conditional, the instructor introduced the idea of adding *else if* to the conditional (e.g., *if* the card is a King, clap; *else if* the card is a Queen, clap). When the students were able to follow each conditional, the instructor then had them create their own unique conditionals for a deck of cards.

Challenge (15 minutes)

The learners were challenged to create their own conditionals that used the features of the deck of cards to assign points. Students worked on their own to create a list of conditionals and the number of points that was awarded based on the card drawn (e.g., if the card color is red, +1 point; else -1 point). Once students created their conditionals, they chose a partner to share their conditional statements and play their card game (which led to the whole group debriefing).

Whole Group Debriefing (15 minutes)

The instructor then led the whole group through a debriefing. Particular emphasis was placed on how students chose to create their conditionals. The instructor began by giving students an opportunity to

share the conditional statements they created with their partner. This created an opportunity to see how other students developed conditionals in different and unique ways. The instructor then asked students to share how they created their conditional and how they chose what points to award each conditional statement. Many students discussed how they assigned a greater point value to cards that were less common in the deck (e.g., face cards = 100 points). Students were given time to discuss how all the conditional statements were the same or different.

Activity 3: Cat-Jump Activity

The third activity challenged students to make the cat character (sprite) in Scratch (https://scratch.mit.edu/) move forward and appear as if it were jumping over a rock. Scratch is a project of the Scratch Foundation (n.d.), in collaboration with the Lifelong Kindergarten Group at the MIT Media Lab. It is available for free at https://scratch.mit.edu The environment uses a block-based approach to programming that makes computer programming a visual experience that is more accessible to younger children than text-based programming environments. Figure 3 displays the starting point for the cat and the rock in the Scratch environment used in this lesson.

Figure 3. Scratch cat character jump challenge with a rock. Note. Image reproduced under Creative Commons Attribution-ShareAlike license. Scratch is a project of the Scratch Foundation (n.d.), in collaboration with the Lifelong Kindergarten Group at the MIT Media Lab. It is available for free at https://scratch.mit.edu

This challenge was developed by the instructor to mirror the unplugged Activity 1: Cup-Stack Activity conducted earlier in the sequence. Like the Cup-Stack activity, the students were asked to create a sequence of commands that moved an object left and right or up and down. The activity also challenged students to move towards more abstract forms of programming in that they were tasked with completing the challenge using the block-based environment in Scratch.

Materials

- Scratch.org programming environment on display with cat and rock arranged as shown in Figure 2
- Markers and paper
- Computer with access to Scratch

Introduction (10 minutes)

The activity began by introducing the challenge and directing students to start a new project in the Scratch environment. The instructor demonstrated how to position the cat and add the rock in Scratch to begin the challenge. Finally, the instructor provided an overview of the code associated with *motion* (e.g., move ___ steps; turn ___ degrees) and *control* (e.g., wait ___ seconds; repeat ___). The instructor showed how to use different blocks to assemble a simple sequence that moved the cat in each direction.

Planning (10 minutes)

The learners were then given 10 minutes to plan a sequence of commands that would move the cat towards the rock and appear to jump over the rock. The students created these plans non-digitally, with markers and paper. This was done intentionally so that students were challenged to use the skills they had developed previously during the unplugged activity on the current challenge.

Programming (20 minutes)

The learners were given 20 minutes to code a sequence of commands, using Scratch, that would make the cat appear as if it is jumping on the rock. Students were encouraged to use their planning sheets to help them create their code. As students worked on creating their code, the instructor monitored students and assisted as needed. When students got stuck, the instructor would ask prompting questions to guide students towards a solution. If students still struggled, the instructor provided direct instruction. If students finished early, they were challenged to add extra features to their Cat-Jump code (e.g., have the cat make a noise if it touched the rock).

Whole Group Debriefing (15 minutes)

The instructor then led the whole group through a debriefing. Particular emphasis was placed on the challenges of transferring their written code during the planning period to block-based code on Scratch. The instructor began by asking students to share what was challenging about the Cat-Jump activity. The instructor prompted students to share how they found solutions to the challenges they faced. For example, students struggled with the right number of steps the cat needed to move to reach the rock or how to make the cat move up and down. The instructor guided the conversation such that a connection was drawn between strategies for moving the cup in the Cup-Stack activity and strategies used in the Cat-Jump activity. If students were unable to resolve their challenges, the instructor provided other students the opportunity to provide help.

Activity 4: Personal Projects

In the final activity, students were challenged with creating a project in Scratch that they found personally interesting. Students could select from the project ideas displayed on the Scratch tutorials page (e.g., play music, act out a scene); however, they were required to use the coding concepts from the unplugged environment to accomplish their goals. This included incorporating loops to repeat a function or action and conditionals to evaluate and respond to specific conditions.

Materials

- Scratch.org tutorials page
- Scratch.org programming environment on display
- Programming Skills assessment
- Markers and paper
- Computer with access to Scratch

Introduction (15 minutes)

The activity began with students taking the Programming Skills assessment. Students were given approximately 10 minutes to complete the assessment. The instructor then introduced the challenge and showed students some sample projects displayed on the Scratch website. The instructor helped the students identify a goal or project of interest, then quickly reviewed the coding concepts of loops and conditionals. Although students had previously been introduced to Scratch codes associated with *motion* (e.g., move ___ steps) and *control* (e.g., wait ___ seconds; repeat ___), they had not been introduced to codes for *events*, which held if-then conditionals. The instructor showed how if-then sequences worked in Scratch.

Planning (10 minutes)

The learners were given 10 minutes to plan out their projects and project goals. They worked in self-selected pairs. During this time, the instructor met with each pair to ensure that at least one loop and one conditional were included in their planning. The students created these plans non-digitally, with markers and paper, so that they were challenged to use the skills they had developed previously during the unplugged activity on the current challenge.

Programming (20 minutes)

Once the planning was completed, the students began programming their projects in Scratch. An important part of the instructor activity during this part of the lesson was being responsive to the students' projects and the code needed to accomplish certain goals. For example, students who were interested in making music were encouraged to explore codes under the category of *sound* (e.g., play sound ____), whereas students whose projects told a story explored codes under *looks* (e.g., next costume, say ___ for ___ seconds). See Figures 4 and 5 for example student projects.

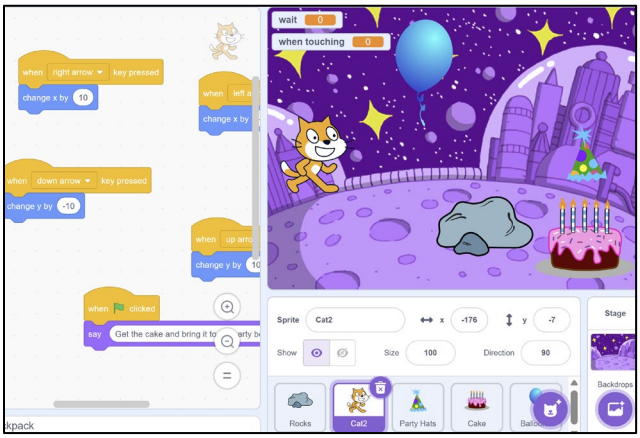


Figure 4. Example of student project.

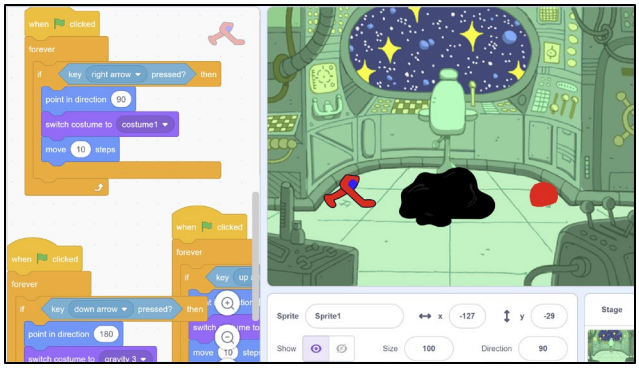


Figure 5. Example of student project.

Whole Group Debriefing (10 minutes)

The instructor then led the whole group through a debriefing. Particular emphasis was placed on providing students an opportunity to share new coding tools they learned while working on their projects. The instructor began by providing students an opportunity to share their personal projects. As students shared their projects, the instructor asked students to share one new-to-them tool and to demonstrate how the tool works (e.g., play sound _____, next costume, If_____ is touched, say _____).

Critical Reflection

The unplugged-to-plugged course sequence was offered once to students in Spring 2024. The sequence is currently being improved based on the results of the implementation, with plans to implement the revisions in Spring 2025. Our reflection on the results is presented below.

Implementation Evaluation

Learning Objectives

The implementation successfully met the learning objectives set forth for the course sequence. The Programming Skills assessment evaluated student learning. The test contained multiple-choice questions that addressed the following coding concepts:

- Basic understanding of Scratch blocks (2 questions).
- Variables (1 question).
- Sequencing (1 question).
- Use of conditionals (1 question).
- Use of loops (1 question).
- Two open-ended questions in which students had to identify and fix errors in a sequence of code.

The total attainable score on the test was 10 points, with 1 point for each multiple-choice item and 2 points for each open-ended question. Open-ended questions were evaluated on a three-point scale, with scores reflecting that students were either successful (2 points), partially successful (1 point), or not successful (0 points). Figure 6 provides an example of a student's response to one of the open-ended questions. In this question, students were asked to fix the code so that the cat would make a square with the prompt, *Scratch is trying to draw a square. How do I need to change the code so that Scratch can make a square?* The student responded, "needs to turn 30 degrees idk :)." The response shown in Figure 6 received a score of 'partially successful' (1) because the student knew that the number of degrees needed to be adjusted but did not use the appropriate number of degrees to create a square.

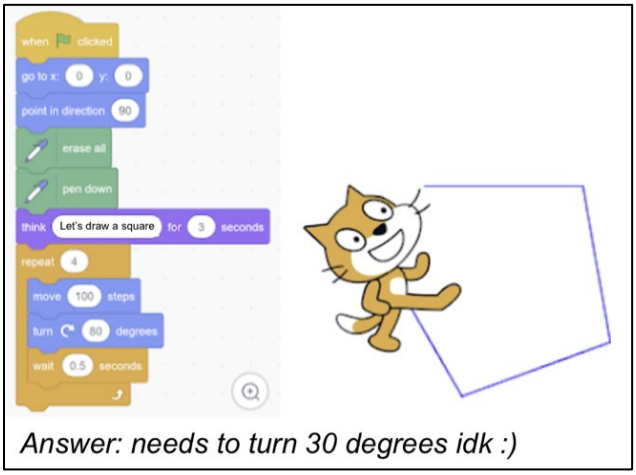


Figure 6. Student response to an open-ended question.

The average test score was 70% (7/10 points) at the conclusion of the instruction, suggesting that, overall, the students did learn the basic coding concepts covered in the instructional sequence. Point loss was mainly due to the fact that most students (60%) were only partially successful or unsuccessful on the open-ended questions, with a smaller sub-group (20%) being successful on both questions. Students also struggled with identifying variables in a block of code;

that item had an average score of 40%. Given that the students had little to no prior programming experience, this result suggests that the instructional sequence did help the students develop a basic understanding of coding concepts.

Concreteness Fading

One goal of the instruction was to support the students in moving from the concrete understanding of coding developed during unplugged activities towards more abstract forms of thinking and coding. Observations from the instructor suggested that this goal was partially achieved. For example, the Cup-Stack activity encouraged the students to use the physical properties of the cup to estimate and plan the movements of the cup. A left or right movement was the width of a single cup, and the up/down movements incorporated the height of a cup. This concept carried over into the plugged environment, where the students used the width and height of the cat sprite to plan the movements needed to 'jump' over the rock.

As the students engaged with Scratch, however, they began moving away from using the physical properties of the cat towards using the coordinate system within Scratch. One group began setting the Y coordinate directly, rather than moving up or down the height of the cat. Two other groups did the same with the X value. Rather than repeatedly moving small distances, they began combining steps into a single value that reflected the entire movement. This suggests that the students began moving away from the concrete representations of coding developed during the unplugged activity as they engaged with block-based programming in Scratch.

Lessons Learned

One thing that we learned was that not all unplugged tasks lend themselves to concreteness fading as described in the literature. The literature suggests that the fading is gradual, from concrete to partially concrete/abstract to fully abstract. In practice, it was messier than this. Some students immediately jumped from using the width and height of an object to directly programming the X or Y value in Scratch, while others did not. Likewise, most students were able to jump to using *if-then* statements in Scratch without needing a deck of cards. This was likely because the *if-else* activity was already somewhat abstract, requiring students to engage more conceptually with conditionals than the physical movements associated with the Cup-Stack activity.

The messy nature of concreteness fading suggests that instructors could do more to support the transition from unplugged to plugged programming. In the next iteration, we intend to challenge students to explain specific connections between their concrete understanding and the skills needed for block-based programming in Scratch. This could be done during instruction, where instructors ask probing questions of each pair as they plan their code in the plugged environment. It could also happen after coding in Scratch, where participants reflect on connections between their concrete understanding of coding and more advanced skills used in Scratch.

Another thing we learned is the importance of intentionality while structuring lessons. In this first iteration, lessons were designed to introduce concepts in an unplugged environment then have students transfer this knowledge in the plugged environment. Therefore, the first half of the lessons were all designed in the unplugged environment. However, when students transitioned to the plugged environment, we observed that they struggled initially because they were not only tasked with applying previously learned coding concepts but learning a new programming tool as well. Students were attempting to do too much at one time. In the next iteration, we intend to plan lessons so that students alternate between the unplugged and plugged environment. This structure will allow students to gradually learn coding concepts while becoming familiar with Scratch's programming platform. It would also be helpful to give students some time to explore Scratch more freely before introducing a plugged challenge. This would help them get familiar with the different types of blocks in Scratch and the possible ways they can interact with the Scratch Cat.

References

Batni, B., & Junaini, S. N. (2024). Redefining computational thinking: Synergizing unplugged activities with block-based programming. *Education and Information Technologies.* https://doi.org/10.1007/s10639-024-12869-8

Caeli, E. N., & Yadav, A. (2019). Unplugged approaches to computational thinking: A

historical perspective. *Tech Trends, 64*, 29-36. https://doi.org/10.1007/s11528-019-00410-5

Code.org. (n.d.a). *Lesson 7: My loopy robotic friends Jr.* https://studio.code.org/s/coursec-2022/lessons/7

Code.org. (n.d.b). *Lesson 12: Conditionals with cards*. https://studio.code.org/s/coursed-2022/lessons/12

Computer Science Teachers Association. (2017). *CSTA K-12 Computer science standards.* https://csteachers.org/k12standards/interactive

Fyfe, E. R., McNeil, N. M., Son, J. Y., & Goldstone, R. L. (2014). Concreteness fading in mathematics and science instruction: A systematic review. *Educational psychology review*, *26*, 9-25. https://doi.org/10.1007/s10648-014-9249-3

Huang, W., & Looi, C. (2021). A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education. *Computer Science Education, 31*(1), 83- 111. https://doi.org/10.1080/08993408.2020.1789411

Kwon, K., Jeon, M., Zhou, C., Kim, K., & Brush, T. A. (2022). Embodied learning for computational thinking in early primary education. *Journal of Research on Technology in Education, 56*(4), 410-430. https://doi.org/10.1080/15391523.2022.2158146

Lee, J., & Junoh, J. (2019). Implementing unplugged coding activities in early childhood classrooms. *Early Childhood Education Journal, 47*(6), 709-716. https://doi.org/10.1007/s10643-019-00967-z

Li, F., Wang, X., He, X., Cheng, L., & Wang, Y. (2022). The effectiveness of unplugged activities and programming exercises in computational thinking education: A meta-analysis. *Education and Information Technologies, 27*, 7993-8013. https://doi.org/10.1007/s10639-022-10915-x

Scratch Foundation. (n.d.). https://scratch.mit.edu

Sullivan, A., & Bers, M. U. (2018). Robotics in the early childhood classroom: Learning outcomes from an 8- week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, *26*, 3-20. https://doi.org/10.1007/s10798-015-9304-5

Wong, G. K. W. (2023). Amplifying children's computational problem-solving skills: A hybrid-based design for programming education. *Education and Information Technologies, 29*, 1761-1793. https://doi.org/10.1007/s10639-023-11880-9

Zhang, Y., Liang, Y., Tian, X., & Yu, A. (2024). The effects of unplugged programming activities on K-9 students' computational thinking: Meta-analysis. *Education Tech research and Development, 72*, 1331-1356. https://doi.org/10.1007/s11423-023-10339-5

Support Materials

- Scratch.org tutorials page

All materials from Code.org (e.g., images, curricular materials, presentations) were reproduced with attribution under a Creative Commons BY-NC-SA 4.0 license. Materials from Scratch.org were reproduced with attribution under a Creative Commons Attribution-ShareAlike license.

About the Authors

Jessica Kerr is a doctoral student at the University of Georgia. Her research focuses on unplugged-to-plugged blended approaches to coding instruction through an embodied approach to cognition. Aside from her graduate studies, Kerr teaches in an elementary school in Georgia.

Theodore Kopcha is a former secondary mathematics teacher whose current research explores the integration of innovative technologies into the classroom and connecting them to mathematics. Over the past two decades, Dr. Kopcha has worked closely with both pre-service and in-service teachers and has published extensively on teacher professional development and technology integration.

SHARING & MODIFICATION PERMISSIONS

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9339

Integrating CT in Science Methods: Advancing Practice and Pedagogy

Ugur Kale[1] and Yuahnua Wang[2]
[1]Indiana University, [2]West Virginia University

Overview

Despite the importance of computational thinking (CT) as a problem-solving process (Wing, 2008) and the growing spread in teacher education (Yadav et al., 2017), existing initiatives for preservice teachers (PSTs) tend to focus on the computer science domain without making explicit connections to disciplinary classroom settings and promoting critical perspectives. As a cohesive unit, this learning representation aims to assist PSTs in integrating CT into their work as they design and implement science-focused lessons.

Centered around a contextual issue: accessing, growing, and sustaining food, this learning representation employs 2D and 3D block-based programming languages coupled with unplugged activities that demonstrate CT practices, processes, and concepts. PSTs' group designs, lesson modifications, and full lesson plans provide opportunities for assessment.

Topics: Computational Thinking, Science Education

Time: 18-20 hours

Materials

- Whiteboard and project connecting a computer
- Session 1 Slide Deck: Access Food
- Session 2 Slide Deck: Grow Food
- Session 3 Slide Deck: Sustain Food
- Group Lesson Design Template
- Modification Guideline
- Lesson Plan Template
- CT Lesson Implementation Rubric
- Garden-based learning supplies; plug trays, soil, cucumber seeds, LED grow light and heat mat
- Minecraft Education
- Scratch Account

Context-at-a-glance

Setting
Undergraduate science methods courses in a teacher education preparation program in a largely rural state nested in the Appalachian region, U.S.

Modality
Face-to-face

Class Structure
The class meets twice a week. The CT component is addressed in three main sessions, each of which takes two weeks, offering knowledge, practice, and application. The room has six desks stationed mimicking a public-school science classroom layout.

Organizational Norms
Accredited by the Council for the Accreditation of Educator Preparation, the teacher education program supports preservice teachers' (PSTs) teaching knowledge via coursework and field experiences.

Learner Characteristics
Third-year undergraduate PSTs with limited CT knowledge. More than 93% were female and white. Slightly more than 25% are first-generation students.

Instructor Characteristics
The instructors have instructional design and science education expertise with good CT knowledge and familiarity with block-based programs.

Development Rationale
The learning representation addresses the existing CT initiatives' limited connection to disciplinary classrooms and critical perspective by providing opportunities for PSTs to relate CT to their practices.

Design Framework
TPACK, Computational Thinking Practices

Setup

The setup for the first and second sessions takes approximately 15 minutes while the third session, which involves the use of Minecraft Education additionally requires the installation of the software in advance. Although PSTs can install it on their laptops, a laboratory setting with computers already installed with Minecraft Education would ensure everyone has access to it timely in case of any technical difficulties with individual laptops. The instructors should review the slides and the templates prior to implementing the activities effectively. The schedule of the sessions can be adjusted according to the instructors' course timeline.

Standards

This learning representation aligns with the International Society for Technology in Education (ISTE) standard "*2.6.c Teach Computational and Design Thinking: Create learning opportunities that challenge students to use a design process and computational thinking to innovate and solve problems*" (ISTE, 2017).

Context and Setting

As a means to solve problems via concepts and techniques from the computer science field, CT has been considered an essential skill and explored in educational research (Tikva & Tambouris, 2021). The increasing interest and applications of emerging tools using Artificial Intelligence (AI) for education even further amplified CT's significance because a meaningful understanding of such systems, key to their effective use, also requires thinking computationally (Dohn et al., 2022; Hsu & Chen, 2024). Specifically, engaging in CT can entail making artifacts such as designing systems in situated environments reflecting the constructionist perspective (Harel & Papert, 1991). This process starts with a problem, which can be decomposed into smaller components, among which patterns can be identified helping to abstract a generic solution, followed by describing algorithms to automate and analyze the generalized solution (Barr et al., 2011; Grover & Pea, 2013; Wing, 2008). These elements closely mirror the problem-solving process including understanding and representing the problem, planning solutions, and executing solutions and self-regulating (Jonassen, 2000; Mayer & Wittrock, 2006; Polya, 1957).

Despite CT's importance as a problem-solving process (Wing, 2008) and its growing spread in teacher education (e.g., Adler et al., 2023; Yadav et al., 2017), two main issues remain. First, the current CT-focused initiatives solely focus on the computer science domain without explicit connections to disciplinary classroom settings (Yadav, et al., 2018). This is problematic because as the key emphasis of the Technological Pedagogical and Content Knowledge (TPACK) framework, a successful integration of technological innovations in teaching requires not only the knowledge of the innovation but also the knowledge of how to use it in particular subject-matters (Mishra & Koehler, 2006). As such, without opportunities to understand and practice CT in relation to their teaching and particular content, PSTs' future implementations with CTs will likely be impeded (Kale et al., 2018; Rich et al., 2019).

There are efforts to connect CT to various subjects (Barr & Stephenson, 2011). To clarify the meaning of CT specifics to mathematics and science practices, for instance, Weintrop et al. (2016) developed a taxonomy with four main categories that can elicit CT processes via various practices:

1. Data Practices (e.g., collect, create, analyze)
2. Modeling and Simulation Practices (e.g., use, assess, design, and construct)
3. Computational Problem Solving Practices (e.g., abstracting and debugging)
4. Systems Thinking Practices (e.g., understanding relationships within a system)

Although the practices look distinct, they are not mutually exclusive, and the last two categories reflect rather a problem-solving process that can be engaged simultaneously with the practices. For instance, learners may analyze data (Data Practices) or construct models (Modeling and Simulation Practices) as they try solutions to a given problem (Computational Problem-Solving Practices) that may involve understanding the relationship among multiple factors (Systems Thinking Practices). Thus, differentiating (1) Data and (2) Modeling and Simulation as CT practices from (3) Computational Problem Solving and (4) Systems Thinking as CT processes, which involve decomposition, pattern recognition, abstraction, algorithms & automation,

and analysis, will be a more accurate approach (see Figure 1 for a modified taxonomy).

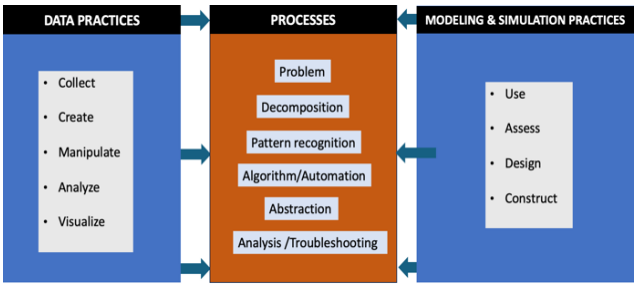


Figure 1. CT practices supporting CT processes.

Second, CT has been viewed mainly as a cognitive outcome, but a rather critical perspective of CT has emphasized self-expression, connection, and questioning (Brennan & Resnick, 2012) and been recently considered as a potential way to engage learners in the social, economic, political challenges of the world that directly impact them (Kafai et al., 2019; Tissenbaum et al., 2019). However, only a few examples exist that focus on such contextual issues via CT such as political resistance in the form of Latino students' storytelling (Aghasaleh et al., 2019) and the design of an environmental awareness mobile app (Tissenbaum et al., 2019).

To address these two issues and to help PSTs gain competence with and increase their interest in CT, a critical perspective of CT was infused into a science methods course offered in an undergraduate elementary teacher education program. Approximately seventy-five PSTs in their third year in the teacher education program enrolled in the course and more than 93% were female and white. According to the West Virginia University Office of Data and Analytics (2024), more than a quarter of undergraduate students enrolled in the college identified themselves as first-generation students. Further, the PSTs' responses to a question in a pre-survey instrument indicated they were between slightly aware and moderately aware of the social, economic, or political challenges in the state. PSTs' responses to this pre-survey instrument (see Kale et al., 2024; Kale & Wang 2024) also showed their varying level of competence in CT, perception of CT's value, interest in CT, and use of CT in teaching. On average, PSTs either agreed or strongly agreed that CT can be relevant to their teaching practices. They also found CT interesting but did not feel competent to integrate it in teaching. A majority reported limited frequency of CT usage (e.g., a few times a semester).

The course focused on in-class activities helping PSTs unpack Next Generation Science Standards (NGSS), assess student thinking, modify an existing science lesson and teach it in their placement schools, and reflect on their teaching. PSTs spend around 10 hours per week in placement classrooms during the semester.

As part of a federally funded project, this learning representation was developed as a CT unit by the authors and various experts such as elementary school teachers and technology integration specialists. The unit centered around a contextual issue - accessing, growing, and sustaining food. For context, the state where the program is located is largely rural, nestled in the Appalachian Mountains in the U.S., and faces an abundance of challenges. Besides the poor health status and low median income, the area this lesson took place is ranked one of the highest in the nation regarding the uncertain availability of nutritionally adequate and safe food at the household level (Gundersen et al., 2020). And, state residents with limited food access tend to be located in certain locations, often labeled as "food deserts" where temporal, spatial, and socio-economic factors act as barriers (Wilson et al., 2017).

Focusing on the food access issue via CT, three sessions were provided, and a Garden-based Learning (GBL) experiment was conducted. The sessions involved two classroom days, each of which lasted 75 minutes, and took pace in a STEM lab. Originally designed to mimic the layout of a science classroom, it has six desks encouraging group work. One session, focusing on the use of Minecraft Education, as described in the next section, was completed in a computer lab where the software was installed onto computers in case the program does not work in the PSTs' laptops.

Learning Representation

During this lesson, italic text identifies prompts for the learners.

Session 1: Access Food

This session is a total of six hours that was spread across four in-class days (1.5 hours each) and occurred during Weeks 2-3 of the semester.

INTRODUCTION (1.5 HOUR)

To start this learning representation, the instructor should first demonstrate to PSTs how the CT approach can be used to understand and deal with *Food Access*. The Session 1 Slide Deck: Access Food (PPT; slides 2-64) includes a detailed explanation of the steps covered for this first session.

In Session 1, the instructor started with a question, "*What's the sum of numbers between 1 and 200?*" and asked the PSTs to think about how to solve it without using any calculators. This was just a brief example about demonstrating the main processes of CT, namely, decomposing the problem (e.g., 1, 2, 3…… 198, 199, 200), identifying the patterns (e.g., 1+200 = 201, 2+199 = 201, 3+198 = 201) and solving the given problem (e.g., Solution = 201*100 pairs), abstracting a generic solution (e.g., for a given number *N*, the Solution = N+1*(N/2)), and developing the algorithm (e.g., outlining steps for a computer program to follow), and analyzing its effectiveness (e.g., testing with different numbers).

The instructor then introduced CT processes as a means to solve problems and highlighted how programming tools may provide opportunities to practice these processes. Two unplugged activities (see Figure 2 and Figure 3) were used to help the PSTs practice the development of algorithms without any technology tools.

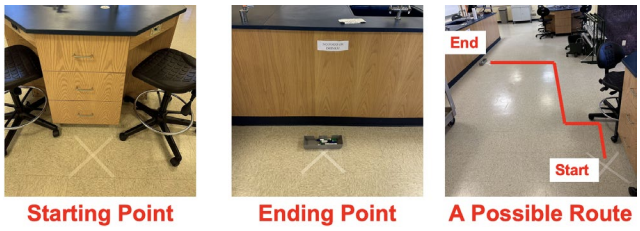


Figure 2. Activity 1 (see Slide 24 in Session 1 Slide Deck: Access Food (PPT)).

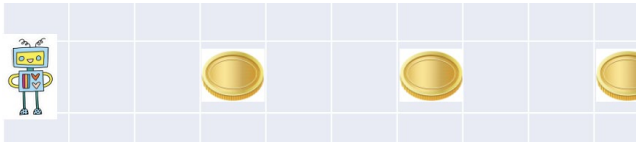


Figure 3. Activity 1 (See slide 26 in Session 1 Slide Deck: Access Food (PPT)).

The first unplugged activity required two volunteers: one PST to act as a robot tasked with moving a marker from one location of the room to another, and another PST to provide directions to the "robot". The second unplugged activity required everyone in the class to come up with the steps a robot, displayed on board, needs to follow in order to collect the displayed coins. After a discussion of what CT processes are addressed in these two activities, another activity followed the same format but rather focused on a scientific concept: the pumpkin growth cycle where PSTs worked on correctly sequencing the given images of the cycle. A few examples of scientific topics such as the state of matter and the water cycle were also discussed regarding the specific CT processes elicited.

To help PSTs further engage in their understanding of the CT processes, the instructor asked them to work in groups to draw an equilateral triangle on a paper and list the steps followed. In each group, one person controlled the pen while the others provided directions to follow. Having shared their drawings and steps (e.g., number of sides to draw and angles to turn), the groups were prompted to draw a square and again share their drawings and steps. This was followed by another prompt where PSTs were asked to recognize the relationship between the kind of shape and its properties (see Figure 4). The PSTs also discussed CT processes elicited in these examples.

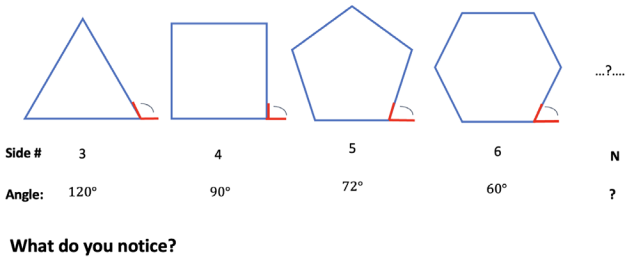


Figure 4. Properties of chapes compared (See Slide 50 in Session 1 Slide Deck: Access Food (PPT)).

After the discussion, the PSTs were prompted to draw the shapes in Scratch for the next fifteen minutes. Prior to the class, PSTs were asked to create accounts in Scratch, get familiar with the interface, and practice some of the codes so they could be ready for this part of the activity. For this, they were guided to use the pen blocks to come up with the algorithm for drawing a triangle first then a polygon with any number of sides given by the user, which necessitates the abstraction process (see Figure 4). PSTs were then prompted to identify the functions of various coding blocks that they use.

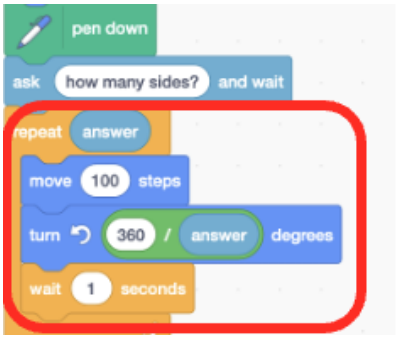


Figure 5. Algorithm to draw a polygon (See Slide 60 in Session 1 Slide Deck: Access Food (PPT)).

Next, the instructor showed another example idea that can be developed in Scratch. This focused on another science concept: energy, and the session ended with a discussion about how coding and programming may be connected to the NGSS (n.d.), which they started to review in the previous week.

Demonstration (1.5 hour)

This part of the activity took place on the second classroom day of the same week (e.g., Thursday of a Tuesday/Thursday class). The Session 1 Slide Deck: Access Food (PPT; slides 65-88) includes a detailed explanation of the steps covered for Day 2. The instructor started by reminding the PSTs of the previous activity of drawing shapes and shows a completed program in Scratch. Then, two example Scratch projects: one simulating the relation between voltage, current, and resistance, and the other one simulating the population of foxes and rabbits changing over time were presented, followed by the instructor discussing the role of simulation in teaching and learning science concepts (see Figure 5).

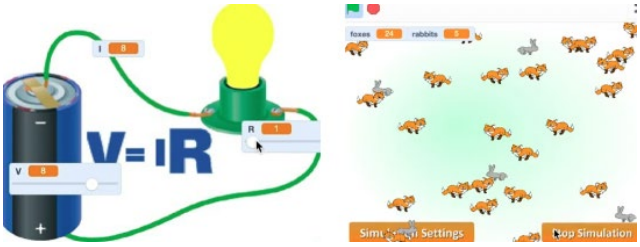


Figure 6. Simulations on Ohm's law and the dynamics of two biological systems (see Slide 69 & 71 in Session 1 Slide Deck: Access Food (PPT)).

PSTs were also prompted to consider any contextual issues that CT may be used for besides the cognitive learning benefits that such simulation examples can offer. To facilitate the discussion, the instructor displayed another simulation that they had previously developed, which shows how modifying factors such as dispersant amount and exposure to sunlight impact how fast an oil spill can be cleaned (see Figure 6).

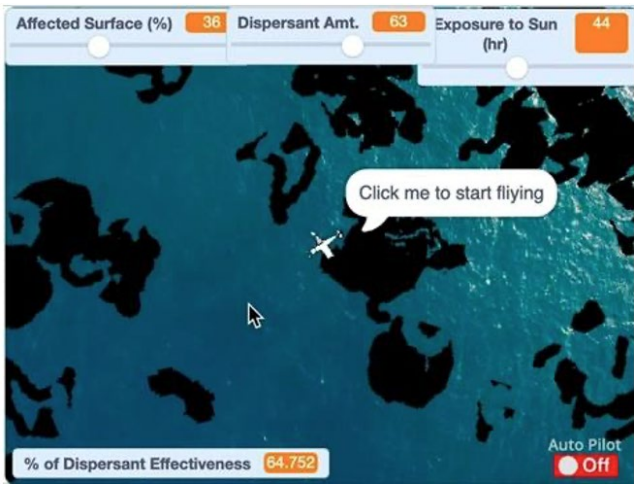


Figure 7. Oil Spill Simulation (See slide 74 in Session 1 Slide Deck: Access Food (PPT)).

Following the simulation, the instructor provided an overview of contextual issues that were more specific to the state focusing on food access. Another Scratch project that the instructor developed previously, which simulates how food access varies depending on a given county, was presented. The PSTs were asked to run the simulation over a certain amount of time, generate data in Excel, and try to identify the equation used in the simulation for the food access index (see Figure 7).

Grocery	Convenience	Food index
1	5	125
4	10	450
6	2	610
3	5	325
8	6	830
19	18	1990
10	3	1015
13	10	1350
19	12	1960

Figure 8. Simulation on food access, and generated data entered in Excel Slides (See Slide 79 in Slide Deck #1).

After 20-30 minutes, the PSTs shared, discussed, and compared their equations and were asked to consider other factors (e.g., family income) impacting food access. In the end, they were tasked with focusing on any of such factor and coding them in

the original simulation before the next classroom. They were asked to use commands engaging in the first three basic CT concepts (e.g., sequences, loops, and events) adapted from Brennan and Resnick's (2012) work on the assessment of CT.

Practice & Application (3 hour)

After the introduction and demonstration in the previous week, the next class focused on practicing and modifying the original Scratch simulation assigned. The Session 1 Slide Deck: Access Food (PPT; slides 89-94) includes a detailed explanation of the steps covered for this practice day. While the instructor modeled, the PSTs were encouraged to complete the coding tasks on their own first, ask questions, and troubleshoot difficulties encountered by others. They continued modifying each project to reflect the new factors to take into account, and remixed other PSTs' projects shared in an online discussion forum.

In the next class day (see slides 95-97 in Session 1 Slide Deck: Access Food (PPT)), having practiced and completed modifying the simulation individually, the PSTs worked in groups of three to start describing a learning activity that incorporates CT in teaching a science concept of their choice. The groupings were based on the grade level of their field experience placement classrooms. They were given a template (see Group Lesson Design Template) to complete their descriptions which would entail determining the contextual issues to be addressed, developing a timeline of planned instructional events, and identifying the CT practices, processes, and concepts that their activity aims to promote. A list and description of CT components were included in the very template for their reference. The PSTs were to submit their group learning activity description to an online discussion forum in a week.

After the group work, the PSTs were given another week to make one modification to an existing science lesson that they acquired from their placement classrooms or a published resource. They were prompted to use their experiences with the sessions and group learning activity so far to modify such an existing plan with a focus of promoting students' CT (see Modification Guideline). They described the existing lesson plan and the modifications as well as responding to reflection questions (e.g., *what are some CT processes that the students would be engaged in this lesson?*). Their responses informed their full lesson plans that they develop throughout the semester (see Lesson Plan Template).

Session 2: Grow Food

This session is a total of six hours that was spread across four in-class days (1.5 hours each) and occurred during Weeks 6-7 of the semester.

Introduction (1.5 hour)

In Week 6 of the semester, the instructor and a guest speaker, a former science educator who also has experience with Garden-Based Learning (GBL) demonstrated to the PSTs how the CT approach can be used to understand and deal with *Grow Food*, one of the means to address the food access issue. Community gardens and active participation in gardening, for instance, have been considered one of the ways to increase local food production (Furness & Gallaher, 2018). As such, engaging the PSTs in basic gardening techniques and employing inquiry skills would expand the concept of food access covered in the first session.

To prepare for the sixth week and to generate some data to analyze, the instructor and the PSTs launched a simple experiment two weeks prior to this classroom. In Week 4 of the semester, the PSTs were guided to work in four groups to sow cucumber seeds in a hydrated potting mix inside plug trays, each of which has 36 cells. A guiding question for the PSTs as well as for an inquiry that they can use in their future classrooms could be "*We cannot harvest cucumbers in this region until early Summer. Can we germinate the seeds earlier in our classroom? What will be the most ideal condition to germinate them when considering the heat, light, and kinds of seeds*?

To examine the impact of heat (no heat-NH, heat-H) and light (normal light-NL, extra light-EL) on germination rates, each group's tray had a different set up representing one of 4 possible variations (see Figure 9):

- Group 1: The tray had no heat mat and was exposed to normal daylight during the day (NH and NL).
- Group 2: The tray had no heat mat (NH) but was exposed an extra light source (EL): an LED grow light placed over it.
- Group 3: The tray was placed on a heat mat (H) and was exposed to normal daylight (NL).

- Group 4: The tray was placed on a heat mat (H) and was exposed an extra light source (EL): an LED grow light placed over it.

To determine the germination rates, groups observed the trays every day until the sixth week and counted the total number of seeds that germinated each day.

Figure 9. Observed outcomes of the experiment.

On the first classroom day of the sixth week, the guest speaker introduced the tenets of GBL and an example project that is being implemented in a local public elementary school. Prior to the session day, he already shared with the PSTs relevant readings and questions about their perspectives on and experiences with GBL. In the classroom, he highlighted specific student artifacts focusing on the germination process in relation to students' science concepts and mathematical understanding and the science and engineering practices of the NGSS. He also directed the PSTs to further resources such as books and guidance about GBL (see Figure 10). The Session 2 Slide Deck: Grow Food (PPT; slides 2-31) includes a detailed explanation of the steps covered.

Figure 10. Guest speaker presenting GBL resources.

Demonstration (1.5 hour)

After conducting the cucumber seed experiments and learning about GBL, the PSTs were ready to analyze their collected data and develop a simulation. In this second classroom day of the sixth week, the instructor reminded the PSTs of the conditions set for their experiments and prompted them to compare the observed outcomes with the focus of identifying any patterns associated with each group. To help facilitate their thinking, the instructor helped them enter the total # of seeds that germinated each day for their condition in a Google Sheet. The PSTs were also guided to develop scatter charts to compare the observed outcomes of the conditions and discuss the relationship between the day and the number of seeds (See Figure 11).

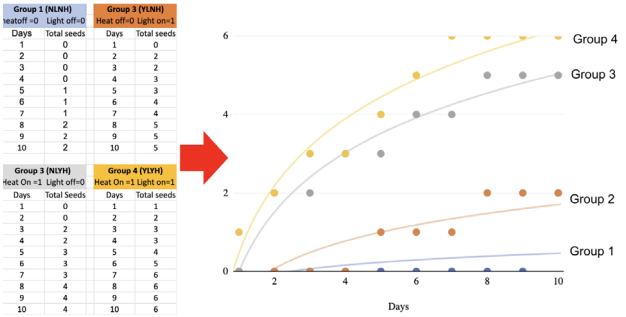


Figure 11. Observed data keyed in and visualized.

Toward the end, the instructor displayed a simplified simulation that allowed users to control multiple conditions and see the effects on the germination process (See Figure 12).

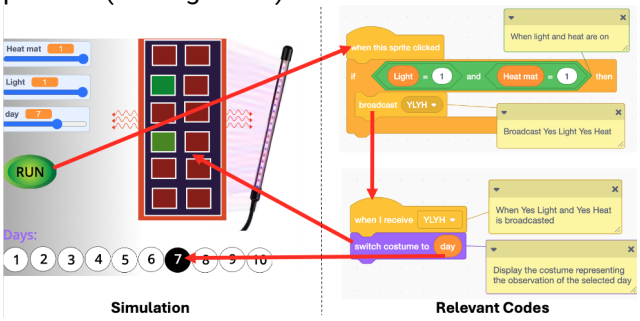


Figure 12. Sample simulation and its relevant codes.

The PSTs were then provided with a link to another version of the same simulation that had partially completed codes. The PSTs were asked to work in groups till the end of the class to finish the coding so that the program would simulate the data observed in their assigned conditions. The instructor supported the PSTs during this time by discussing their codes.

For homework, to be completed before the next class session, the PSTs were tasked with individually developing a new simulation of their choice in Scratch with the observed cucumber seed data. Given their developing experiences with Scratch and the potential complexity of the created simulations, they were provided specific commands to use engaging in both basic and advance CT concepts

(e.g., sequences, loops, events, conditionals, and data) adapted from Brennan and Resnick's (2012) work. The Session 2 Slide Deck: Grow Food (PPT; slides 32-43) includes a detailed explanation of the steps covered in this demonstration class.

Practice & Application (3 hour)

In the following week, the session followed the same format as the Session 1: Access Food practice and application. This practice and application occurred during Week 7 of the semester, encompassing both in-class meeting days. To start, the PSTs brought their individual Scratch simulations to class (the previous week's homework). These simulations were reviewed and modified in both peer-groups and individual instructor review. The instructor again modeled and encouraged them to work on first updating the code of their simulations on their own with the peer and instructor feedback and troubleshooting difficulties encountered by peers.

On the second classroom day of this week, PSTs started working with the Group Lesson Design Template in their previous groups to create a new learning activity incorporating CT in their field experiences the following week. Based on their newly gained understanding from the cucumber seed experiment and Scratch simulations, PSTs used another week to individually modify the existing lesson plan (that they chose last time) toward supporting students' CT. The Session 2 Slide Deck: Grow Food (PPT; slides 44-52) includes a detailed explanation of these steps.

Session 3: Sustain Food

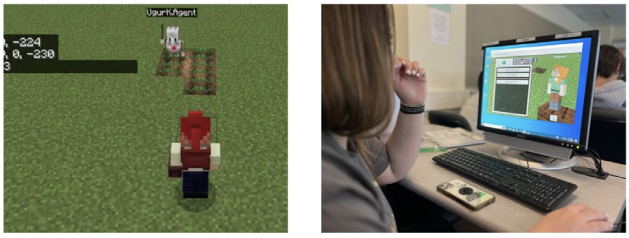
This session is a total of six hours that was spread across four in-class days (1.5 hours each) and occurred during Weeks 10-11 of the semester.

Introduction (1.5 hour)

In the 10th week of the semester, the instructor demonstrated to the PSTs how the CT approach could be used for sustaining food, another key factor for local and healthy food access. Small-scale farming is critical to local and healthy food access (Blackwell, 2016) yet sustainability of food from such sources can be problematic. To make farming more sustainable, agricultural robots have been explored to automate various dull tasks such as picking, harvesting, weeding, spraying, or monitoring for farmers (Dao, 2020; Robotics Online Marketing Team, 2017). While experimenting with such robots may not be feasible due to their high cost, simple and accessible programming platforms like Minecraft (a 3D virtual space where users develop structures via manipulating blocks) have been explored to model a variety of robotics tasks (Aluru et al., 2015). Thus, engaging PSTs in the use of such tools to practice automating relevant tasks would further their understanding of food access and growth covered in the first two sessions. The guiding question for the PSTs, as well as an inquiry they can use in their future classrooms, was: "*A large number of cucumber seeds need be planted in a family-owned farm but it takes a lot of work for the family to prepare the soil and plant the seeds in a short amount of time. Can we automate these tasks such that they are more efficiently completed for a specified planting area layout as well as for any given planting layout*?

To prepare for this week, the PSTs were instructed to download the Minecraft education edition onto their computers and practice navigating some of the existing worlds a week prior to the session. On the first classroom day of the tenth week, the instructor started by presenting information on the agricultural practice and trends in the state and displayed the example uses of robots in agriculture as a means to deal with the sustainability of food in the context. The instructor then introduced and discussed the use of Minecraft (education edition) as an efficient and effective platform to test robotic tasks without using robots in the real world (see Session 3 Slide Deck: Sustain Food (PPT; slides 1-14).

Next, the instructor presented a previously developed Minecraft world (see Figure 13), which simulated the automation of farming tasks (e.g., tilling, planting) via the agent- a robotic-looking character executing the given commands. During this time, he prompted the PSTs to discuss the possible algorithm used, and shared the URL of the world, which the PSTs used to join and explore it for the next 10-15 minutes.



Minecraft World Shared | Minecraft World Explored

Figure 13. Existing Minecraft world shared and explored.

In the remaining part of the class, the PSTs practiced using the Minecraft Education interface such as moving the player around, placing or destroying blocks, opening the code builder window, using commands to code the agent, a character simulating the robot to move it around, make it place and destroy blocks, or teleport it to specific coordinates. The Session 3 Slide Deck: Sustain Food (PPT; slides 2-29) includes a detailed explanation of the steps covered for Day 1.

Demonstration (1.5 hour)

On the second classroom day of week 10, the PSTs were asked to examine the original algorithm used in the previous demonstration, practice modifying it, and observe its impact on the agent's task completion. This time, the instructor shared with the PSTs the URL of the coding file, which the PSTs would use in Minecraft Education to be able to review the codes (see Figure 14).

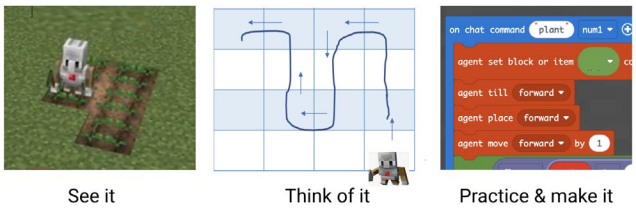


Figure 14. Code file shared, discussed, and modified.

Having explored the codes, the PSTs were asked to work in groups to create new commands that enables the agent to perform the same tasks but in a new planting area with a different size (e.g., 9X9). The instructor monitored the PSTs progress and helped them modify the original codes for the new planting area. At the end of the session, the PSTs were tasked with developing a new world with new commands to simulate the automation of any task prior to the next classroom. They were encouraged to think about their own lessons and accordingly create a world (e.g., biomes) with relevant structures (e.g., buildings, lakes, roller coasters, waterslides) and animals (e.g., cows, wolves, fish). Given their limited experiences with the interface of Minecraft's coding platform (Makecode), they were asked to use commands engaging mostly in the basic CT concepts (e.g., sequences, loops, events). The Session 3 Slide Deck: Sustain Food (PPT; slides 30-47) includes a detailed explanation of the steps covered in Day 2.

Practice & Application (3 hour)

In Week 11 of the semester, the students engaged in the practice and application portion of Session 3, following the same format of the practice and application portions for Session 1: Access Food and Session 2: Grow Food. In the first day of class in Week 11, the PSTs continued working on their Minecraft simulation and modifying the code file while the instructor continued to model and encourage them to troubleshoot the difficulties encountered by their peers.

On the second class day of Week 11, the PSTs got into their groups, using the Group Lesson Design Template, to create a new learning activity incorporating CT for their field experience the following week. Based on their newly gained understanding from the demonstrations in-class and the group work, the PSTs used another week to individually work on modifying the existing lesson plan (that they already chose last time) toward supporting students' CT. The Session 3 Slide Deck: Sustain Food (PPT; slides 48-56) includes a detailed explanation of the steps covered in these practice and application classes.

Assessment Opportunities

The assessment related to the three sessions had two main components. The first component provided the PSTs with an opportunity to reflect on CT right after each session. As described in the practice and application sections, the PSTs worked in groups to design a learning activity incorporating CT into teaching science concepts. The Group Lesson Design Template guided the PSTs to describe the contextual issues to be addressed and the sequence of the instruction to take place as well as identifying various CT components (see Table 1).

Table 1. CT Components Reflected

CT Components	Specific Inclusion
Data Practices	• Collecting data • Creating data • Manipulating data • Analyzing data • Visualizing data

CT Components	Specific Inclusion
Simulation and Modeling Practices	• Using computational models to understand concepts/find/test solutions • Assessing computational models • Designing computational models • Constructing computational models
Processes	• Confrontation • Decomposition • Pattern recognition • Abstraction • Algorithm/Automation • Test/Debug
Concepts	• Sequences • Loops • Events • Parallelism • Conditionals • Operators • Data

As also described earlier, the PSTs worked individually, after their group design, making one modification to an existing science lesson that they acquired from their placement classrooms or a published resource. The Modification Guideline helped their reflection where they described the existing lesson, the modifications they made, and how the modifications will enable their students' CT.

The second component of the assessment was embedded in the PSTs' individual lessons. Having practiced with CT tools in three sessions (Scratch, Excel/Google Sheets, and Minecraft), designed group lessons on CT, and modified existing lessons, the PSTs individually taught full lessons at their placement classrooms incorporating CT. The Lesson Plan Template was shared with them at the beginning of the semester while its elements such as objectives, assessment, and instructional strategies were reviewed throughout the semester to help guide their lesson development. A section toward the end of the Lesson Plan Template focuses on the enactment of teaching where the PSTs respond to questions such as:

- *Based on your implemented lesson, what specific CT practices and processes did you notice students were engaged in? What evidence do you have that indicates students practice computational thinking?*
- *How do you think the modifications better supported your students' conceptual understanding of the lesson focus? What evidence do you have that supports your thinking (i.e. what did you see or hear students say and do that supports why you think the modifications were effective or not towards supporting students' thinking and reasoning)?*

A potential rubric could also further guide PSTs' teaching of CT-infused lessons (see CT Lesson Implementation Rubric).

Critical Reflection

The learning representation was implemented with two cohorts of PSTs in Spring 2023 and Spring 2024. The main goal of infusing CT into the science methods course was to help PSTs gain competence with and increase interest in CT so that they can incorporate it into their science instructional practices.

In each semester, we collected both pre and post surveys as well as conducted interviews focusing on PSTs' competence with, motivation for, and use of CT. As part of the larger research project, we reported our detailed analysis and results in relevant research (Kale et al., 2024, Kale & Wang, 2024). In this section, we provide our reflections on the outcomes for both cohorts, describing major findings along with modifications to the curriculum.

For the first cohort, we observed that the PSTs' competence with CT in teaching science as measured via the survey items at the end of the semester was significantly higher than those at the beginning of the semester. Most PSTs also reported that their understanding of CT improved after completing the course, which had a positive effect on their ability to use it.

However, the PSTs' interest in CT and perception of CT's utility value (e.g., how relevant they consider CT

to teaching) decreased over time while no change was observed regarding the frequency of their CT usage. Congruent with our earlier research (Kale & Akcaoglu, 2017), these findings suggest that the initial interest and perceived value of a technological innovation can diminish if users encounter difficulties with its use. Interview responses emphasized that the use of coding and the complexity of simulations posed challenges for the PSTs, which may have led to a decrease in the initial novelty of CT. This was also reflected in their midterm course evaluations, such as the comment: "*I don't enjoy the coding assignments because they are too complex and confusing for me, let alone my elementary students*". The perceived limited application of CT in placement classrooms appeared to be another factor. One PST noted in their midterm evaluation "*I do not understand why we have a whole class about computational thinking when it is not used in our placement classrooms*."

Based on the findings from the first cohort, we made two main revisions to the curriculum. The first one involved simplifying the coding tasks. As part of Session 1: Grow Food, the PSTs were asked to create a simulation in Scratch to display the number of seeds germinating. The complexity of such coding was amplified especially when the PSTs were introduced to the trendlines and generation of the equation formula in Excel, which would be coded in Scratch to produce the total number of seeds that germinate on any given day. Although using the formula can generate results in the simulation beyond the observation timeline, the PSTs considered it difficult to comprehend. As such, we simplified the coding task by removing the formula generation. Rather, the task can focus on identifying the number of seeds observed in the week only and creating simple conditionals for each day in Scratch (e.g., if Day 1 is selected, then display 2 seeds being germinated), which is much easier for the PSTs to practice. An alternative approach may be to provide pre-training opportunities on the use of Excel for trendline and equation. However, the existing course timeline and other assignments did not allow for such an additional activity.

The second main revision was the use of open-ended personalized coding tasks. Toward the end of the demonstration sections of each session, the PSTs were tasked with completing close-ended tasks such as working on an existing simulation (e.g., coding additional factors in the Food Access simulation). While these tasks allowed practicing many CT concepts, the PSTs, without planning and choosing their own topic, may not have developed the ownership on their coding projects, which may be essential to their motivation (Gonzalez-Maldonado et al., 2022). As such, in addition to closed-ended tasks, we recommend incorporating open-ended, personalized ones, such as selecting, planning, designing, and developing their own projects similar to the one we used in Session 3: Sustain Food where the PSTs are asked to design and develop their own Minecraft worlds.

For the second cohort, we found a similar outcome regarding the competence. The PSTs felt significantly more competent with CT at the end of the semester than at the beginning of the semester. Again, this observation reflects the effectiveness of the activities on PSTs' developing competence. However, unlike the first cohort, the second cohort of PSTs (Spring 2024) were able to maintain their interest and perception of CT's value while a slight increase in their usage was also observed toward the end of the semester. Given the lowered motivations observed with the first cohort, these second cohort observations are promising, which speak to the positive impact of the changes made to the curriculum.

In addition, two potential changes can further help the PSTs recognize the relevance of CT (utility value). The first one requires stronger connections between CT and NGSS (n.d.). This could simply involve activities that more frequently explore the connections between the dimensions of NGSS, such as Science and Engineering Practices and Crosscutting Concepts, and both CT processes and practices (See slide 84 in the Session 1 Slide Deck: Access Food (PPT)). The second one involves exemplifying CT in grade-specific classrooms. Observable pedagogies via existing classroom videos (e.g., Brush et al., 2009; Luna & Sherin, 2017) can be a good solution here though finding or developing such cases in timely manners may not be feasible. PSTs videoing their teaching practices via their mobile phones and securely sharing it with their peers may provide an alternative. While not all of these cases would represent exemplary practices, the challenges faced, the strategies used to overcome them, and the successes seen from peers' perspectives could serve as valuable examples to demonstrate CT's application, thereby enhancing its perceived relevance to teaching.

REFERENCES

Aluru, K.C., Tellex, S., Oberlin, J.G., & MacGlashan, J. (2015). Minecraft as an experimental world for AI in robotics. *AAAI Fall Symposia*. https://aaai.org/papers/11725-11725-minecraft-as-an-experimental-world-for-ai-in-robotics/

Adler, R. F., Hibdon, J., Kim, H., Mayle, S., Pines, B., & Srinivas, S. (2023). Assessing computational thinking across a STEM curriculum for pre-service teachers. *Education and Information Technologies, 28*, 8051-8073. https://doi.org/10.1007/s10639-022-11508-4

Aghasaleh, R., Enderle, P., & Puvirajah, A. (2019). From computational thinking to political resistance. *Journal for Activist Science and Technology Education, 10*(1), 29-44. https://doi.org/10.33137/jaste.v10i1.32915

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20-23.

Blackwell, A. (2016, May 12). *Best practices for creating a sustainable and equitable food system in the United States.* Center for American Progress. https://www.americanprogress.org/issues/poverty/reports/2016/05/12/137306/best-practices-for-creating-a-sustainable-and-equitable-food-system-in-the-united-states

Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking.* In Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada. https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Dao, E. (2020, May 18). This professor wanted to make farming more sustainable, so he started a company to build a fleet of smart robot. *TerraMapp*. https://terra-mepp.illinois.edu/news/professor-wanted-make-farming-more-sustainable-so-he-started-company-build-fleet-smart-robots

Dohn, N. B., Kafai, Y., Mørch, A., & Ragni, M. (2022). Survey: Artificial intelligence, computational thinking and learning. *KI-Kunstliche Intelligenz, 36*(1), 5-16. https://doi.org/10.1007/s13218-021-00751-5

Gonzalez-Maldonado, D., Pugnali, A., Tsan, J., Eatinger, D., Franklin, D., & Weintrop, D. (2022, August). *Investigating the use of planning sheets in young learners' open-ended scratch projects.* In Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1 (pp. 247-263). https://doi.org/10.1145/3501385.3543972

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Gundersen, C., M. Hake, A. Dewey, E. Engelhard (2020). The impact of the coronavirus on food insecurity v1 [Data file and FAQ]. Available from Feeding America: research@feedingamerica.org.

Harel, I., & Papert, S. (1991). *Constructionism*. Westport, CT, US: Ablex Publishing.

Hsu, T. C., & Chen, M. S. (2024). Effects of students using different learning approaches for learning computational thinking and AI applications. *Education and Information Technologies, 30*, 1-23. https://doi.org/10.1007/s10639-024-13116-w

International Society for Technology in Education. (2017). *ISTE standards: Educators*. Retrieved May 14, 2025, from https://www.iste.org/standards/iste-standards-for-teachers

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development, 48*(4), 63-85. https://doi.org/10.1007/bf02300500

Kafai, Y., Proctor, C., & Lui, D. (2019, July). *From theory bias to theory dialogue: Embracing cognitive, situated, and critical framings of computational thinking in k-12 cs education*. In

Proceedings of the 2019 ACM Conference on International Computing Education Research (pp. 101-109). https://doi.org/10.1145/3291279.3339400

Kale U., & Akcaoglu, M. (2017). The role of relevance in future teachers' utility value and interest toward technology. *Educational Technology Research and Development. 66*(2), 283-311. https://doi.org/10.1007/s11423-017-9547-9

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018) Computational what? Relating computational thinking to teaching. *TechTrends*, *62*(6) 574-58. https://doi.org/10.1007/s11528-018-0290-9

Kale, U., Wang Y., Aldebyan, A., Mann T., & O'Brian, N. (2024, April). *Future teachers' interest, competence, and perception of utility value regarding computational thinking in teaching science.* Paper presented at American Educational Research Association Annual Conference.

Kale, U., & Wang Y., (2024, October). *Interested in computational thinking? Role of competence and value in preservice teachers' interest in computational thinking*. Paper presented at the Association for Educational Communications and Technology Annual Conference, Kansas City, MO.

Mayer, R. E., & Wittrock, M. C. (2006). Problem solving. In P. A. Alexander, P. H. Winne, P. A. Alexander, & P. H. Winne (Eds.), *Handbook of educational psychology* (pp. 287-303). Erlbaum.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A new framework for teacher knowledge. *Teachers College Record, 108*, 1017-1054. https://doi.org/10.1111/j.1467-9620.2006.00684.x

Next Generation Science Standards. (n.d.). *Read the standards.* https://www.nextgenscience.org/search-standards

Polya, G. (1957). *How to solve it: A new aspect of mathematical method* (2nd ed.), Princeton, NJ: Princeton University Press.

Robotics Online Marketing Team (2017, December 12). Robotics in agriculture: Types and applications. *Association for Advancing Automation* [Blog]. https://www.automate.org/blogs/robotics-in-agriculture-types-and-applications

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education, 27*(2), 165-205. http://dx.doi.org/10.70725/303733vqleui

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, *162*, Article 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM, 62*(3), 34-36. https://doi.org/10.1145/3265747

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. https://doi.org/10.1007/s10956-015-9581-5

Wilson, B., DeBruin, J., Higgins, A., Gross, T., Gum, H., & Cannon,T. (2017). *Bringing our farms into focus: Specialty food crops, farm viability and market profitability in West Virginia.* Food Justice Lab. Retrieved April 15, 2025, from http://foodjustice.wvu.edu/wp-content/uploads/2018/12/WV-Farm-viability_study_final.pdf

West Virginia University Office of Data and Analytics. (2024, December 1). *Enrollment overview*. https://dataoffice.wvu.edu/reports-analytics/enrollment-overview-board

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

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM. 60*. 55-62. https://doi.org/10.1145/2994591

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms:

measuring teacher understanding of computational ideas for teaching science. *Computer Science Education, 28*(4), 371-400. https://doi.org/10.1080/08993408.2018.1560550

About the Authors

Ugur Kale is an Associate Professor in Indiana University's Learning Design and Adult Education Department. His teaching focuses on instructional design, technology integration, and science education. He also develops and researches innovative approaches that prepare educators to integrate computational thinking.

Yuanhua Wang is an Assistant Professor of Science Education in West Virginia University. Her teaching focuses on elementary science education and mathematics education. She researches the implementation of inquiry-based methods in science education as well as STEM teacher knowledge and computational thinking.

Acknowledgement

Activities described in this publication were produced with the assistance of a grant from the National Science Foundation (Award #2451843; #2142274).

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9353

The Plot Thickens: Literacy Through Computational Thinking

Bataul Alkhateeb[1] and Eiman Abushihab[2]
[1]University of Delaware, [2]Qatar University

Overview

In this lesson, students analyzed word frequency, character interactions, and recurring themes in *Frankenstein* to predict how the novel unfolds before reading the next chapter. Students mapped a decision tree for Victor Frankenstein's choices and predict alternative endings. This lesson utilized Plotting Plots (n.d.-a), created Dr. Tom Liam Lynch as a classroom resource that blends literature with data analysis. This tool leverages computational and quantitative approaches to understanding books by creating visual representations of literary data. The Much Ado About K-12 Computer Science: A Crash Course for ELA and English Teachers YouTube video playlist videos (Plotting Points, n.d.-c) are designed for secondary English Language Arts teachers to strategically map keywords and to support students' exploration and inferences of what they could mean for the plot of the text.

Topics: Computational Thinking, Mixed Literary Analyses, Multimodal Literacy

Time: 45 minutes

Materials

- Plotting Points (n.d.-b) Plots Page
- Computer device with internet access
- 51 book selections from secondary English Language Arts classrooms
- A Guide to Plotting Plots (Lynch, n.d.-a)
- How to Plot a Plot Tutorial (Lynch, n.d.-b)

Context-at-a-Glance

Setting
9th-12th grade English Language Arts classroom at a private faith-based Islamic high school in the U.S.

Modality
Synchronous in-person learning

Class Structure
This lesson is 45 minutes with desks in groups. Students meet three times a week for class sessions.

Organizational Norms
The school emphasizes unique pedagogies, epistemologies, and ways of knowing, being, and doing, but also follow state standards for teaching English Language Arts.

Learner Characteristics
Students were in 9th-12th grade with knowledge on interpretating texts but were not familiar with the Plotting Plots website or using literary data to make textual inferences. All students identified as Muslim.

Instructor Characteristics
The instructor is a learning scientist and English Language Arts teacher. Instructors may need familiarity with collaborative and multimodal learning and pedagogical knowledge of literary texts. A basic understanding of using data to analyze texts is needed, alongside pre-reading skills.

Development Rationale
The purpose, technology, and computational thinking behind Plotting Plots (n.d.-a) has allowed us to imagine an opportunity where Muslim students can analyze texts that reflect their religious and cultural knowledges in creative texts.

Design Framework
Structured experimentation and unstructured play

Setup

To teach this lesson, teachers and students will need a device with access to the Internet. Teachers may find that a projector or large screen can be helpful to display curriculum materials to the class. First, teachers can create a line graph to model to students. Secondly, students can create their own line graphs based on their selected text, character names, and keywords.

Teachers can select a text, ideally one they are teaching, from the Plotting Points (n.d.-b) Plots Page, a searchable database of every word in a selected book and the number of times it was used. Next, teachers can ask students to identify a character's name or keywords that may be of importance to the plot of the text based on the front cover, title, or other indicating points. Then, a line graph with the frequence of the name of keyword used in the text should appear.

Teachers can approach this lesson using distant reading, a method of analyzing literature that focuses on patterns and looking at the larger structure and scope of a text, rather than a single word or concept.

Teachers can set up a digital collaborative space, like Google Slides, for students to post line graphs and share with the larger class. They may also arrange the classroom for small and large group work so students can work together to create line graphs.

After students create their line graphs, teachers can project them on the screen so other students can see each other's line graphs and visualize the selected keywords.

This lesson is adaptable to different classroom setups and can be implemented in an unplugged classroom by manually creating line graphs with paper and pencil, but access to the Plotting Plots website will still be needed for Dr. Lynch's text mining database for literary analysis.

Standards

This lesson supports the International Society for Technology in Education (2016) student standards:

- 1.4 Innovative Designer, 1.4.a – Design Process: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
- 1.5 Computational Thinker, 1.5.a – Problem Definitions: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- 1.5 Computational Thinker, 1.5.b – Data Sets: Students collect data or identify relevant data sets, use digital tools to analyze them and represent data in various ways to facilitate problem-solving and decision-making.
- 1.5 Computational Thinker, 1.5.d – Algorithmic Thinking: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

This lesson also supports the Computer Science Teachers Association (2017) standards:

- 3B-DA-05 – Developing and Using Abstractions: Use data analysis tools and techniques to identify patterns in data representing complex systems.
- 3B-DA-06 – Communicating About Computing: Select data collection tools and techniques to generate data sets that support a claim or communicate information.

Context and Setting

This 45-minute lesson is the first of a reading unit on Mary Shelley's Frankenstein The lesson focuses on students' pre-reading skills to engage in mixed literary analysis. Learning includes combining multiple methods to understand the text, including data-driven inferences and interpretations of patterns and how they impact understanding the plot.

Students create their own line graphs before reading *Frankenstein* and think computationally about this literary work, how numerical data affects their interpretation of the text, and share their interpretations. Their graphical data engages in mixed literary analysis by analyzing word frequencies (quantitative) on the line graph and citing textual evidence (qualitative).

Plotting Points

Plotting Plots (n.d.a) is an opportunity for students to engage in question-asking about a text, where they can see and identify patterns in a text by looking holistically at its words. The tool's creator explains this as the process of "zooming out" of a close reading and considering characterization, structure, genre, writing style, symbolism, and evidence-based interpretations of a text's plot (Lynch, 2019).

This lesson was taught at a faith-based Islamic school in Northern Virginia. While the school has its unique pedagogies, epistemologies, and ways of knowing, being, and doing, they also follow state standards for teaching English Language Arts. The standards identify literacy goals and instructional and formative and summative assessment expectations, which are similar and different by grade level.

For example, the Virginia Department of Education (2024) highlights that 11th grade students are expected to create and deliver multimodal presentations and expressions while considering how digital literacy is positioned for specific audiences.

Muslim Students' Funds of Knowledge

At this school, students bring with them their lived experiences and positionalities as Muslims to their reading experience and interpretation of characters and texts. While this did not directly inform how they plotted line graphs for our in-class reading of *Frankenstein*, it may have impacted their qualitative analysis and how they chose to engage in meaning-making about the text.

As we consider the location of a faith-based school, students' cultural literacies, and religious funds of knowledge, we reimagined Plotting Plots through culturally responsive practices like pedagogies of witnessing (Baxley & Sealey-Ruiz, 2009; Ladson-Billings, 2009). In Islam, we believe God is a Witness of the self (Quran 41:53).

In critical English studies, witnessing is a process of self-restoration, where literature is a place for "worthy and responsive witnessing" and a location for healing (Baxley & Sealey-Ruiz, 2021; hooks, 1994). Reading, discussing, and engaging with texts that bear witness to students' personal and poetic selves fosters community, wellness, and a radical love for oneself and its knowledges (Silvas, 2020).

As Bishop (1990) writes, books and discussions about books should be "mirrors, windows, and sliding glass doors," or current representations of students' cultural repertoires and the possibilities for cultural representations that are becoming. Plotting Plots includes examples of mainstream literature taught in secondary schooling, and while we teach those books in our faith-based school, we also teach other texts situated within our socialized schooling environment.

For example, we taught Adania Shilbi's *Minor Detail*, which uncovers themes of colonialism, violence, memory, and identity. Students engaged in close reading, by analyzing specific passages from the first and second parts of the split novel to witness narrative voice and the mirroring of a felt dehumanization.

Unlike the characters in the book, the students at this faith-based school have not experienced the very specific and nuanced military occupation that the character in this text has. However, like the character, Muslim students often feel in-betweenness as a liminal and hybrid space of existence, belonging, and negotiation of self as they navigate between two or more cultural, linguistic, and/or social contexts (Sarroub, 2002; Turner, 2021).

Muslim students may not have the space in school to become their whole self as the negotiate who they are, even in a faith-based school that reflects their religious and cultural repertoires. However, in-betweenness also affords students the creativity and agency to uncover a new sense of self that is authentic to who they are.

Computational Thinking

The purpose, technology, and computational thinking behind Plotting Plots has allowed us to imagine an opportunity where Muslim students can analyze texts that reflect their religious and cultural knowledges in creative texts, like Shibli and Jaquette (2020), but also in religious texts, like the Quran. For example, the Arabic word for Knowledge or عِلْمَ (in its many forms as a verb and noun) is mentioned 854 times in the Quran, emphasizing to learners that knowledge, in its many forms, is a responsibility.

By quantifying the importance and understanding the intentional choice and added context of where a word

appears, the learner develops critical consciousness about their role in spreading knowledge.

This is also true for other faiths and religious texts, like the Holy Bible. While some students may struggle to relate to prophetic knowledge, trials, tests, and tribulations, they may be able to empathize and connect to characters in multicultural literature and their experiences turning to faith as a spiritual support (Rackley, 2024).

Learning Representation

This lesson is for an introductory lesson on textual analysis in the secondary English classroom. Dr. Lynch positions this lesson over a 15-day unit to be taught as a larger lesson (see Lynch, 2017), but this lesson may be considered an adaptable single class snapshot to engage student interest and set a foundational understanding of the tool and its function. This lesson can be a standalone activity or a component of a broader sequence depending on instructional time and instructor purpose.

This lesson blends students' need for structured experimentation and unstructured play. For the former, students are given specific textual elements to engage with and receive modeled guided analysis techniques. For the latter, students experience space for experimentation, interpretation, and question-asking through creative re-interpretation of the text and its keywords and interpretive flexibility through open-ended possibilities of what a graph may look like in this activity.

Students' prior knowledge about textual analysis and their selected text may vary. They do not necessarily need to know how to use the website prior to this lesson and one instrumental component of the lesson is navigating different tools prior to selecting a book and its keywords. Students should have some familiarity with making textual inferences about a book, including major themes, characters, and stylistic elements, which will become essential data points in the plot.

Do It For the Plot (45 minutes)

During this lesson, italic text identifies questions or prompts for the learners.

Warm-Up (10 minutes)

To start this lesson, the instructor engaged students in a discussion about their knowledge of qualitative and quantitative data and how they may define the two methods.

Students were asked to share examples of each method and explain that quantitative focuses on quantity (things that can be counted) and qualitative considers quality (stories or feelings behind the data). Then, the instructor shared an example plot (see Figure 1) by providing enough context so the students could make an inference. At this point, the title of the text, *Romeo and* Juliet, had not yet been provided to the students.

> *This is an example of a play by William Shakespeare. If we look at the x axis (act/scene), the y axis (frequency of words), and the graph, what might we learn?*

The teacher asked guiding quantitative and qualitative questions to the students:

- *What does it mean for the plot if we see the word, 'love' being used 25 times in scene 2.2?*
- *What does it mean that the word 'death' was used 18 times in scene 5.2?*

From just those two words, students made the inference that this is *Romeo and Juliet* and they made the inference that scene 2.2 is the ball where the couple meets and scene 5.2 is a tragedy (no spoilers!).

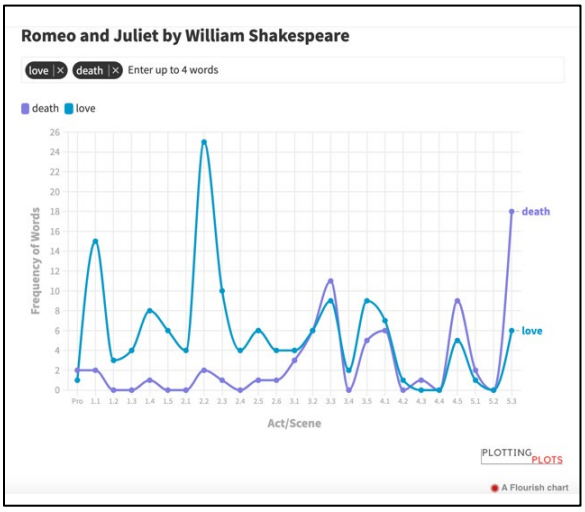


Figure 1. Plot of Romeo and Juliet.

Lesson & Demonstration (8 minutes)

After the opening discussion, the instructor facilitated discussions about students' pre-reading and prior knowledge. Students were asked to consider what they know about the text:

- *What do you already know about the text?*
- *What do you know about the author?*
- *How might this text connect to what we have read before in this class?*

Students were asked to select *Frankenstein* from the Plotting Plots website. Scaffolding questions about the text and the database were provided:

- *Are there any specific terms of ideas you're familiar with?*
- *How about similar topics or themes?*

Then, students were asked to identify three keywords or character names from the text (see Figure 2). If they were unsure of which words to select, based on the title, book cover, or other contextual clues, they were encouraged to play around with the database and see what words were available to them.

Plotting Plots allows for an exploratory process of learning by providing all the possible words (in alphabetical order) that were written in the book, including articles. Students can choose from an array of words, even if they are unsure about what they mean in the context of the text just yet.

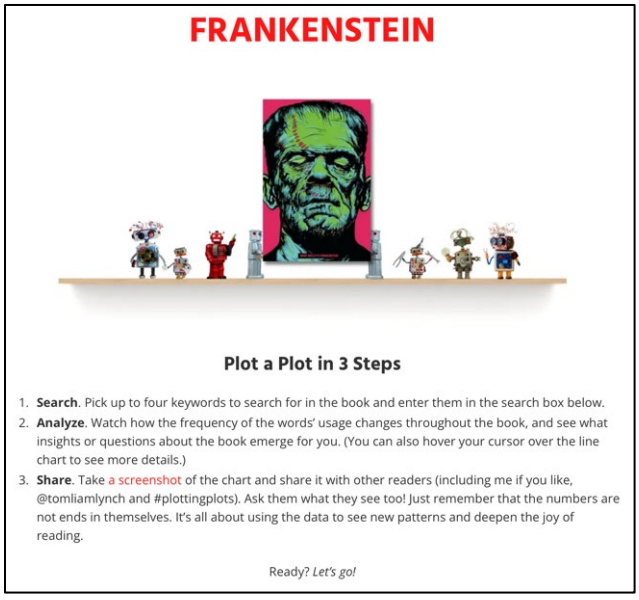


Figure 2. Steps to creating a plot.

Guided Practice (15 minutes)

The instructor modeled to students how to use Plotting Plots. The instructor selected four keywords and explained to students how and why, based on their pre-reading and prior knowledge skills, the word "blessed" appears twice in chapter 8 (see Figure 3).

Throughout the text, Frankenstein mentions how knowledge (a theme) has blessed him, but it ultimately results in his downfall, which is why we see it again in the last chapter. Again, no spoilers!

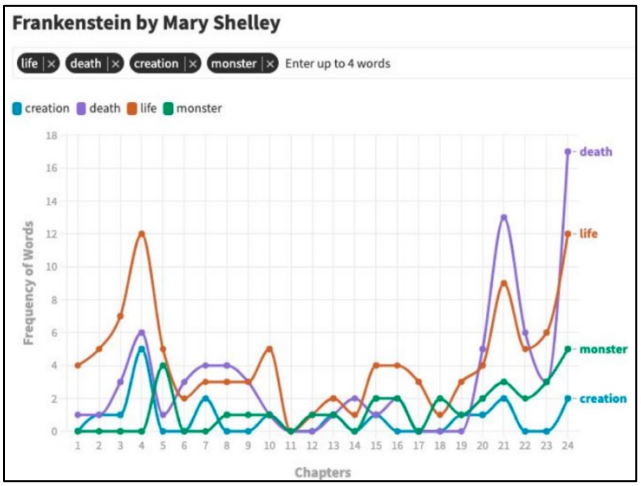


Figure 3. Teacher example of a plot.

Share-out (6 minutes)

Students explored different possibilities and were encouraged to connect their inferences back to the text (see student artifacts in Figures 4 and 5).

Students were asked to explain why they chose the keywords they chose and what they think this reflects about the narrative, the tone of the piece, and the author's intent:

> *What does the word 'betrayal' suggest about the relationships between characters?*

After students shared their keywords, they were asked to compare with their peers. This sharing activity opened a dialogue about different interpretations and allowed them to see how others viewed the same text through a different lens. This analysis can be a discussion tool to identify patterns or surprising outliers that might spark further analysis.

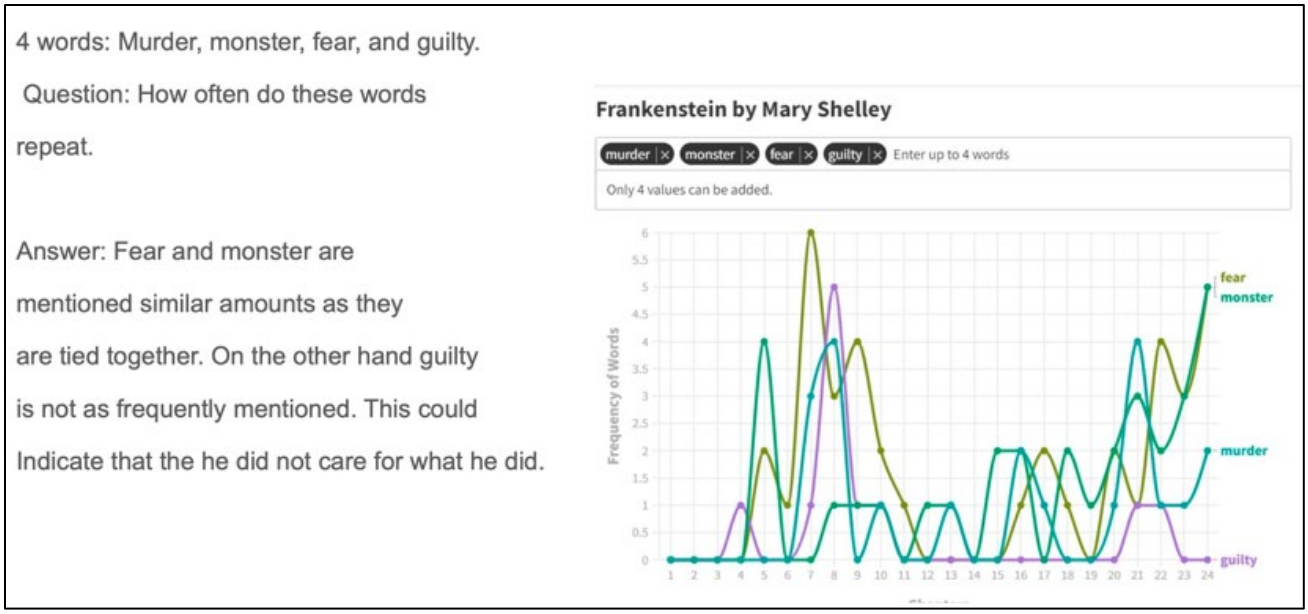


Figure 4. Example of student artifact with the keywords, "murder," "monster," "fear," and "guilty." Based on the data, the student drew an inference regarding Frankenstein's emotional state.

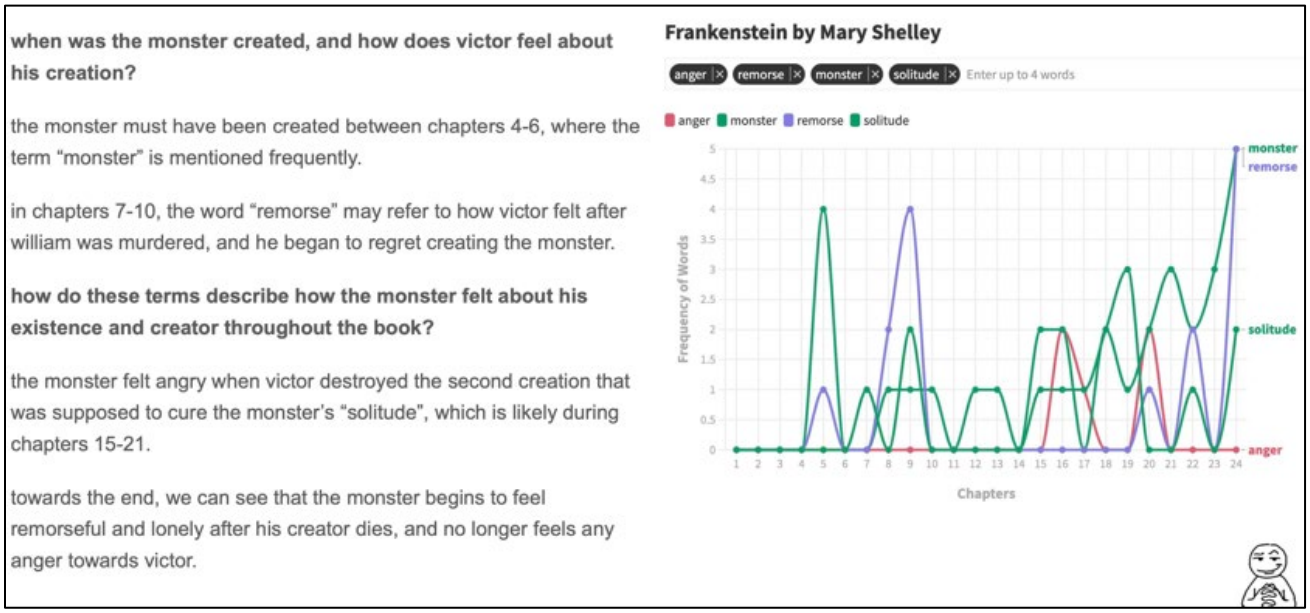


Figure 5. Example of student artifact with the keywords, "anger," "solitude," "remorse," and "monster." Based on the data and textual evidence, this student hypothesized a different interpretation to Frankenstein's emotional state.

After selecting their keywords, students were asked to find specific quotes or scenes that align with each word. This part of the activity reinforced the connection between their inferences and the actual text.

Reflections & Exit Ticket (6 minutes)

At the end of the lesson, the instructor encouraged guided reflection by asking students to select one of two prompts designed to check for understanding:

- *How did your keyword choice help you better understand the narrative? Did hearing others' keywords change or deepen your perspective?*
- *If you had to convince someone else that your keyword is central to the text, what argument and evidence would you use?*

Critical Reflection

The purpose of this tool is to demonstrate how computational methods can expand meaning and deepen the humanistic work of reading literature (Lynch, 2019). With line graphs, teachers can engage students in discussions about how and why certain characters interrelate and when to use quantitative data to identify patterns and then turn to the text to understand its qualitative meanings.

This lesson offers the opportunity to blend structured literary analysis with open-ended exploration and caught the interest of students intending to major in STEM disciplines. During our reading of *Frankenstein,* students began to generate their own questions about the text based on the data visualization and relied little on my guiding questions. While the website does not offer every book listed that a secondary English teacher may teach during an academic year, one can still leverage the tool's resources for teaching data analysis, making meaningful connections between the text and data for interpretation between form and theme, while also modeling and scaffolding pre-reading skills.

This lesson is situated as an introduction to the teaching of *Frankenstein*, but in future iterations, I would like to include more opportunities for students to consider genre and how structure impacts the way a narrative is told, which then impacts the distribution of keywords and overall plot. There is no right answer in this lesson, allowing students to engage in an authentic inquiry process that emphasizes interpretation, discovery, and critical thinking, all of which can be supported with textual evidence and thematic investigations.

Thinking Ahead

Digital tools help us to ask new questions about a text and to engage deeply with its words and where they may be distributed within a text. This can provide students with the opportunity to be deeply interpretive as a pattern of meaning, not just a technical feature within a larger digital tool. It is important, then, to model to students how to pair a data point with close-reading and making "if-then" statements about potential plot points in a text.

By focusing on one book in a lesson, like we did, we can encourage students to identify multiple interpretations from the same data, where different students may draw different conclusions from the same visualization with text-based evidence. This allows for negotiated meaning with data and students can share competing analyses of the same graph and wonder how each got to different points while given the same prompt.

Also, if focusing on one book, students can experience forward-thinking and comparative analysis by comparing different word frequencies and patterns by looking across different works by the same author. This encourages them to evaluate authorial choices. For example, after reading *Frankenstein*, students might apply similar analytical approaches to Mary Shelley's *The Last Man!*

With any tool like Plotting Plots, the aim is to empower students to see that data can support interpretation, not replace it. The numbers prompt richer qualitative questions and the interpretations give the numbers meaning. This approach not only demystifies data but also deepens students' analytical thinking and meaning-making.

For Muslim students at this faith-based school, there are opportunities to analyze texts that are religiously situated and may inform students' understanding of religious concepts. Like creative texts, there is significance in word recognition as it relates to religious concepts and texts.

References

Baxley, G. & Sealey-Ruiz, Y. (2021). In the Black radical tradition: Poetry as a praxis for healing and resistance in education. *Research in the Teaching of English, 55*(3), 311-321.

Bishop, R. S. (1990). Mirrors, windows, and sliding glass doors. *Perspectives, 6* (3), ix-x.

Computer Science Teachers Association (2017). CSTA K-12 computer science standards, revised 2017. https://csteachers.org/k12standards/

Hooks, B. (1994). *Teaching to transgress: Education as the practice of freedom.* Routledge.

International Society for Technology in Education (2016). ISTE standards: Students. https://iste.org/standards/students

Ladson-Billings, G. (2009). *The dreamkeepers: Successful teachers of African American children* (2nd ed.). Jossey-Bass.

Lynch, T. L. (n.d.-a). A reader's guide for book lovers who like data. *Plotting Points.* https://plottingplots.com/wp-content/uploads/2021/09/plotting-plots-classroom-DRAFT-2.pdf

Lynch, T. L. (n.d.-b). How to plot a plot tutorial. *Plotting Points.* https://plottingplots.com/how-to-plot-a-plot/

Lynch, T. L. (2017, May 21). ELA (6-8). Plotting plots [Google doc]. https://docs.google.com/document/d/1zbYvpdW8R_dssw5hOZe0RwBxG2I6zhzsvbplx4WW6Sc/edit?tab=t.0

Lynch, T.L. (2019). Electrical evocations: Computer science, the teaching of literature, and the future of English education, *English Education, 52*(1), 15-37. https://www.jstor.org/stable/26826196

Plotting Plots. (n.d.) *Frankenstein.* Retrieved May 31, 2025 from https://plottingplots.com/frankenstein/

Plotting Points. (n.d.-a). *Home*. Retrieved May 31, 2025 from https://plottingplots.com/

Plotting Points. (n.d.-b). *Plots*. Retrieved May 31, 2025 from https://plottingplots.com/plots/

Plotting Points. (n.d.-c) *Much ado about K-12 Computer Science: A crash course for ELA and English teachers* [Video playlist]. YouTube. https://www.youtube.com/playlist?list=PL_YqVfTEvsGJ8j-5pJiKVPN7bGjMfDvnL

Rackley, E. (2024). 'Not reading just seems crazy to me': Religious youths' textual ideologies of sacred texts, *Language and Literacy*, *26*(1), 104-122. https://doi.org/10.20360/langandlit29676

Sarroub, L.K. (2002). In-betweenness: Religion and conflicting visions of literacy. *Reading Research Quarterly, 37*(2), 130-148. https://www.jstor.org/stable/748154

Shibli, A., & Jaquette, E. (2020). *Minor detail.* New Directions.

Silvas, T. (2020). Writing for liberation. *Language Arts, 98*(2), 80-82. https://www.jstor.org/stable/27140994

Turner, J. D. (2021). Mapping the intersections of religion, literacy, and public schooling for displaced, immigrant, and refugee children: A conversation with Loukia K. Sarroub. *Language Arts, 98*(3), 149-155. https://www.jstor.org/stable/27058935

Virginia Department of Education. (2024, March 28). *2024 standards of learning for English*. https://www.doe.virginia.gov/teaching-learning-assessment/k-12-standards-instruction/english-reading-literacy/standards-of-learning

About the Authors

Bataul Alkhateeb is a Ph.D. in Education student specializing in Learning Sciences in the School of Education at the University of Delaware.

Eiman Abushihab is a Lecturer in Arabic for non-native speakers at Qatar University.

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9113

Transformations Throw Down: Extending Mathematics Knowledge with Assemblr

Brian T. Johnson, Rebecca Bramwell, Josef Stamps, Katie Patterson, and Kyle Jones, Lakeside Junior High School, Springdale Arkansas

Overview

The purpose of these 8th grade math lessons was to extend students' knowledge of sequences of mathematical transformations by providing students with a digital experience of transformations in a three- dimensional environment. Assemblr, an augmented reality app was used to create these experiences for students after they learned these concepts in 2D. Past studies noted that augmented reality activities and gamification promote active learning and increase academic performance (Lampropoulus, et al., 2022; Sukriadi et al,. 2023; Kurniawan, et al., 2024). Students used Assemblr to extend their knowledge in a 3D environment. Their learning was expressed in a game where correct answers to Assemblr challenge questions related to transformations initiated a turn for a team to connect dots and create a square.

Topics: Transformations, Rotations, Symmetry, Dilations, Augmented Reality

Time: 2, 40-minute class periods

Materials

- Assemblr Website
- Teacher Presentation Slides
- Student Presentation Slides
- Student Reflection Survey
- Transformations Videos (Johnson, 2024a, 2024c, 2024d)
- Whiteboards for student answers (1 per team)
- Whiteboard markers (1 per team in various colors)
- A large white board for the Dots game
- Chromebooks, tablets, or smartphones
- A computer connected to a projector to display answers

Context-at-a-Glance:

Setting
8th grade students in an 8-9th grade, public, suburban junior high school in the United States

Modality
Face to face instruction with student groups of 2-4

Class Structure
Two 40-minute 8th grade math extension classes were held for groups of 15-20 students during intervention class periods

Organizational Norms
Three 8th grade math teachers used assessment data to assign students to two sequential extension days in the library for this lesson.

Learner Characteristics
Students who were evaluated at or above proficiency for the unit on transformations. Students were new to using Assemblr.

Instructor Characteristics
One mathematics teacher and one librarian co-taught

Development Rationale

Students extended their learning with transformations while engaging with augmented reality. Teams observed how a polygon moved on a coordinate grid and identified the type of transformation that occurred. They earned units that could be used in playing "the dot game" (The Game Gal, 2018). Adding this additional strategic component helped to engage the students in the competition.

Design Framework
5 E's Framework (Bybee & Landes, 1990)

Setup

Students completed the activity in the library. There was a table with a whiteboard, marker, and eraser for each team as well as a table for the game board. All of these were spaced out enough that teams could have private conversations about their answers (see Figure 1). The library also had displays around the room so that each team could easily view the challenges displayed on the screen as well as the timer.

Figure 1. Students collaborating to determine a solution for a scene in Assemblr

Standards

The following Arkansas Math 8 standards aligned with this lesson (Arkansas Department of Education, 2016):

8.GM.7: Identify a single transformation used to transform one figure onto another on a coordinate plane

8.GM.8: Given two congruent figures, describe a sequence of transformations that maps one figure to another.

8.GM.11: Given two similar two-dimensional figures, describe a sequence of transformations that exhibits similarity, including rotations, reflections, translations, and dilations

The following International Society for Technology in Education standards for educators (2017) were used:

2.5.b Design authentic learning activities that align with content area standards and use digital tools and resources to maximize active, deep learning.

2.6.b Manage the use of technology and student learning strategies in digital platforms, virtual environments, hands-on makerspaces or in the field.

2.6.c Create learning opportunities that challenge students to use a design process and computational thinking to innovate and solve problems.

This lesson also aligned with four of eight Mathematical Practice Standards from the Common Core State Standards for Mathematics (National Governors Association Center for Best Practices & Council of Chief State School Officers, 2010):

CCSS.MATH.PRACTICE.MP1 Make sense of problems and persevere in solving them.

CCSS.MATH.PRACTICE.MP2 Reason abstractly and quantitatively.

CCSS.MATH.PRACTICE.MP6 Attend to precision.

CCSS.MATH.PRACTICE.MP7 Look for and make use of structure.

Context and Setting

This instructional unit was implemented at an 8-9th grade junior high school in a suburban region of the mid-south United States. In this school setting, students have 1:1 access to Chromebooks. Teachers in the school frequently collaborate with the librarian to plan instructional extensions. Students participate in content area instruction and extension activities in the library.

This instructional unit emphasized describing sequences of transformations, aligned with Arkansas Math 8 Standards (8.GM.7, 8.GM.8, 8.GM.11). It pushed students to look for patterns to recognize the type of transformation performed, as well as opportunities to describe sequences performed in 3D.

The lessons were developed for extension sessions held during enrichment class periods as a part of an 18-day instructional unit on rigid and non-rigid transformations. The lessons were designed to help students who had already demonstrated proficiency in working with transformations to continue to extend their learning (Arkansas Math 8 Standards 8.GM.7, 8GM.8, 8.GM.11).

The lessons also added elements of autonomy for learners to think critically about sequences of transformations. Students had learned about transformations in a two- dimensional environment on paper during classroom instruction. Extension lessons pushed students to think critically about transformations in the three-dimensional space of the Assemblr app. In practice, students' critical thinking included their small group discussions during the game and their correct responses written on the white boards for each challenge.

Learning Representation

Engage

The goal of these lessons was to extend students' knowledge of sequences of mathematical transformations by providing students with a digital experience of transformations in a three-dimensional environment. As students entered the library, they were sorted into teams of three and directed to a table. Each table had a whiteboard, dry-erase marker, and three chairs. Each team was directed to open one Chromebook and navigate to a shared Google Slides presentation as part of the "Transformation Throwdown!" Each team was challenged to read the description of a transformation on the slide and select all "scenes" that depicted the transformation(s). To ensure each team was ready to access the challenge, they were directed to the next slide which linked to the augmented reality site, Assemblr (free version). While familiar with the idea of transformations, this was the first time students used an augmented reality site or worked with polygons they could see moving in real-time on the coordinate grid. In teams, students were given about five minutes to play with the program and familiarize themselves with the actions that might happen as they clicked, zoomed, and rotated the figures on the screen, or changed the scene (see Figure 2).

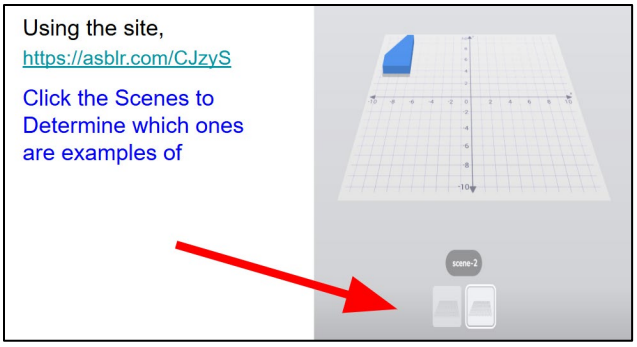


Figure 2. Slide highlighting Assemblr scenes.

Explore & Explain

After students familiarized themselves with the program, they were directed to an Assemblr scene where they observed a transformation and answered the displayed questions like, "On scene 1, describe the reflection." An online timer posted on the screen informed students how long they could respond to each question. Teams had to agree on their answers and record them on a whiteboard at their table.

When the timer concluded, teams with correct answers sent a representative to the game board to play "dots" (see Figure 3; The Game Gal, 2018). On a small grid consisting only of dots, the team representative created a line segment by connecting two dots either vertically or horizontally. Teams created squares by connecting line segments around four dots. The team with the most squares at the end of class won the game. Placing segments was strategic because teams could use any existing segment on the board to help them create a square. This game was new to some students, but they picked up the rules quickly.

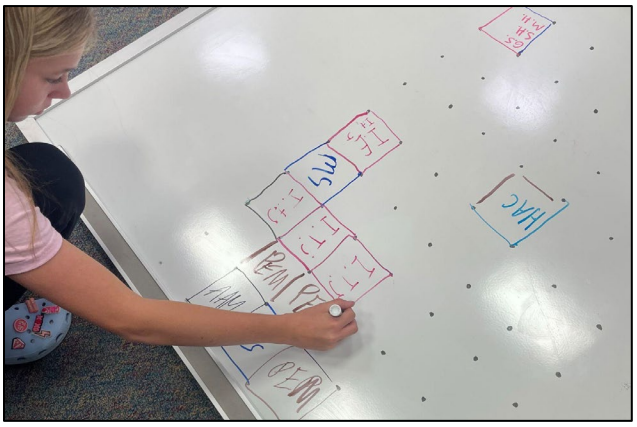


Figure 3. Students add lines to the connect the dots board.

We spent the first day continuing this process for a few rounds, letting team members work together to answer questions on specific screens, reviewing key terms related to transformations, and getting familiar with the dots game. This process set up teams for the competition on day two!

ELABORATE

On day two of the transformation throwdown, students went back to their teams, opened their Google Slides, and opened a tab for Assemblr.

The competition began and the questions got a little more difficult. Instead of being directed to a specific scene and asked about the details of a given transformation, teams were given a series of transformations and asked to list all scenes that demonstrated the transformation (see Figure 4). Within the round, teams were directed to specifically find all scenes that matched criteria 1, "Translated into a new quadrant". They were given either one or two minutes to complete the task and write the numbers of each scene meeting the criteria on their team whiteboard. When the timer expired, teachers checked for correct answers. Teams with correct answers placed a segment on the dot game board. For example, Round 1 had six opportunities for teams to demonstrate their understanding and make a segment on the dot game board.

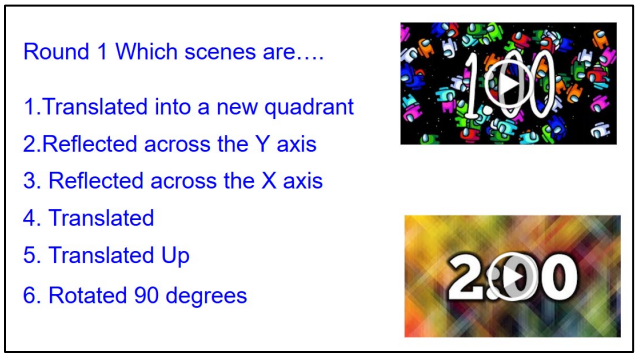


Figure 4. Instructions for Round 1 where students compiled a list of all scenes showing the identified transformations.

In the next round, teams were asked to complete the same task, but each scene showed a sequence of transformations rather than a single transformation. Teams used the features in Assemblr to drag and rotate the coordinate plane to view the transformation from different angles. They were able to get very specific about how they described the transformation. The conversations between the teammates were rich with academic math language as they debated which scenes included the specific criteria.

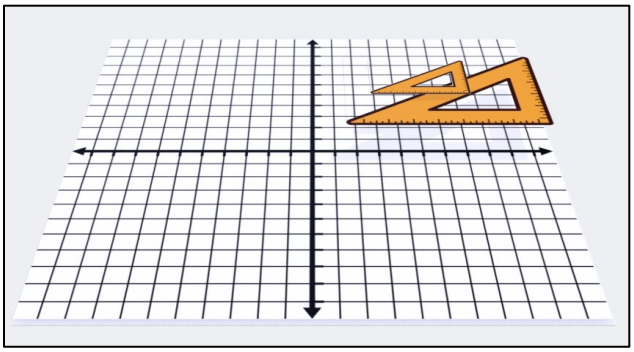


Figure 5. Initial view of polygons in Assemblr

For example, Figures 5 and 6 show the initial view of the polygons and the transformed view of the polygon scene, respectively. Paper assignments often show something like Figure 6, but from a more straight-on angle. Students are told which image is the pre-image and lean into procedural knowledge to arrive at their solution. In this activity with Assemblr, teams moved the plane around. Because the polygons almost seem to hover above the coordinate plane, there is somewhat less precision about the exact location of the vertices of the polygon. Lacking this precision led to conversations that pushed students to demonstrate a conceptual understanding of the whole process of the transformation. Students debated which image was the pre-image and why, explained why a translation might be a certain direction or distance, and proved a specific dilation scale factor to their other teammates by zooming in or out and pointing to the points they were using.

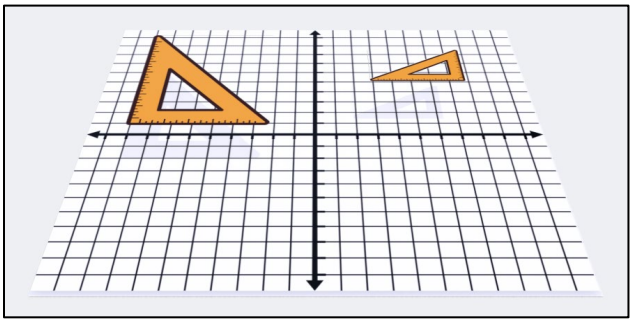


Figure 6. View of polygons in Assemblr after initiating transformation sequence.

In the following round, teams were asked to look for lines of symmetry as part of their criteria with three

dimensional shapes (see Figure 7). Although this topic was briefly touched in class, this was a newer concept for these students, especially on a site where they were able to consider more lines of symmetry than the two-dimensional activities seen beforehand.

Figure 7. Identifying Lines of Symmetry

In the last round, teams were asked to compare three-dimensional figures existing on the same scene and on the same coordinate plane to determine which figures were congruent. This helped tie academic vocabulary and concepts to the rigid transformations they had seen in round one.

Evaluate

Teachers were able to evaluate students' understanding by questioning their answers, listening to rich discussions of each team during challenges, and asking representatives who came to the dot board to elaborate on how they arrived at their conclusions.

In addition to evaluating their mathematical knowledge, after each intervention/enrichment period, students were encouraged to complete a brief survey about their experience via a Google Form. Students were asked to rate the activity on a scale of 1-5 and answer four reflection questions about the activity.

One third of students who completed the survey said that the Transformations Throw Down helped them to better visualize transformations.

Critical Reflection

While the teachers heard academically rich conversations throughout the activity, in the future they will create a reference sheet of "listen-for" statements or vocabulary to help assess and give feedback to groups in the activity.

In addition to evaluating their mathematical knowledge following the activity, after each intervention/enrichment period, students were encouraged to complete a brief survey about their experience via a Google Form. Students were asked to rate the activity on a scale of 1-5 and answer four reflection questions about the activity.

One third of students who completed the survey said that the Transformations Throw Down helped them better visualize transformations. Many students who rated the activity favorably noted that their favorite part of the Transformation Throw Down was the competitive nature of the activity.

This led the teachers to discuss how other more commonly known games might be utilized in the same or other similar mathematical lessons. In future iterations of this activity, we might use a game that is more familiar for the competition component, such as Connect 4.

Other ideas for improving the session included an additional class period so that there is enough time to have teams complete all challenge questions. While some groups were able to complete all rounds, it was rushed for just two 40-minute class periods and not all groups were able to accomplish every round. If an additional class period were added, it might allow students to create their own sequences and challenge questions using Assemblr to present to other teams. For either option, it would be important to have the initial experience so that students understand how the platform worked.

Another critical reflection was the realization that this activity might benefit all learners, not just those who had demonstrated proficiency in the standard. The visual of watching a reflection physically flip over an axis could help other students beyond the use of traditional physical manipulatives. It would be

interesting to see how the teachers could adapt this lesson to a full 85-minute class period for all students rather than just the enrichment periods. Other keys to successful implementation of the lesson might include:

1. Making sure the game board isn't too big for the dot game. The strategic thinking that comes with this game will be more apparent if the board is small enough that the teams end up playing off each other's segments to build squares.

2. Strategically creating the teams. Some kids are more familiar with technology. Some are super competitive while others feel stressed by competition. By creating heterogeneous groupings, each team was able to use their strengths and learn from each other rather than feeling stuck or stressed.

3. If students are struggling with analysis paralysis, adding a speed bonus. For some classes, the starting rounds left students feeling stuck or too worried to write an answer on the board. When we added a speed component - a bonus line segment for the dot game for whichever team had the correct answer on their board first - students were willing to give it a try and write an answer down, even if they weren't 100% confident.

4. Rotating team roles. The teams typically consisted of 3-4 students. If there was not a specific role given and rotated, there's a good chance that one student is "in charge" of the computer or one student takes over all the dot game work. Student roles could include: image mover to rotate scenes in the Assemblr app, scribe (for the whiteboard), and marker of the lines on the gameboard. By rotating roles, students were motivated to work together and share their knowledge, especially when it came to the new concepts of the augmented reality platform or the dot game strategies.

References

Arkansas Department of Education. *(2016). Arkansas Mathematics Standards Grades 6-8.* https://dese.ade.arkansas.gov/Files/20201211113747_Arkansas_Mathematics_Standards_6_8.pdf

Assemblr Studio. (2025). [Computer software]. https://studio.assemblrworld.com/projects

Bybee, R., & Landes, N. M. (1990). Science for life and living: An elementary school science program from Biological Sciences Curriculum Study. *The American Biology Teacher, 52*(2), 92-98. https://doi.org/10.2307/4449042

International Society for Technology in Education. (2017). *ISTE standards: For educators.* Retrieved May 19, 2025 from https://iste.org/standards/educators

Johnson, B.T. (2024a, October 16) Transformations throw down congruent examples [Video]. YouTube. https://youtube.com/shorts/bcdyZc7nOaM?feature=share

Johnson, B.T. (2024b, October 16). Transformations throw down lines of symmetry explanation [Video]. YouTube. https://youtu.be/XRrMgulytCg

Johnson, B.T. (2024c, October 16). Transformations throw down reflection explanation [Video]. YouTube. https://youtube.com/shorts/ZKoQTn-BapY

Kurniawan, P. Y., Nisa, E. K., Sari, F. K., & Ramdhan, N. A. (2024). Revolutionizing language learning: Exploring the efficacy of augmented reality technology through Assemblr Studio. E3S Web of Conferences, 500, 1–9. https://doi.org/10.1051/e3sconf/202450001020

Lampropoulos, G., Keramopoulos, E., Diamantaras, K., & Evangelidis, G. (2022). Augmented reality and gamification in education: A systematic literature review of research, applications, and empirical studies. *Applied Sciences, 12*(13), https://doi.org/10.3390/app12136809

The Game Gal. (2018) *The dot game*. Retrieved October 17, 2024 from https://www.thegamegal.com/2010/07/25/the-dot-game/

National Governors Association Center for Best Practices and Council of Chief State School Officers. (2010). *Mathematics standards.* Retrieved May 20, 2025 from https://www.thecorestandards.org/Math/

Sukriadi, S., Kusdar, K., Djangka, L., Muhlis, M., Febiola, D., & Salim, N. A. (2023). Feasibility of developing creative mathematics learning media augmented reality building materials. *Technium Social Sciences Journal, 40*, 139–147. https://doi.org/10.47577/tssj.v40i1.8480

About the Authors

Brian T. Johnson is the School Library Media Specialist at Lakeside Junior High School in Northwest Arkansas. He also serves as an adjunct professor for the University of Memphis in the Instruction and Curriculum Leadership department. His research interests include information literacy, coding in K-12 education, makerspaces, successful technology integration and flipped learning. He can be contacted at b.t.johnson16@gmail.com.

Rebecca Bramwell is an Instructional Facilitator at Lakeside Junior High School in Springdale, Arkansas. She has served as a math teacher for ten years. She holds a Math degree from Eastern Illinois University and a Master's degree in Administration and Leadership from William Woods University. She is a Google Certified Educator and Trainer. Her favorite thing about education is the opportunity to collaborate with others to create meaningful learning opportunities for students.

Josef Stamps is an 8th grade math teacher at Lakeside Junior High School in Arkansas. He is a graduate of John Brown University and earned his Master's degree in teaching through Arkansas State University. In his free time, he enjoys being outdoors, reading, and playing board games.

Katie Patterson is an 8th grade math teacher at Lakeside Junior High in Springdale, AR. She is a graduate of Kansas State University and received her M.Ed from Clemson University. In her free time, she enjoys hiking, reading and playing with her kids.

Kyle Jones is an 8th grade math teacher and football coach at Lakeside Junior High in Springdale, Arkansas. He graduated from Harding University with a bachelor's degree in Middle Level Math & Social Studies Education. In his free time, he enjoys fishing, hiking, and spending time with his wife.

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9329

Scratch Encore: Creating and Sustaining Culturally Responsive Computer Science Education

Rasha Alkhateeb[1], J. Elisabeth Kasner[2], David Weintrop[1], Jennifer Palmer[3], Merijke Coenraad[4], Minh Tran[3], and Diana Franklin[3]
[1]University of Maryland, [2]Florida State University, [3]University of Chicago, [4]Digital Promise

Overview

Scratch Encore (Canon Lab, n.d.) is a culturally relevant, student-centered, 14 module, computer science curriculum for 4th to 8th-grade learners that introduces foundational computing topics using the Scratch environment. It employs three key design goals: (a) supporting teachers, (b) supporting learners, and (c) using culturally responsive practices to address long standing inequities in computing. The curriculum offers equitable and effective learning experiences for students who have historically not had equal opportunities to fully participate in computing while providing a wide array of supports for educators who may be inexperienced with Scratch and/or programming. This article features a high-level overview of Scratch Encore and the first 6 modules in greater detail to help teachers understand the content and pacing of the curriculum.

Topics: Computer Science Education, Culturally Responsive Pedagogies, Programming

Time: 14 modules. Each module is approximately 2-4 lessons. Each lesson takes 60-120 minutes.

Materials

- Scratch Encore (Canon Lab, n.d.) registration
  - Materials provided in Scratch Encore (module lesson plans, student worksheets, worksheet answer keys, Scratch Encore assessments, videos introducing focal concepts). Refer to Support materials at the end of this article for additional sources.
- Student devices with keyboards (1:1 or 1:2)
- Student Scratch accounts
- Projector/Screen

Context-at-a-Glance

Setting
4th-8th grade classrooms with a focus on computer science and/or programming.

Modality
Face-to-face, hybrid, or online (synchronous or asynchronous)

Class Structure
14 modules with 2-4 lessons each. Lessons vary from 60-120 minutes.

Organizational Norms
Modules have a common, Use→Modify→Create gradual release structure. The content is cumulative so modules should be completed in order. Students should work individually or in pairs.

Learner Characteristics
Scratch Encore's modules, specifically designed to support learners from groups underrepresented in computing, can be used to address the challenges of bringing high-quality CS instruction to historically excluded and minoritized students in computing.

Instructor Characteristics
Scratch Encore is designed for 4th-8th grade teachers. The curriculum offers numerous teacher supports, including teacher guides, assessments, and videos.

Development Rationale
This work supports teachers in teaching CS in a culturally responsive way. It provides a deep dive into computing concepts in the Scratch Environment.

Design Framework
Constructionist Design Theory; Leverages the Use→Modify→Create Pedagogical Approach, blending structured and open-ended activities.

Setup

Scratch Encore (Canon Lab, n.d.) was designed to allow for teacher choice as they work to meet the needs of their students (Franklin et al., 2020). In the first 6 lessons, teachers choose which thematic strand they want to use (multicultural, youth culture, or gaming) and then use the provided materials in their classroom.

Across the Scratch Encore curriculum, teachers will introduce computing concepts by situating them in themes that resonate with students. Students will also have the opportunity to create Scratch projects that reflect their own interests and ideas.

Teachers can choose between digital and print versions of the student resources. Scratch Encore is designed to be used in the classroom but can be used for remote, hybrid, and asynchronous instruction.

To participate, students need a device with a keyboard that can run Scratch. Teachers may find that a projector or large screen can be helpful to display curriculum materials to the class.

Standards

Scratch Encore supports the following standards:

- Computer Science Teachers Association (CSTA, 2017) standards:
  - 1B-AP-10; 1B-AP-11; 1B-AP-12; 1B-AP-13; 1B-AP-14; 1B-AP-15; 1B-AP-17; 2-AP-10; 2-AP13; 2-AP-16; 2-AP 19; 1B-1C-20; 1B-IC-21
- International Society for Technology in Education (ISTE, 2016) standards for students:
  - 1c; 1d; 2b; 4a; 4c; 4d; 6b; 6d; 7a

Context and Setting

Scratch Encore is aligned with CSforAll (n.d.), a national movement to bring high-quality computer science education to all students. The Scratch Encore curriculum fulfills a specific need in upper-elementary and middle school education for culturally responsive computer science education.

Scratch Encore is designed to live between first-touch computer science experiences and more advanced curricula that use text-based programming languages. As such, the curriculum is designed for 4th-8th grade students learning computer science. The curriculum was initially designed for face-to-face classroom instruction but, in response to the COVID-19 pandemic, the curriculum can now also be taught fully virtually as all materials (e.g., activities, assessments) can be completed through online forms. Each module has a common structure relying on various, research-backed pedagogical approaches. The details of the curriculum are more fully described in the Learning Representation section.

Scratch Encore consists of 14 modules with 2-4 lessons per module. Lessons vary from 60-120 minutes depending on the prior experience of the students and the pace the teacher sets. While the full curriculum may be completed across multiple school years, many teachers use only the first few modules due to time constraints. The modules are cumulative, with later modules building on the concepts and skills introduced in earlier modules.

Scratch Encore was created with three key design goals in mind: (a) supporting teachers, (b) supporting learners, and (c) being culturally responsive to address long-standing inequities in computing.

To support teachers, the curriculum is fully featured, meaning it includes detailed lesson plans, worksheets with answer keys, automated assessments, supporting videos, and additional resources related to multicultural topics. Further, there is an online, asynchronous Scratch Encore professional development course that teachers can elect to join (see edX, n.d.). Collectively, these scaffolds make it possible for teachers with little prior computer science or Scratch experience to successfully teach the curriculum in their classrooms.

To support students, Scratch Encore provides a scaffolded introduction to the Scratch environment, programming concepts, and computer science more generally. Students have the opportunity to engage with computing content and create personalized projects that reflect their ideas and interests. The curriculum was designed using research backed pedagogical strategies including TIPP&SEE (Salac et al., 2021), Use→Modify→Create (Franklin et al., 2020), and structured planning sheets to help students create their own projects (Tsan et al., 2022).

Finally, towards the goal of making a culturally responsive curriculum, Scratch Encore includes three thematic strands. Each strand situates the content in a different theme, allowing the teacher to choose how to present the content and providing opportunities for them to enrich the context based on their own knowledge and experiences. These themes were co-designed with educators, students, and parents (Coenraad et al., 2019). When explaining concepts, examples can be drawn from students' current knowledge and existing prior experiences (e.g., Scott et al., 2015).

Learning Representation

In this section, a high-level overview of Scratch Encore (Canon Lab, n.d.), including its core ideas, its structure, and how it supports students and teachers, is presented.

Each Scratch Encore module is situated within a specific youth-oriented theme. The themes used in the curriculum were designed in collaboration with educators, students, and parents to ensure that they resonate with middle school learners (Coenraad et al., 2019). Modules 1-6 contain three thematic strands that situate CS content: multicultural, youth culture, and gaming.

The curriculum draws on constructionist learning theory, which emphasizes hands-on learning where students construct personally meaningful artifacts (Papert, 1980), which, in this case, are Scratch programs. Each module utilizes a Use→Modify→Create structure to scaffold knowledge of that concept.

Scratch Encore's unique stranded structure empowers teachers to select the version of the module that they think will resonate with their students most. The content and learning goals are the same across the thematic strands to ensure students gain content mastery regardless of which strand the teacher selects. In the section, how each of these design goals are achieved is presented.

The Scratch Encore Curriculum

Scratch Encore comprises 14 modules, each covering a different topic of growing conceptual complexity. Each lesson is situated within a specific theme, with the first 6 lessons having 3 themes the teacher can choose from: multicultural, youth culture, and gaming.

Each strand situates the content of the activities within a specific context designed to relate to learners while covering the exact same CS material (see Figure 1). In teaching the first 6 modules, teachers can decide which thematic strand they wish to use based on the theme and their students. Given the content is the same, teachers can move between strands as they progress from module to module.

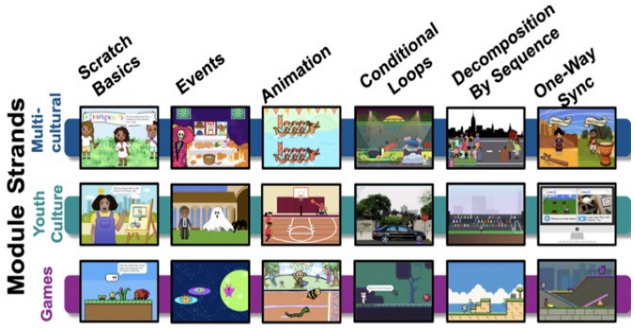


Figure 1: Scratch Encore module display of strands.

A feature of programming is that its concepts and practices are not tied to a specific context. Thus, it is possible to teach a concept, such as conditional loops, using different themes, such as waiting for a ride (transportation), Brazil's Carnival celebration, or a side-scrolling video game. Situating computing content in different themes can help students see their broad applicability and help them better relate to the content (Ladson-Billings, 2009). In this way, Scratch Encore draws on students' prior knowledge, cultural practices, and interests to help them connect with the content and feel a greater sense of belonging.

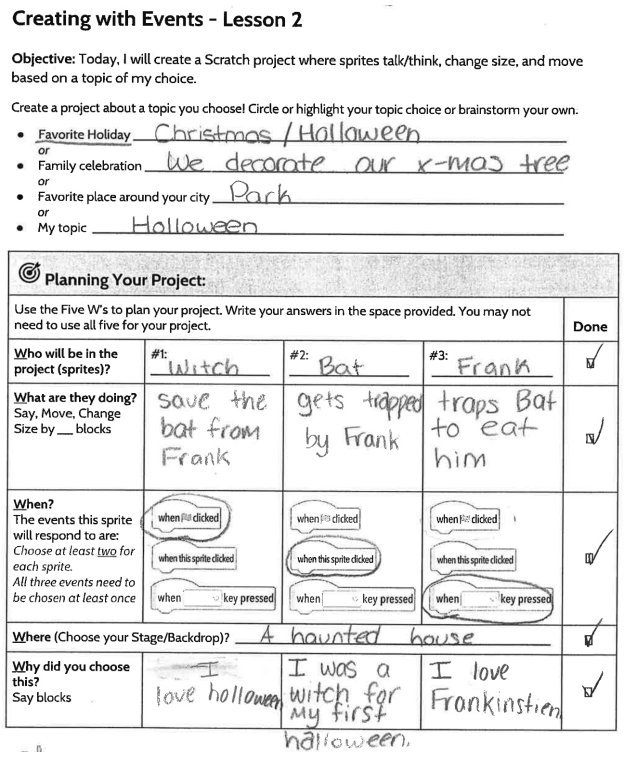
In terms of an overall pedagogical design approach, Scratch Encore's design draws from constructionist design principles which emphasize hands-on activities and give learners the opportunity to personalize the learning experience. To accomplish this, each module of the curriculum employs the Use→Modify→Create pedagogical approach. Use→Modify→Create is a research-backed, highly scaffolded progression of activities used to introduce learners to foundational computer science concepts (Lytle et al., 2019).

Learners begin by *Using* the focal CS concept before being asked to *Modify* a program that includes that

concept. Students do this by interacting with Scratch projects provided by the curriculum. Having seen the concept and modified its behavior, students then *Create* their own programs using the concept. Use→Modify lessons utilize TIPP&SEE, a novel learning strategy that helps students navigate the Scratch interface while learning from example projects.

The structure and thematic design of the *Use* and *Modify* projects provide a means for integrating cultural ideas into the curriculum. For example, students can be introduced to Events by using and modifying a Scratch project about creating an ofrenda for Día De Los Muertos or learning how to decompose a program by working on a project about soccer. This design provides a way to blend cultural and youth interest topics with computing content.

To facilitate the final, *Create,* projects, students are provided with a worksheet to help them brainstorm a topic, think through how they will use the focal concepts, and begin to plan out exactly what their scratch project will look like (see Figure 2). For example, to facilitate the *Create* project, students are provided creative prompts, like "What is your favorite sport?," that empower them to tailor projects to their own interests and allow them to choose how to connect to their personal culture (Ryoo et al., 2013).



Creating with Events - Lesson 2

Objective: Today, I will create a Scratch project where sprites talk/think, change size, and move based on a topic of my choice.

Create a project about a topic you choose! Circle or highlight your topic choice or brainstorm your own.

- Favorite Holiday Christmas / Halloween
  or
- Family celebration We decorate our x-mas tree
  or
- Favorite place around your city Park
  or
- My topic Halloween

Planning Your Project:

Use the Five W's to plan your project. Write your answers in the space provided. You may not need to use all five for your project.				Done
Who will be in the project (sprites)?	#1: Witch	#2: Bat	#3: Frank	☑
What are they doing? Say, Move, Change Size by ___ blocks	save the bat from Frank	gets trapped by Frank	traps Bat to eat him	☑
When? The events this sprite will respond to are: *Choose at least two for each sprite. All three events need to be chosen at least once*	when flag clicked; when this sprite clicked; when ___ key pressed	when flag clicked; when this sprite clicked; when ___ key pressed	when flag clicked; when this sprite clicked; when ___ key pressed	☑
Where (Choose your Stage/Backdrop)? A haunted house				☑
Why did you choose this? Say blocks	I love hallowen	I was a witch for my first halloween.	I love Frankinstien	☑

Figure 2. Example Module 2 create worksheet.

This structured approach and the associated scaffolds (e.g., worksheets, provided Scratch projects) support teachers who are new to computer science. Additionally, the lessons are designed to be flexible so they can be modified based on available classroom time, the speed at which students are learning the material, and whether they are encountering the curriculum for the first time or reviewing it in a later year.

Scratch Encore's Student Supports

The Scratch Encore curriculum includes various resources and strategies to support students in learning to program and having a positive computer science learning experience. These supports include mnemonic devices to help with authoring, debugging, and submitting Scratch projects as well as worksheets to help them plan and execute their ideas for Scratch projects. These resources also support students in personalizing the learning experience and encourage them to bring their own interests, ideas, and values into the programs they are creating.

Student Worksheets

Writing programs is difficult. This includes both technical and conceptual challenges and deciding what the program will do and figuring out how to write the program to accomplish the envisioned goal. Scratch Encore provides worksheets to help students with each phase of the Use→Modify→Create process. Each Scratch Encore worksheet is available as a Google Doc, PDF, or Google Form.

In the *Use* phase, the Scratch Encore curriculum offers worksheets to help students with the TIPP&SEE process (discussed in the next section). This includes reminding students of each step in the process and asking specific questions about the project that require students to attend to different aspects of the project (e.g., the sprites, behaviors when different events occur). These worksheets help students learn how to *read* Scratch projects and make sense of their behaviors.

The *Modify* worksheets (see Figure 3) help students plan how they are going to modify the provided Scratch project. This includes having them think through the sprites they want to add/modify and the behaviors they want those sprites to carry out. Additionally, they include clearly defined tasks that

students mark as complete, so students can make sure they accomplish all they set out to do.

For students that need additional scaffolding, the curriculum also includes step-by-step instructions that walk students through each step of the *Modifying* process, allowing students to track progress to make sure they have done everything that needs to be done.

The *Create* worksheets (see Figure 2) provide planning prompts for students as they design a new project from scratch, defined tasks for programming, and reflection prompts. This worksheet helps students decide on a topic for their *Create* project and then think through how they are going to use the focal concept in their program. Additionally, it helps to make sure the resulting project includes the focal concept of the module.

Events - Modify Project

Strand: Multicultural
Objective: Today, I will modify a Scratch project to use different events to trigger actions.

Planning Your Project:	
Pick three special people who have passed away to be on your Ofrenda. Traditionally, people on Ofrendas are from our families, but you can also choose a famous person who has passed away.	Done
Special Person 1: _____ Favorite Memory: _____	☐
Special Person 2: _____ Favorite Memory: _____	☐
Special Person 3: _____ Favorite Memory: _____	☐

Modify Tasks:		
Setup:	Done	
• Reload, Remix, Share, and +Add to Studio (RoaR and hiSS) from the Modify project: https://scratch.mit.edu/projects/323737554/.	☐	
• Choose costumes for Special Person 1, Special Person 2, and Special Person 3.	☐	
Now implement your plan for the Left Sprite:	Coded	Tested
• `When this sprite clicked`, it gets larger, says Special Person 1's name and favorite memory, then gets smaller.	☐	☐
Now implement your plan for the Middle Sprite:		
• `When this sprite clicked`, it gets larger, says Special Person 2's name and favorite memory, then gets smaller.	☐	☐
Now implement your plan for the Right Sprite:		
• Copy the 2 scripts from the Middle sprite.	☐	☐
• `When this sprite clicked`, it gets larger, says Special Person 3's name and favorite memory, then gets smaller.	☐	☐

Figure 3. Module 2 multicultural modify worksheet.

TIPP&SEE Strategy

TIPP&SEE is a scaffolding approach that uses reading comprehension strategies to help learners break down a Scratch project to understand how it behaves and why (see Figure 4). The approach is based on research teaching students how to read and is a key feature of the *Use* portion of the module where students are being introduced to a new topic. Research on the TIPP&SEE strategy shows that it improves student learning, especially among students who have historically not succeeded in computing contexts, including students from racial groups historically excluded from computing contexts, students performing below grade level, and students with learning disabilities (Salac et al., 2021).

The TIPP portion guides students in previewing various components of a Scratch project before they examine any code. Students begin by finding key project features: *Title*, *Instructions*, and *Purpose*. Then they run the code while carefully observing events and actions. Students are encouraged to *Play* the project and accompanying code to see what it does. As students engage in purposeful play, they are prompted to interact with specific elements of a Scratch project to observe what happens after various events, such as 'click the green flag' or 'click on a sprite.'

Start with TIPP&SEE!
Get a TIPP from the Project Page:
Title: What is the title of the project? Does it tell you something about the project?
Instructions: What do the instructions tell you to do?
Purpose: What is the purpose of this activity? What will this code teach you?
Play: Run the project and see what it does! Look at which sprites are doing the actions.

What happened when you played the project? Circle or highlight the action(s) that happened for each event.

When I clicked the green flag:				
	talked waved did nothing		talked changed size did nothing	 talked changed size did nothing	 talked changed size did nothing
When I pressed the space bar:				
	talked waved did nothing		talked changed size did nothing	 talked changed size did nothing	 talked changed size did nothing
When I clicked on the left picture  **:**				
	talked waved did nothing		talked changed size did nothing	 talked changed size did nothing	 talked changed size did nothing
When I clicked on the middle picture  **:**				
	talked waved did nothing		talked changed size did nothing	 talked changed size did nothing	 talked changed size did nothing

Figure 4. Module 2 TIPP&SEE worksheet.

The SEE portion helps students locate specific code related to the focal CS concept within the Scratch project (*Sprites* and *Events*). They click the See Inside button to explore the code that implements what they observed in the TIPP section. Then, students *Explore* the code to make suggested changes to the example code and note the project's behavior after the changes. The *Explore* questions on the TIPP&SEE

worksheets are designed to highlight specific things that students should notice before they start the Modify tasks.

TIPP&SEE outlines a systematic process for students to explore and understand a Scratch project and how the code functions through intentional experimentation and meaning-making.

WHAT?!? A MESS Debugging Strategy

Debugging is an essential aspect of programming but is often overlooked in formal instruction. To support students in debugging their projects, Scratch Encore developed the *WHAT?!? A MESS* debugging strategy. The acronym outlines strategies for identifying what the bug is as well as potential causes and steps to resolve it.

Students start by identifying *What* the programmer intended the project to do and then find *How* the program is behaving differently than expected. Next, students *Analyze* sprite behavior and code. If a student cannot debug on their own, they are encouraged to *Talk* to their peers to find a solution.

While analyzing code, students look for A MESS that might be causing an issue in their code:

- *Arguments*: A number in a white circle is incorrect.
- *Missing*: Block(s) are missing.
- *Extra*: Extra block(s).
- *Scrambled*: Blocks are out of order.
- *Substitute*: Used the wrong block(s).

Scratch Encore's Teacher Supports

In an effort to be easily adoptable by teachers, Scratch Encore includes numerous resources to help teachers. All of the Scratch Encore resources and materials are organized into a single page for each module. Figure 5 shows the three Module 2 strands (depicted by the three colored columns) and the full suite of supporting materials, including videos, teacher resources, worksheets, and assessments. Each module has a page like this with all instructional materials in one place. These pages can be found at Canon Lab (n.d.) after completing a brief, free registration. The registration is used to keep track of who has reviewed the materials to share this information with stakeholders.

Module 02 - Events (M2)	Multicultural 		Youth Culture 		Gaming 	
Unit Plans	Unit Plan Google Doc	Unit Plan pdf	Unit Plan Google Doc	Unit Plan pdf	Unit Plan Google Doc	Unit Plan pdf
Lesson 1 (M2L1) - Exploring Events (60-120 minutes)	M2L1 Engage Video	YouTube	M2L1 Engage Video	YouTube	M2L1 Engage Video	YouTube
	Teacher/Demo Project		Teacher/Demo Project		Teacher/Demo Project	
	Student Project		Student Project		Student Project	
	TIPP&SEE	pdf / Google Form	TIPP&SEE	pdf / Google Form	TIPP&SEE	pdf / Google Form
	TIPP&SEE Answer Key		TIPP&SEE Answer Key		TIPP&SEE Answer Key	
	L1 Modify Sheet	pdf / Google Form	L1 Modify Sheet	pdf / Google Form	L1 Modify Sheet	pdf / Google Form
	L1 Step-by-Step	pdf	L1 Step-by-Step	pdf	L1 Step-by-Step	pdf
	Día de los Muertos Information Sheet		N/A		N/A	
Lesson 2 (M2L2)- Creating with Events (120 minutes)	L2 Create Sheet	pdf / Google Form	L2 Create Sheet	pdf / Google Form	L2 Create Sheet	pdf / Google Form
Assessment	Google Doc		pdf	Google Form	Answer Key	

Figure 5. Module 2 strands and events.

Lesson Plans and Class Resources

Each module in Scratch Encore includes a comprehensive unit plan, covering the focal computing content of the module, discussion prompts, details about the specific Scratch projects and activities students will work through. The lesson is also mapped to ISTE and CSTA standards (see Figure 5). The curriculum also includes posters related to topics like TIPP&SEE and WHAT?!? A MESS to help students as they work through the curriculum.

Each lesson plan begins with a high-level overview of the focal content of the lesson before outlining how each lesson in the module progresses. While not intended to serve as a script, the Unit plan does include prompts for opening discussion and closing reflections. It also includes links to the materials that are used in each lesson along with helpful pointers for the teacher to be aware of as they teach the lesson, including common questions students ask and challenges they face.

For each of the modules in the multicultural strand, teachers are provided with an information sheet about the specific theme used. For example, the multicultural theme used in Module 3 is dragon boat racing. The Dragon Boat Information Sheet provides a description of dragon boat races, their cultural significance, their history, and links to additional resources about the topic. Finally, the curriculum includes formative and summative assessments, including automated project assessments—discussed in greater detail in the next sections.

Content Videos

Each module includes a short (usually around 4 minutes) video introducing the core computing concept of the lesson. These videos can be shown by the teacher in class or shared with the students for them to review throughout the lesson if they need additional support. The videos use Scratch animations to define and demonstrate the core concept of each module.

Assessments

The Scratch Encore curriculum provides both formative and summative assessments. For formative assessments, the *Use* and *Modify* worksheets (discussed in the Student Resources section) include multiple-choice and short-answer questions for students to answer. Answer keys are provided for both sets of worksheets.

For summative assessments, each module includes a quiz that can be administered at the end of the module asking students questions about the focal topic of the lesson. The quizzes are multiple-choice and often include images of Scratch projects and scripts that students have to interpret (see Figure 6). Like the worksheets, the summative quizzes are available in Google Doc, Google Form, and PDF formats.

Finally, Scratch Encore provides teachers with an automated Scratch project checker that allows them to check to see if a student's *Create* project includes the required content. To use the Scratch Encore Student Project Checker (n.d.), you input the URL for the Scratch project and it reports which required aspects of the *Create* project were implemented. This tool can also be shared with students so they can check their own progress as they work on their *Create* projects. Additionally, teachers can use the Scratch Encore Studio Grader (n.d.) to monitor multiple students' work at a time (see Figure 7).

For this tool, the teacher inputs the Studio URL where student projects are stored. The tool runs the Student Project Checker on each program, providing results for the whole class. It is recommended to double-check students' work if the checker thinks something is missing because students can use creative solutions the checker does not anticipate.

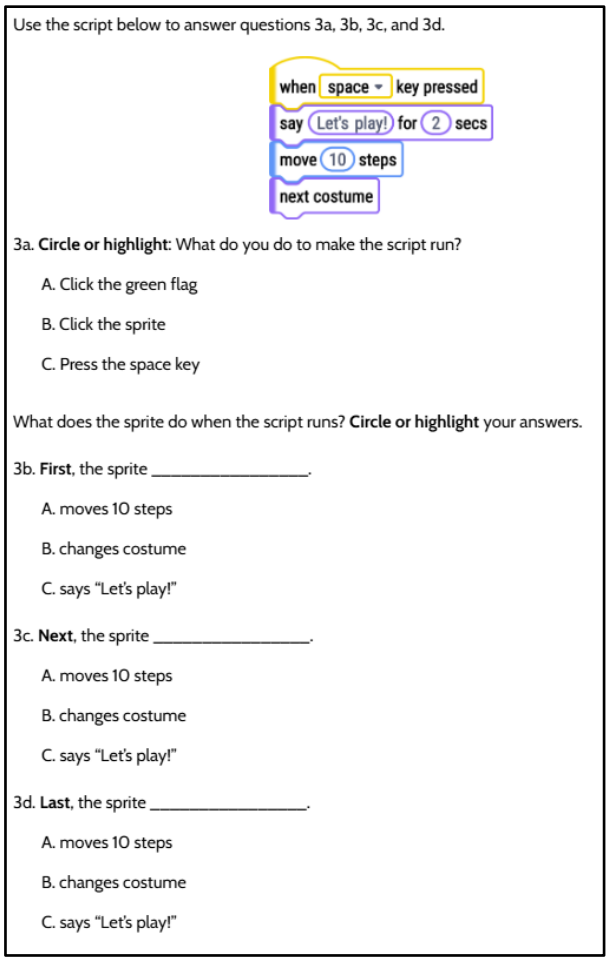


Figure 6. A Module 2 summative assessment section.

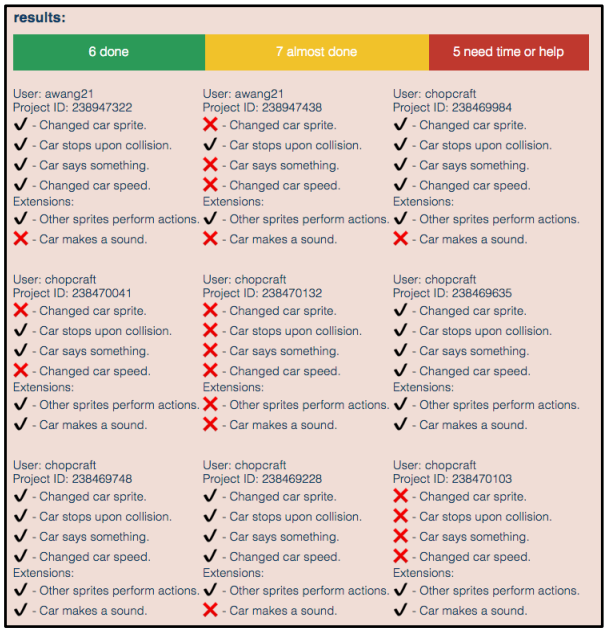


Figure 7. Studio grader example.

Scratch Encore Modules 1-6 Overview

In this section, the first 6 modules are presented in greater detail to help teachers understand the content and pacing of the curriculum. As a reminder, each of these first 6 lessons can be taught using three different themes: multicultural, youth culture, or gaming.

Module 1 - Scratch Basics

Module 1 has three lessons.

Length: Lesson 1: Introducing Scratch (60 minutes), Lesson 2: Exploring Scratch (60 minutes), Lesson 3: Creating with Scratch (60 minutes)

Overview: This module orients students to the Scratch programming environment by introducing color-coded block categories, block shapes, and the basic components of a Scratch project (e.g., sprites, backdrops). Students explore the Scratch environment in a scavenger hunt activity, modify an existing project, and create a new project of their own design.

Objectives: Students will be able to:

- Reload, remix, share, and add Scratch Projects to a Studio.
- Add a backdrop to a blank Scratch project.
- Create and delete sprites.
- Start projects with the green flag.
- Stop projects with the stop sign.
- Add text to a sprite using say__for__secs.
- Add movement to a sprite using move__steps.
- Place a sprite in a specific location using go to x:__y:__.
- Create a project that includes text and step movement.

Module Themes:

- Multicultural: Holi celebration
- Youth culture: Communities
- Gaming: Helen the Hedgehog

Module 2 - Events

Module 2 has two lessons.

Length: Lesson 1: Use/Modify (60-120 minutes), Lesson 2: Create (120 minutes)

Overview: In event-based programming, scripts are triggered when specific events occur (e.g., a sprite is clicked, a key is pressed). In this module, students learn to use a variety of events to trigger scripts that resize sprites or cause a sprite to say something.

Objectives: Students will be able to:

- Define an event.
- Write scripts to run when a specific key is pressed.
- Write scripts to run when a specific sprite is clicked.
- Change and set the size of sprites.
- Copy a script from one sprite to another.

Module Themes:

- Multicultural: Día De Los Muertos Ofrenda
- Youth culture: Fantasy - Wizard and Ghost
- Gaming: Race in Space

Module 3 - Animation

Module 3 has two lessons.

Length: Lesson 1: Use/Modify (120 minutes), Lesson 2: Create (120 minutes)

Overview: A sprite can be animated by repeating sets of blocks that alternate between switching costumes or moving to the next costume along with repeated step movement. Adjusting the wait times of these repeated actions will increase or decrease the speed of the animation. Naming each of the costumes and changing the size of the sprites can assist in the creation of the animation.

Objectives: Students will be able to:

- Animate a sprite at one location using a repeat loop and multiple costumes.
- Animate a sprite with movement using a repeat loop.
- Distinguish between the functionality of switch costume and next costume blocks.

Module Themes:

- Multicultural: Dragon Boat Festival
- Youth culture: Basketball
- Gaming: Animal Races

Module 4 - Conditional Loops

Module 4 has two lessons.

Length: Lesson 1: Use/Modify (120 minutes), Lesson 2: Create (120 minutes)

Overview: Within a conditional loop, repeated actions will continue until a specific condition becomes true. In Scratch, the condition that is fulfilled can relate to another sprite or a specific color. In this module, students will implement conditional loops that control the repeated actions of sprites. The sprites will stop moving when the condition of the loop becomes true due to a sprite reaching a color or another sprite.

Objectives: Students will be able to:

- Create an animation with one costume and repeated movement.
- Write a script using a conditional loop that repeats an action until a condition is true.

Module Themes:

- Multicultural: Carnival Parade
- Youth culture: Waiting for a Ride
- Gaming: Ninja Cat & Gem

Module 5 - Decomposition by Sequence

Module 5 has two lessons.

Length: Lesson 1: Use/Modify (120 minutes), Lesson 2: Create (120 minutes)

Overview: A series of actions by multiple sprites can be decomposed based on the order in which the actions occur and the conditions that cause one action to stop and another to start. To decompose a project into a sequence of actions, it is necessary to identify the actions that occur based on certain events. In this module, students identify a set of events based on sensing conditions (e.g., when one sprite touches another sprite) and the actions that occur as a result of those sensing conditions.

Objectives: Students will be able to:

- Decompose a sequence of events.
- Create scripts that will trigger the action of one sprite dependent on the action of another sprite.
- Use sensing blocks to stop and start actions.
- Plan and create an animation based on a set of events and actions.

Module Themes:

- Multicultural: Protest Marchers
- Youth culture: Soccer
- Gaming: Fix the Game

Module 6 - One-Way Synchronization

Module 6 has two lessons.

Length: Lesson 1: Use/Modify (120 minutes), Lesson 2: Create (120 minutes)

Overview: Synchronization is the coordination and timing of actions *between sprites*. Previously, events were all directly from the user. However, a single sprite is limited in what events it can sense. For example, a sprite cannot sense when a user clicks on a different sprite. In one-way synchronization, one sprite sends an invisible message to other sprites, thereby initiating an event. This is called message passing - a single communication event between two or more sprites takes place in which one sprite passes a message, and one or more sprites receive the message and respond accordingly.

Objectives: Students will be able to:

- Create scripts that use message passing for one-way synchronization between two sprites.
- Create scripts that use message passing for one-way synchronization that initiate simultaneous actions on multiple sprites.

Module Themes:

- Multicultural: Music, Navajo flute and Djembe
- Youth culture: Encore TV
- Gaming: Remote Control Cars

Deep Dive - Module 2: Events

During this section, italic text identifies questions or prompts for the learners.

Having presented the content of the first 6 modules, a more detailed look into one module is provided. The goal here is to provide a greater sense of exactly what it is like to teach Scratch Encore. For this

section, Module 2: Events was chosen. In this module, students are introduced to event-based programming. They identify the events that sprites should respond to and program the actions that should be carried out if that event occurs.

There are two lessons in the Events module: Lesson 1: Exploring Events (60-120 minutes) and Lesson 2: Creating with Events (120 minutes). In both Lessons 1 and 2, teachers can select from three themes to use in their classroom: multicultural, youth culture, or gaming. Following instruction, the module assessment uses multiple-choice questions to gauge students' mastery of using events to trigger an action.

To gain access to the specific worksheets and materials for this module, register for free at Canon Lab (n.d.).

Module 2 Themes

Module 2 has three themes' teachers can choose from: multicultural, youth culture, or gaming.

Multicultural Theme: Students use and modify a project that depicts a Día de los Muertos ofrenda. They begin with an ofrenda featuring grandparents, a family pet, and offerings. When picture frames are clicked their inhabitants grow and share a special memory. Students can also use the spacebar to learn more about the holiday. After exploring the events in the teacher/demo project, students modify the ofrenda to include people who are important to them and a favorite memory. Accompanying the lesson materials is an Information Sheet about Día de los Muertos, ofrendas, and the cultural significance of the holiday in Mexican culture.

Module 2 Youth Culture Theme: Students use and modify a project that depicts a wizard and a ghost. When characters are selected, they grow, and the spacebar causes the wizard to cast a spell shrinking the ghost. After exploring the events in the teacher/demo project, students modify the scene to make the ghost grow when clicked and code a previously unused cat sprite.

Module 2 Gaming Strand: Students use and modify a project that depicts two aliens, Bop and Beep, in flying saucers who are racing to save a third alien, Bork. When clicked, the saucers grow and provide directions for the race. The right arrow key causes Bop to move and the left arrow key causes Beep to move. Students can use the arrow keys to decide who saves Bork. After exploring the events in the teacher/demo project, students modify the race to change the text for Bop and Bork. Students also program events for aliens to grow and speak using the right arrow key, mouse click, and green flag.

Module 2 Deep Dive

Lesson 1: Exploring Events

In this lesson, the concept of Events in programming is introduced. Students create and modify scripts that handle basic events. A starter project is provided with sprites and example code that handles an event and then responds with an action. For this deep dive, the Día de los Muertos materials from the multicultural strand is featured.

Engage (20-30 minutes)

Introduction (10-15 minutes)

The lesson kicks off with an introductory discussion (10-15 minutes) that is designed to introduce the focal concept in a way that connects with students' everyday lives. To start, pose questions like, *What is an event? What do you do if someone taps you on the shoulder?* Through this discussion, students are introduced to events in programming and how they are used in Scratch. An animated video of the Events Engage discussion (Scratch Encore, 2020) is available for students who were absent, those who benefit from multiple exposures to content, or for online/asynchronous instruction.

TIPP&SEE (10-15 minutes)

The next portion of the lesson reintroduces students to the TIPP&SEE strategy (10-15 minutes). Announce that you will model part of TIPP&SEE and students will finish the rest themselves. Ask students if they remember what TIPP&SEE is from the previous lesson and explain they will begin looking at the project as a class first. Distribute the Module 2 TIPP&SEE worksheet and, if possible, display the Ofrenda Student Modify Project (ScratchEncore, 2019a; see Figure 8).

Ask students questions to lead them through TIPP&SEE, such as: *What does the title tell us? What*

is the purpose of looking at this code? What are the sprites' names and pictures? Share a new Scratch skill with students: copying scripts from one sprite to another with drag and drop. Select the Middle sprite. Drag the 'when green flag clicked' script from Middle to Right. Now those blocks are also in Right's scripts.

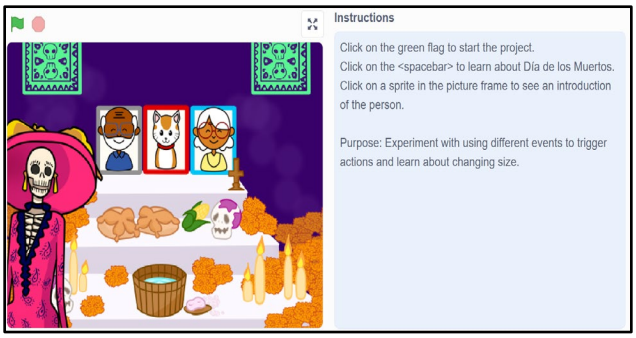


Figure 8. A Module 2 Ofrenda student modify project.

Explore (30-65 minutes)

TIPP&SEE (10-20 minutes)

Provide students time to use the TIPP&SEE strategy alone or with a partner (10-20 minutes). They should work through the TIPP&SEE strategy in more detail on their own.

Modify Ofrenda Project (20-45 minutes)

Next, students modify the Ofrenda Student Modify Project (20-45 minutes), using the Events Modify worksheet as a guide. A step-by-step worksheet is available as a means to support differentiation for students who need additional assistance. In this activity, students will copy the script from Left sprite to Middle and Right sprites, as well as personalize the project by selecting costumes and modifying the say blocks for all three sprites. Remind students to save and share their work before exiting Scratch.

You can also offer extension challenges to students. These challenges include changing the scripts that handle events to see how the functionality changes and adding scripts to handle additional events.

Reflect (10-25 minutes)

Students answer the reflection questions on the Events Modify worksheet (5-10 minutes). They will answer questions, such as: *How would you explain what an event is in computer programming? How is having different types of Events blocks helpful when coding?*

Consider having students present their projects to the class (5-15 minutes). You can also have students participate in a gallery walk where students run each other's projects on individual computers. If there is time, consider leading a discussion about how events could be used in their projects.

Lesson 2: Creating with Events

During the Create portion of Lesson 2, students use events as they create a brand new Scratch project about a topic of their choice.

Engage (15 minutes)

The lesson opens with a demonstration. Ask if anyone remembers what an event is in programming and if they remember which three event blocks were used in the last lesson. Display the Events: Ofrenda Sample project (ScratchEncore, 2019b), click See Inside, and then click on the Left sprite to reveal the blocks. Show what happens when you click the left sprite (on the stage) a few times (the sprite gets bigger, introduces itself, and shares a memory). Ask: *What block makes the sprite bigger each time the sprite is clicked?*

Click on the sprite again and stop the project using the stop sign while the sprite is talking so that it is stopped while it is big. Then, click the green flag to show that the sprite returns to its original size.

Ask the following questions to clarify the difference between: change size by ___ and set size to___%: *What block makes the sprite return to its original size? How are change size by ___ and set size to ___% different?*

Demonstrate these things enough times that it becomes clear that set size to ___% always sets the sprite to the same size while change size by ___ continually makes a sprite bigger (or smaller) each time an event occurs.

Explore (80 minutes)

In this portion of the lesson, students will create their own, new Scratch project using events (80 minutes). Introduce the activity by telling students they will

create a project using events. Distribute the Events Create worksheet and give students guidance throughout the project. This worksheet facilitates coming up with a Scratch project theme and planning out the sprites, events, and actions of the project.

To complete the task, students will:

- Create a new Scratch project.
- Brainstorm content for the project and plan how the project is going to be implemented.
- Add a backdrop.
- Select three sprites.
- Write scripts to handle all three types of events:
  - When green flag clicked
  - When this sprite clicked
  - When <key> Pressed
- Use say___ for___sec blocks to introduce themselves, say something about a topic, and describe what 'event' will cause them to move.
- Use change size by___ to change size, set size to ___% to initialize it when green flag clicked.

Explain to students that they can create any project of interest, but some ideas include:

- Describing a holiday you like to celebrate.
- Describing a family celebration and what makes it unique or special to your family.
- Introducing a favorite place around your city.
- A topic of your choice!

Remind students to run their project to test that it works as expected or debug and re-test as necessary. Ensure students Share and Add to Studio before exiting Scratch.

You can also offer an extension challenge to students. Have students add scripts to handle additional events (e.g., students make the sprites move when an event occurs; sprites get bigger when one event occurs but then get smaller when a different event occurs).

Reflect (10 minutes)

Students should finish the reflection questions included on the Events Create worksheet (10 minutes). The reflection includes questions like, *What was challenging about this project? How did you work through the challenges you faced? What is the difference between the change size by ___ and set size to ___% blocks?*

Afterward, consider having students present their projects to the class (15 minutes). You can also have students participate in a gallery walk where students run each other's projects on individual computers. If there is time, lead a discussion about how events could be used in other projects (e.g., map movements in a game to arrow keys, map different sounds to each sprite when it is clicked).

Assessment & Differentiation

Teachers and students can utilize the Scratch Encore Student Project Checker (n.d.) to make sure the project has all the required elements. The teacher can also use the Scratch Encore Studio Grader (n.d.) to monitor students' progress across the class. While the Automated Project Checker does a good job of reading student code for including specific tasks, it is recommended that teachers double-check submissions before use in grading, as sometimes students complete the task in a way the checker cannot anticipate.

Following Module 2, teachers can administer an end-of-module assessment. The Module 2 assessment consists of 6 multiple-choice questions related to events in Scratch. Students are required to read sample scripts to identify which event will cause a specific action, which scripts are initiated by a specific event, and what follows after a script is initiated.

The Events Module also offers various forms of differentiation. You can prepare block definition sheets for students to add to their journals. Options include: list the block names (students fill in the definitions) or provide the block names and definitions (students match block names to definitions). You can also provide a list of the type and number of blocks needed to complete each activity.

Critical Reflections

Lessons Learned

The Scratch Encore curriculum has been under active development since 2017. Over the last seven years, we have worked with countless teachers and students to create what we think is a high-quality,

easily adopted, and effective curriculum. Further, the curriculum has seen relatively widespread adoption, having been downloaded over 2,500 times by teachers and the Scratch projects from Module 1 having been viewed over 35,000 times and remixed over 11,000 times. Additionally, significant research has been conducted on the curriculum resulting in over 15 publications in peer-reviewed academic venues related to teacher experiences and student outcomes.

Over the course of the project, several key insights have helped make the curriculum what it is today. First, a key insight is the importance of providing a high level of support for teachers while also not being overly prescriptive. In this way, the curriculum can be taught by those new to computer science but also still provide ample room for teachers to personalize the instructional approach based on their own expertise, experience, and knowledge of the students. A good example of what this looks like in the curriculum is the inclusion of short videos that introduce the core concept of each module and specific prompts for leading class discussions. Some teachers may rely heavily on these resources while others may not use them at all.

A second key insight is the importance of providing step-by-step scaffolds for learners to help them through the Modify and Create activities. It is not enough to just help them with programming but also idea generation and mapping their ideas onto the core constructs of the lesson. Our research has shown that these worksheets helped students create more complex and correct Scratch programs.

A final insight from this work has been how valuable it is to have teachers and district leaders as partners in the design of classroom-ready curricula. Since the project began, we have had bi-weekly meetings with our district partners. Additionally, teachers and students have been close collaborators throughout the process. This has helped us create a curriculum that is easily adoptable and can work with a wide array of classroom and school contexts. This flexibility can be seen in the modular structure of the lessons, the fact that materials are available in multiple formats (e.g., PDF, Google Docs, Google Forms), and various supplementary supports and information, including standards mappings and information sheets for the multicultural strand modules.

All that being said, there are challenges associated with adopting this curriculum. Central among them is the need for dedicated classroom time to teach the materials. Scratch Encore has a fixed, regular structure and cumulative design that incrementally builds students' computer science knowledge and skills. In contexts where teachers have little time for computing instruction or instructional times come at irregular interviews, teaching Scratch Encore has been found to be challenging.

Cultural Relevance

One of the three focal design goals of this curriculum was to be culturally responsive. Several design decisions were made to accomplish this goal and support a diversity of students. First, the inclusion of multiple thematic choices for each module allows the teacher to tailor their instruction to the cohort of students they have. These learning modules are drawn from several sources, including projects that speak to students of specific cultural backgrounds as well as ideas generated by several participatory co-design sessions involving parents, students, and educators.

Second, we provide opportunities for students to express themselves through their projects. Our techniques range from customizing minor elements of projects (choosing what sprites say in modified projects) to the Create projects which have many dimensions on which students can be creative.

Finally, care was taken to make sure the sprites used in the shared Scratch projects reflect the racial, cultural, and gender diversity of the students we hope to present in classrooms. For example, the multicultural strand projects highlight cultural practices and celebrations from around the world.

Different students have different learning needs. Therefore, we have included support for a variety of learners. We provide differentiation through extension exercises and open-ended create activities. We also provide scaffolding through graphic organizers and activity sheets that support student planning through fill-in-the-blank diagrams. Finally, we include a new strategy we developed called TIPP&SEE to help students navigate the complex Scratch interface and focus on the learning goals of each activity.

Future Directions

Harmonizing Scratch Encore

While the Scratch Encore curriculum is complete, we are still actively working on it as a research project. Our current work is to explore ways to empower teachers to create their own versions of Scratch modules that draw on the prior knowledge and cultural practices present in their specific classrooms, schools, and communities. We call this process *Harmonizing* (Tran et al., 2024).

To *Harmonize* a lesson, teachers create a new Scratch project and accompanying student worksheets. Like with the curriculum itself, we have developed a series of supports to facilitate both coming up with theme ideas and then creating the materials for the *Harmonized* module. To support this process, we are also developing a generative AI tool, called Conjurer.

Using Generative AI to Support Harmonizing

To facilitate the *Harmonizing* process, we have explored using publicly available generative AI to support teachers in the project ideation phase of the *Harmonizing* process. With a simple prompt template, a teacher can interact with GPT-3 to brainstorm multiple *Harmonized* project ideas. Many GPT-generated ideas can be efficiently remixed from a sample Scratch Encore project without major changes in the project description (Tran et al., 2025). Further, we are developing a streamlined, generative-AI-powered tool to help with every step of the *Harmonizing* process: from brainstorming project ideas to implementing complete instructional materials, including Scratch projects (code, images) and student-facing materials (Figure 9). The goal is to enable teachers to *Harmonize* a full Scratch Encore module in 20 minutes or less. These *Harmonized* materials can be taught in place of existing projects in the Scratch Encore curriculum, allowing a teacher to leverage the existing materials for other lessons and continue to use the auto-grading features: the Scratch Encore Student Project Checker (n.d.) and the Scratch Encore Studio Grader (n.d.)

Professional Development

To introduce teachers to the *Harmonizing* process, we are creating a new week-long professional development workshop focused on culturally responsive pedagogy, design strategies to support it, and an introduction to customizing Scratch Encore modules. The PD covers culturally relevant design principles in general as well as several common customization categories (e.g. literature, science, community), followed by instruction on how to match a theme to a particular activity and the process of customizing a Scratch Encore module. Teachers will leave the PD with at least two customized projects and the knowledge to quickly make more.

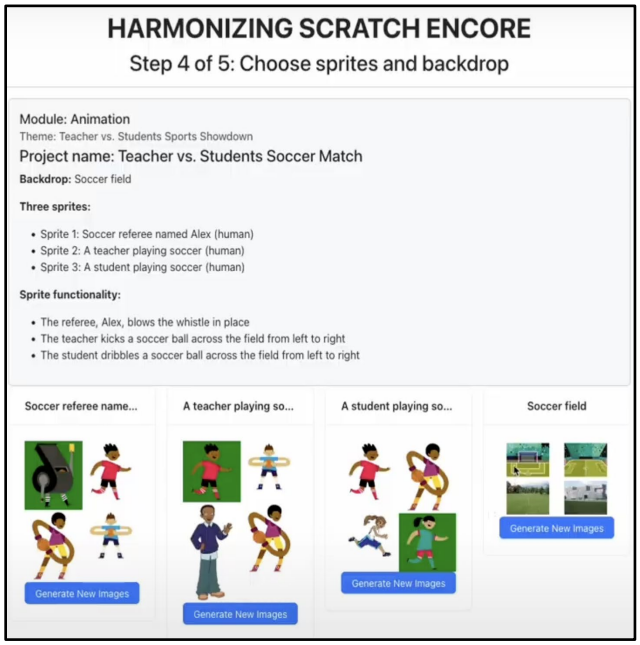


Figure 9. AI generator example.

References

Canon Lab. (n.d.). *Scratch Encore.* Retrieved May 17, 2025, from https://www.canonlab.org/scratchencoremodules

Computer Science Teachers Association. (2017). K-12 Standards. http://www.csteachers.org/standards

Coenraad, M., Palmer, J., Franklin, D., & Weintrop, D. (2019). Enacting identities: Participatory design as a context for youth to reflect, project, and

apply their emerging identities. *IDC '19: Proceedings of the 18th ACM International Conference on Interaction Design and Children*, 185-196. https://doi.org/10.1145/3311927.3323148

CSforAll. (n.d.). *Home*. https://csforall.org/

edX. (n.d.). *UChicagoX: Teaching coding in grades 5-8 with Scratch Encore*. Retrieved May 17, 2025, from https://www.edx.org/learn/teacher-training/university-of-chicago-teaching-coding-in-grades-5-8-with-scratch-encore

Franklin, D., Coenraad, M., Palmer, J., Eatinger, D., Zipp, A., Anaya, M., White, M., Pham, H., Gökdemir, O., & Weintrop, D. (2020). An analysis of use-modify-create pedagogical approach's success in balancing structure and student agency. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 14-24. https://doi.org/10.1145/3372782.3406256

International Society for Technology in Education. (2016). *ISTE standards: For students*. https://iste.org/standards/students

Ladson-Billings, G. (2009). *The dreamkeepers: Successful teachers of African American children* (2nd ed.). Jossey-Bass.

Lytle, N., Cateté, V., Boulden, D., Dong, Y., Houchins, J., Milliken, A., Isvik, A., Bounajim, D., Wiebe, E., & Barnes, T. (2019). Use, modify, create: Comparing computational thinking lesson progressions for STEM classes. *ITiCSE '19: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 395-401. https://doi.org/10.1145/3304221.3319786

Papert, S. (1980). *Mindstorms. Children, computers and powerful ideas*. Basic books.

Ryoo, J., Scott, S., & D'Angelo, C. (2013). Culturally responsive computing education: Bridging the gap between theory and practice. *In Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 235-240.

Salac, J., Thomas, C., Butler, C., & Franklin, D. (2021). Supporting diverse learners in K-8 computational thinking with TIPP&SEE. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 246-252. https://doi.org/10.1145/3408877.3432366

Scott, S. D., Sheridan, J., & D'Angelo, C. M. (2015). Culturally responsive computing instruction. *In Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 432-437.

Scratch Encore. (2020, August 6). *Scratch Encore module 2 engage: Events* [Video]. YouTube. https://youtu.be/zdilngtpeuY?si=8w-VuC5s7ePF654V

Scratch Encore Student Project Checker. (n.d.). Retrieved May 17, 2025, from https://people.cs.uchicago.edu/~dmfranklin/assessment/automated-assessment-master/projects.html

Scratch Encore Studio Grader. (n.d.). Retrieved May 17, 2025, from https://people.cs.uchicago.edu/~dmfranklin/assessment/automated-assessment-master/index.html

ScratchEncore. (2019a, August 9). *Events: Ofrenda - student modify*. Scratch. https://scratch.mit.edu/projects/323737554/

ScratchEncore. (2019b, August 9). *Events: Ofrenda teacher sample.* Scratch. https://scratch.mit.edu/projects/323737287/

Tsan, J., Eatinger, D., Pugnali, A., Gonzalez-Maldonado, D., Franklin, D., & Weintrop, D. (2022). Scaffolding young learners' open-ended programming projects with planning sheets. *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education*, 372-378. https://doi.org/10.1145/3502718.3524769

Tran, M., Gonzalez-Maldonado, D., Zhou, E., & Franklin, D. (2025). Can GPT help? Supporting teachers to brainstorm customized instructional scratch projects. *SIGCSE 2025: Proceedings of the 56th ACM Technical Symposium on Computer Science Education, 1,* 1134-1140. https://doi.org/10.1145/3641554.3701858

Tran, M., Killen, H., Palmer, J., Weintrop, D., & Franklin, D. (2024). Harmonizing Scratch Encore: Scaffolding K-8 teachers in customizing culturally responsive computing materials. *SIGCSE 2024: Proceedings of the 55th ACM Technical*

Symposium on Computer Science Education, *1*, 1335-1341. https://doi.org/10.1145/3626252.3630756

Support Materials

Scratch Encore Public Assessment Links. (n.d.). Retrieved May 17, 2025, from https://docs.google.com/document/d/e/2PACX-1vQdDrteCGMOUFlIE7zn0hbTOTnyARJSBGmbrCBLbQ0YQpunn2XRimR5QOHobQY0uLFctnCmBMkBGy5b/pub

Scratch Encore Student Project Checker. (n.d.). Retrieved May 17, 2025, from https://people.cs.uchicago.edu/~dmfranklin/assessment/automated-assessment-master/projects.html

Scratch Encore Studio Grader. (n.d.). Retrieved May 17, 2025, from https://people.cs.uchicago.edu/~dmfranklin/assessment/automated-assessment-master/index.html

About the Authors

Rasha Alkhateeb is a PhD Candidate in Literacy Education and a Writing Fellow at The Graduate School's Center for Writing and Oral Communication at the University of Maryland, College Park.

J. Elisabeth Kasner is an Assistant Professor of Elementary STEM Education in Anne's College at Florida State University. Her research focuses on the processes of science teacher identity and learning, elementary and middle level education, and integrated STEM+C.

David Weintrop is an Associate Professor and the Dean's Impact Professor in the Department of Teaching & Learning, Policy & Leadership in the College of Education with a joint appointment in the College of Information at the University of Maryland. His research focuses on the design, implementation, and evaluation of effective, engaging, and equitable computational learning experiences.

Jen Palmer is a Curriculum Specialist at the University of Chicago. For over 15 years, she has supported high-quality science, engineering, and computer science instruction in K-8 classrooms. Her work has ranged from curriculum and tool development to the design and facilitation of professional development opportunities and classroom coaching for teachers.

Merijke Coenraad is a Program Director focused on Inclusive Computing Research at Digital Promise. Her area of expertise includes co-design, research-practice partnerships, and equity in emerging technologies.

Minh Tran is a PhD student in Computer Science at the University of Chicago.

Diana Franklin is an Associate Professor in Computer Science at the University of Chicago where she leads the CANON (Computing for ANyONe) Lab. CANON Lab researches both 3rd-8th grade computer science interventions and quantum computing education for novices of any age with a particular focus on fostering more equitable learning experiences.

Acknowledgements

National Science Foundation grant assistance (Award #2201312) helped authors to produce article activities.

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9347

Artificial Intelligence-Enhanced Digital Storytelling: Empowering Young Creators in a Summer STEM Camp

Bulent Dogan and Amani Itani, University of Houston

Overview

This lesson engages students in grades 3–6 in creating digital stories enhanced by artificial intelligence (AI), encouraging logical thinking, creativity, and problem-solving. Using a structured, project-based format, students developed multimedia-rich narratives with support from AI tools for idea generation, writing refinement, and visual content creation. The students explored STEM role models, wrote story drafts, and edited videos using WeVideo. The project concluded with the production of AI-supported digital stories, assessed through a structured rubric.

Topics: Computational Thinking, Media Literacy, Project-Based Learning, Writing

Time: Multiple 60-120 minute sessions totaling 10-12 hours

Materials

- Laptops or tablets with internet access
- Headphones with microphones
- Projector or large display screen
- WeVideo (or other video editing software)
- AI Tools:
  - ChatGPT
  - QuillBot
  - NightCafe Creator
  - Adobe Podcast AI Enhance Speech
- Google Drive (or other resource) for file sharing
- Digital storytelling lesson plan
- DISTCO rubric for assessing digital stories
- List of STEM personas for students to choose

Context-at-a-Glance

Setting
A STEM summer camp for 3rd-6th grade students in Houston, Texas, USA.

Modality
Face-to-Face combined with online AI-assisted tools

Class Structure
Multiple 60-120 minute sessions equaling about 10-12 hours of work within a flexible collaborative space.

Organizational Norms
Instruction emphasizes constructivist and experiential learning, differentiated to meet student needs.

Learner Characteristics
88 3rd-6th graders participated in the camp. They had a variety of prior experiences with AI.

Instructor Characteristics
15 undergraduate mentors with technical proficiency in AI participated in the camp. They were previously trained to integrate AI into digital storytelling.

Development Rationale
Learning was designed to promote computational thinking through AI-driven digital storytelling.

Design Framework
Project-based learning (PBL) was used in this lesson to support deep, inquiry-driven engagement and real-world application (Larmer et al., 2015).

Standards

This learning representation aligns with the International Society for Technology in Education (ISTE, 2017) standards for students, fostering essential 21st-century skills like creativity, collaboration, critical thinking, and digital fluency. The following list of ISTE standards are met within this lesson and how further described in the article:

1. Empowered Learner – Students take ownership of story development.
3. Knowledge Constructor – Students curate and synthesize digital content.
4. Innovative Designer – Students apply design thinking through iteration.
5. Computational Thinker – Students analyze structure and logic in storytelling.
6. Creative Communicator – Students express ideas clearly using multimedia tools.

Setup

Depending on your choice of instruction, the learning environment setup can be face-to-face, online, or hybrid. The authors facilitated this face-to-face to support the full experience of the training program.

Face-to-Face Setting

The face-to-face setup can either be arranged for individual work in a computer lab or the seating can be placed in small groups for collaboration. Ensure all students have access to a laptop or tablet. Set up a demonstration station with a projector for instructor-led tutorials. Provide a quiet space for audio recording. We recommend a separate room for audio recording so students can finish their audio narrations. Allocate at least 30–45 minutes for setup before each session.

Online Setting

Use a video conferencing platform (e.g., Zoom, Google Meet) with screen-sharing and breakout room capabilities. Create breakout rooms so students can finish their audio narrations in a quiet place. Provide students with access to digital resources and AI tools before the session. Use shared Google Drive folders or a Learning Management System (LMS) for script collaboration and multimedia submission. Allocate 15–30 minutes for instructor setup before each session.

Hybrid Setting

Combine in-person collaboration with online resources and AI tools. Ensure students can access the required software both at school and at home. Use discussion boards or chat tools for asynchronous support. Allocate 30–45 minutes for setting up both physical and digital environments.

Context and Setting

The ITECH-STEM Summer Technology Camp at the University of Houston provided an ideal environment for developing this AI-enhanced digital storytelling lesson. As an informal STEM learning setting, the camp encouraged exploration, creativity, and hands-on engagement with emerging technologies. Unlike formal classrooms, which are often constrained by rigid pacing and standardized assessments, this program allowed students to engage deeply with AI tools through creative storytelling, critical thinking, and multimedia production (Dogan, 2020).

The learning population was diverse with students in grades 3–6, 47% female and 53% male, with varying levels of prior knowledge which shaped the curriculum design. Some participants had no experience with digital storytelling or AI tools, while others were more technologically fluent. To support all learners, the curriculum employed scaffolded AI integration, beginning with guided use of tools like ChatGPT, QuillBot, and NightCafe Creator, and progressing to independent content creation. This ensured that students with limited writing or artistic experience could still produce high-quality stories, while more advanced learners explored deeper creative possibilities (Dogan, 2018; Mercader & Gairín, 2020).

The Project-Based Learning (PBL) framework was central to the instructional design of the sessions. Students engaged in a structured process: researching STEM personas, drafting scripts, generating visuals, and producing videos using WeVideo. These real-world tasks gave students ownership of their learning, allowing them to make creative decisions about content and structure. Collaborative elements—including peer feedback sessions and group discussions—supported

communication skills and diverse storytelling perspectives, aligning with PBL's emphasis on active, student-centered learning (Dogan, 2018; Hung et al., 2012).

Design-Based Research (DBR) informed ongoing improvements to the learning experience. Student feedback, instructor observations, and performance data were used to refine lesson flow, tool integration, and instructional support (Abdallah & Wegerif, 2014). For example, early observations revealed that younger students struggled to personalize AI-generated content. As a result, additional scaffolds were added, including sentence starters, visual planning templates, and checklists to guide revisions and promote student voice.

The role of AI tools was carefully balanced to support—rather than replace—student creativity. Tools like ChatGPT and QuillBot helped students brainstorm and refine scripts, while the NightCafe Creator enabled them to generate custom visuals. These tools acted as creative partners, not final content providers. Instructors emphasized human-AI collaboration, encouraging students to adapt, critique, and personalize AI outputs. Visuals were often blended with hand-drawn illustrations or personal photos, reinforcing intentional design choices and maintaining student ownership (Gocen & Aydemir, 2020; Robin, 2008).

Instructor and mentor preparation was another critical design element. The camp was facilitated by 15 undergraduate mentors, many of whom were new to AI-assisted teaching. A pre-camp training program was implemented, including AI literacy modules, tool tutorials, and best practices for differentiation. This prepared mentors to support a range of student needs—from guiding writing with AI prompts to helping with multimedia integration—ensuring that all students could succeed, regardless of their starting point (Fernández-Batanero et al., 2021).

To maintain engagement in this informal setting, where participation relies on intrinsic motivation, the curriculum prioritized student choice and personal relevance. Learners selected their own STEM personas, ranging from historical scientists to contemporary innovators, and decided how to frame and present their stories. This autonomy, combined with the integration of familiar tools and opportunities for creative expression, made the experience meaningful and enjoyable (Dogan & Almus, 2017; It's About Time, 2015).

The decision to embed AI in a creative storytelling context was driven by the need to introduce AI literacy in an accessible and engaging way. Traditional approaches often rely on coding or technical exercises, which can alienate less tech-inclined students. By integrating AI tools into narrative creation, students explored real-world applications of AI in a way that felt relevant and approachable (Dogan & Itani, 2024). This not only supported skill development in media and digital literacy but also encouraged ethical reflection on AI-generated content.

Learning Representation

This learning experience follows a structured workflow, where students research STEM personas, script narratives, create multimedia elements, and produce digital stories. Through the integration of AI, students develop computational thinking, problem-solving, and digital literacy skills while engaging in creative self-expression across several different phases of instruction. *In this lesson, italic text identifies questions or prompts provided to learners.*

Tools and Functionality

Students in this learning experience used multiple tools. WeVideo was used for digital story creation, but any collaborative video creation software would work. Similarly, any large language model (LLM) chatbot and art generator would work for AI tools.

ChatGPT is an AI LLM chatbot used to help students brainstorm, develop plots, explore character creation, and overcome creative blocks through prompts and iteration. Natural language processing makes it a versatile tool for experimenting with different writing styles and genres.

QuillBot is an AI LLM chatbot that refines written narratives by suggesting improvements in grammar, structure, tone, and clarity, enhancing the overall quality of digital stories. In this learning experience, it aided in paraphrasing, summarizing, and improving coherence to ensure the story flowed smoothly.

NightCafe Creator is an AI art generator that creates custom visual art based on text prompts. This allowed students to create characters, settings, and

symbolic elements that enhanced their digital storytelling and aligned with narrative themes.

WeVideo is a cloud-based platform that enabled students to create multimedia stories by combining audio, music, visuals, and effects—providing polished and engaging content.

Phase Summaries

This experience was separated into six phases:

Phase 1

In this five-hour phase, pre-service teachers were equipped with the knowledge to guide students in AI-enhanced digital storytelling. They learned key concepts, explored AI tools, and engaged in hands-on training to support students' creative projects.

Phase 2

In this 45–60-minute phase, STEM camp students were introduced to the fundamentals of digital storytelling and its connection to STEM learning. They explored key concepts, drew inspiration from previous projects, and began selecting a STEM persona for their own storytelling projects.

Phase 3

In this 90-minute phase, students developed a structured script for their digital story with AI tool assistance. They brainstormed ideas, drafted their narrative, and refined their script using AI-powered tools to enhance clarity, coherence, and grammar.

Phase 4

In this 2-3-hour phase, students developed multimedia skills by creating and editing their digital stories. They used video editing tools, generated AI-enhanced visuals, and incorporated voice narration and music to assemble their final projects.

Phase 5

In this 90-minute phase, students refined the audio and video elements of their digital stories. They recorded and enhanced their narration, then finalized the editing by synchronizing visuals and audio to produce a polished final project.

Phase 6

In this 60-90-minute phase, students presented their final digital stories, reflecting on their creative process and learning journey. Their projects were assessed using a detailed rubric, followed by a final reflection and feedback session to highlight strengths and areas for improvement.

Phase 1: Inquiry (5 Hours)

In Phase 1, mentors engaged in inquiry by exploring the concepts of digital storytelling, AI tools, and pedagogical applications. The focus was on equipping mentors with the foundational knowledge they needed to guide students through the PBL process, including understanding copyright compliance and how AI tools can support storytelling.

Objective

Equip pre-service teachers with the knowledge and skills to effectively guide students in AI-enhanced digital storytelling.

Activities

The instructional phase began with self-study materials and online sessions (2 hours total). This included an overview of digital storytelling concepts and benefits (Dogan & Almus, 2017), an introduction to AI tools for education and their pedagogical applications (Fernández-Batanero et al., 2021), and copyright compliance training focused on finding and citing royalty-free media.

Following this introduction, participants took part in a hands-on training workshop (2 hours total). During the workshop, they practiced using WeVideo for digital story creation, engaged in AI-assisted script writing with ChatGPT and QuillBot, and explored NightCafe Creator and Canva AI Image Generator for AI-powered visuals.

Assessment of Teacher Readiness

Phase 1 concluded with the completion of an AI-enhanced digital story prototype using the provided tools. Mentors then participated in a peer review and feedback session (1 hour), evaluating one another's sample projects using the provided rubric that would later be used to assess the students' digital stories.

Phase 2: Ideation (45-60 Minutes)

In Phase 2, a 45–60-minute phase, the 3-6 grade students began the ideation process by exploring digital storytelling fundamentals and brainstorming ideas for their own projects. By selecting a STEM persona and researching their achievements, students initiated the creative process, making connections between storytelling and STEM learning.

Objective

Introduce students to digital storytelling fundamentals and its connection to STEM learning.

Activities

This phase began with an overview of digital storytelling (10 minutes). During this time, students were introduced to the definition, purpose, and impact of digital storytelling (Dogan, 2014). They viewed sample digital stories from the DISTCO Gallery (Dogan, 2020) and examined the key elements of effective digital stories, including originality, visuals, emotional music, and narration. Next, students were given a project overview and inspiration session (15 minutes). This included a presentation of previous student digital stories and an explanation of the project expectations.

Following that, students began selecting a STEM persona (20–30 minutes). They chose an influential scientist, engineer, or innovator from a pre-curated list and conducted brief research on the chosen persona's achievements, contributions, and significance. As part of this activity, students answered the following questions: *What did this person accomplish? Why do you find them inspiring? How does their work relate to your interests?*

Phase 3: Iteration (90 Minutes)

Phase 3 allowed students to iterate on their initial ideas by developing a structured script in 90 minutes. Using AI tools like ChatGPT and QuillBot, students refined and revised their narratives, ensuring logical flow and clarity. This iterative process helped them move from initial brainstorming to a more refined, structured script.

Objective

Guide students through the process of developing a structured script with AI assistance.

Activities

Phase 3 began with brainstorming and research (30 minutes), during which students used ChatGPT for idea generation related to their selected STEM persona. They documented the AI-generated responses and worked with mentor support to refine and develop their ideas further. Following the brainstorming session, students moved on to drafting the script (30 minutes). They created their narrative structure in Google Docs, which included two main parts: the STEM persona's achievements and significance, and the student's personal connection and aspirations.

The final part of this phase involved script refinement using AI tools (30 minutes). During this time, students used QuillBot to enhance the clarity, grammar, and coherence of their written narratives.

Assessment

At the end of this phase, students submitted a first-draft script that included AI-assisted revisions. Instructors then provided feedback focusing on clarity, structure, and logical sequencing.

Phase 4: Iteration (2-3 Hours)

In this 2–3-hour Phase 4, students continued the iteration process as they developed multimedia elements for their digital story. They used WeVideo and AI-generated visuals to build and refine their project, experimenting with different edits, visuals, and sounds to create a polished final product.

Objective

Develop multimedia storytelling skills through video editing, AI-enhanced imagery, and voice narration.

Activities

This phase began with an introduction to WeVideo (30 minutes), during which students participated in a hands-on tutorial covering timelines, inserting media, and adding transitions. They then practiced assembling sample clips to become familiar with the platform. Next, students moved on to generating AI-enhanced visuals (30–45 minutes). Using NightCafe Creator and Canva AI Image Generator, they created a range of images and selected 20 AI-generated visuals to include in their digital story.

The final segment of this phase focused on editing and assembly (60 minutes). Students imported their selected images and videos into WeVideo, adjusted display times, added transitions and animations, and layered background music from WeVideo's royalty-free library to enhance the storytelling experience.

Phase 5: Finalization (90 Minutes)

This 90-minute Phase 5 allowed students to finalize their digital stories through the iterative process of voice recording and editing. They refined the synchronization of their visuals and narration, adjusting their work to achieve the desired emotional and narrative impact. This final round of iteration ensured the story was coherent and professionally presented.

Objective

Ensure students refine audio and video elements for final production.

Activities

This phase began with voice recording (30 minutes). Students recorded their narration in a quiet space using WeVideo, and Adobe Podcast AI Enhance Speech was used to improve the clarity of the audio as needed. Following the recording session, students proceeded to final editing (60 minutes). During this time, they synchronized visuals with narration, adjusted volume levels, trimmed clips, and finalized transitions to complete their digital stories.

Assessment

After submitting their digital stories, students were evaluated using the DISTCO rubric's criteria on 'voice consistency,' which includes voice quality being clear and audible throughout the presentation and 'voice pacing,' which refers to rhythm and voice punctuation that fits the storyline.

Phase 6: Reflection (60-90 Minutes)

In Phase 6, 60-90-minutes, students engaged in reflection by presenting their completed digital stories to peers and mentors. They assessed their own work and received feedback from others, using structured evaluation criteria. The final reflection allowed students to evaluate their learning journey, and the effectiveness of the AI tools used in their projects.

Objective

Evaluate student work and celebrate creativity and learning achievements.

Activities

This phase began with a classroom viewing session (30–45 minutes), during which students presented their final projects to peers and mentors. They discussed the challenges they encountered, the inspirations behind their stories, and their key takeaways from the experience.

Following the presentations, students' projects were assessed using the DISTCO rubric (30–45 minutes). The evaluation criteria, as outlined by Dogan and Almus (2017), included purpose and originality, script coherence and creativity, image and video quality and relevance, voice consistency, pacing, emotional engagement, and copyright compliance.

Final Reflection & Feedback

To conclude the phase, students wrote self-reflections on their learning experience and their use of AI tools throughout the project. Mentors then

provided structured feedback, highlighting each student's strengths and identifying areas for growth.

Assessment Alignment

This learning scenario incorporates the DISTCO rubric as an evaluation tool. Below is how the DISTCO rubric categories were utilized in this lesson.

Digital Literacy Objectives and Evaluation

During Phases 2, 3, 4, and 5, students engaged in various digital literacy practices and digital literacy objectives were assessed using specific categories from the DISTCO rubric. The *Purpose & Script/Story* category addressed the learning objective that students will create coherent digital narratives with a clear purpose (Phase 3). Evaluation in this category focused on the students' ability to establish and maintain focus while developing original content.

The *Economy & Duration* category supported the objective that students will demonstrate effective digital communication through concise storytelling (Phases 2, 3, 4, and 5). This was evaluated by measuring the students' ability to create appropriately timed content that conveyed the necessary level of detail.

Finally, the *Image Relevancy & Quality* category aligned with the objective that students will select and create visual media that enhances the meaning of their narratives (Phases 4 and 5). Evaluation focused on students' visual literacy, specifically their ability to match images to the narrative content and to create an appropriate atmosphere that supports the story.

AI Fluency Objectives and Evaluation

During Phases 2, 3, 4, and 5, students engaged in various AI fluency practices which were also evaluated using the DISTCO rubric. The *Creativity & Script/Story* category addressed the objective that students will use AI tools to enhance rather than replace creative thinking (Phases 3 and 4). This was evaluated by assessing the originality and creative contributions evident in their work, even when AI-assisted production was involved.

The *Voice & Language* category aligned with the objective that students refined AI-generated content to maintain an authentic voice (Phases 3, 4, and 5). Evaluation in this category focused on the students' ability to adapt and personalize AI outputs, considering aspects such as voice quality, pacing, and the appropriateness of language.

Finally, the *Copyright Issues* category addressed the objective that students navigate ethical considerations when using AI-generated content (Phases 2, 3, 4, and 5). This was evaluated by assessing their understanding of attribution and their ability to use AI tools ethically and responsibly.

Critical Reflection

The AI-enhanced digital storytelling learning representation was implemented during the ITECH-STEM Summer Technology Camp at the University of Houston, using a Design-Based Research (DBR) approach for iterative improvements in the summers of 2023 and 2024 (Hall, 2020). The program successfully introduced students to AI-assisted storytelling, with most achieving the intended outcomes of enhanced logical thinking, digital literacy, and creativity. Evidence from student reflections, mentor observations, and final projects indicated growth in narrative structure, tool usage, and multimedia integration.

A key challenge involved selecting the right AI tools. Some image-generation platforms had limited free credits, restricting experimentation. This was resolved by switching to NightCafe Creator, which offered more accessible features. ChatGPT was introduced to support brainstorming and scriptwriting, but many students used it primarily as a search tool rather than for drafting narratives. For example, students often generated lists of STEM topics but didn't use the tool to develop dialogue or scene structure. This revealed a need for more explicit instruction on how AI can be used throughout the storytelling process.

To address this, future iterations will include structured guidelines showing how AI tools can support brainstorming, outlining, drafting, and revision. These models will emphasize that AI should support, not replace, original storytelling. Mentor supervision was also critical for guiding ethical use. Students submitted AI-generated content for review

before integrating it into their stories. Mentors encouraged students to revise and personalize content, ensuring their voice remained central. In one case, a student who relied entirely on AI for a script was guided to restructure and infuse the story with personal elements, reinforcing ownership.

Student engagement was notably high when learners selected their own STEM personas. This autonomy led to greater investment, as students chose figures they admired—such as medical researchers, astronauts, or engineers. This choice-based approach supported differentiated learning and aligned with the project's goal of promoting creativity and relevance. To further build engagement, future implementations will expand peer review activities to support collaboration, feedback, and iteration.

Based on these insights, several enhancements are planned. A structured AI-use framework will be introduced, outlining best practices and examples for using tools like ChatGPT and NightCafe Creator across different storytelling phases. Pre-project training will be provided to both mentors and students, focusing on ethical AI use, effective prompting, and creative adaptation. Exploring additional AI tools will also increase flexibility and mitigate limitations from tool access or availability.

Overall, the program demonstrated that AI-enhanced digital storytelling can foster key 21st-century skills, including creativity, computational thinking, and digital literacy. With scaffolded instruction, student autonomy, and guided mentorship, students balanced AI support with authentic content creation, resulting in meaningful learning experiences. Future research could examine how AI integration affects not just the efficiency of storytelling, but also its impact on student creativity, engagement, and narrative quality. Comparing traditional and AI-supported storytelling methods may offer valuable insights into how emerging technologies influence learning outcomes in informal STEM education.

References

Abdallah, M. M. S. & Wegerif, R. B. (2014). Design-based research (DBR) in educational enquiry and technological studies: A version for PhD students targeting the integration of new technologies and literacies into educational contexts. *Eric,* Article ED546471. https://files.eric.ed.gov/fulltext/ED546471.pdf

Digital Storytelling Content (2018). *DISTCO rubric*. Retrieved June 2, 2025, from https://distco.org/documents/

Dogan, B. (2014). Educational uses of digital storytelling in K-12: Research results of a digital storytelling contest (DISTCO) 2013. In M. Searson & M. N. Ochoa (Eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2014* (pp. 520–529). Association for the Advancement of Computing in Education. http://www.editlib.org/p/130802

Dogan, B. (2018). Project based learning (PBL) with digital storytelling approach: Research results of digital storytelling contest (DISTCO) PBL 2017. In E. Langran & J. Borup (Eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2018* (pp. 293–304). Association for the Advancement of Computing in Education. https://www.learntechlib.org/p/182539

Dogan, B. (2020). A summer technology camp's impact on elementary students' stem attitudes, spatial thinking, 21st century skills, and growth mindset: The innovative technology challenges for science, technology, engineering, and mathematics (ITECH-STEM). In D. Schmidt-Crawford (Ed.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2020* (pp. 1733–1741). Association for the Advancement of Computing in Education. https://www.learntechlib.org/p/216051

Dogan, B., & Almus, K. (2017). Research results of digital storytelling contest (DISTCO) 2016. In *Society for Information Technology & Teacher Education International Conference 2017* (pp. 241–252). Association for the Advancement of Computing in Education. https://www.learntechlib.org/p/177307

Dogan, B., & Itani, A. (2024). Teaching and developing digital stories through artificial intelligence in a summer STEM camp designed for elementary-aged students. In J. Cohen & G. Solano (Eds.), Proceedings of Society for Information Technology & Teacher Education International

Conference (pp. 411-416). Association for the Advancement of Computing in Education (AACE), Las Vegas, NV, United States. https://www.learntechlib.org/primary/p/223966/

Fernández-Batanero, J. M., Román-Graván, P., Reyes-Rebollo, M. M., & Montenegro-Rueda, M. (2021). Impact of educational technology on teacher stress and anxiety: A literature review. *International Journal of Environmental Research and Public Health, 18*(2), Article 548. https://doi.org/10.3390/ijerph18020548

Gocen, A., & Aydemir, F. (2020). Artificial intelligence in education and schools. *Research on Education and Media, 12*(1), 13-21. https://doi.org/10.2478/rem-2020-0003

Hall, T. (2020). Bridging practice and theory: The emerging potential of design-based research (DBR) for digital innovation in education. *Education Research and Perspectives*, *47*, 157–173. https://www.erpjournal.net/wp-content/uploads/2021/02/07_ERPV47_Hall_1.pdf

Hung, C. M., Hwang, G. J., & Huang, I. (2012). A project-based digital storytelling approach for improving students' learning motivation, problem-solving competence and learning achievement. *Journal of Educational Technology & Society, 15*(4), 368-379.

International Society for Technology in Education. (2017). *ISTE standards: Educators*. Retrieved May 22, 2025, from https://www.iste.org/standards/iste-standards-for-teachers

It's About Time. (2015, August 20). STEM for elementary school students – How to instill a lifelong love of science. *Medium.* https://medium.com/@ItsAboutTimeEDU/stem-for-elementary-school-students-how-to-instill-a-lifelong-love-of-science-daf24d6dc3f5

Larmer, J., Mergendoller, J. R., & Boss, S. (2015). *Setting the standard for project based learning: A proven approach to rigorous classroom instruction.* ASCD.

Mercader, C., & Gairín, J. (2020). University teachers' perception of barriers to the use of digital technologies: the importance of the academic discipline. *International Journal of Educational Technology in Higher Education, 17,* Article 4. https://doi.org/10.1186/s41239-020-0182-x

About the Authors

Dr. Bulent Dogan is a Clinical Associate Professor in the Department of Curriculum and Instruction at the University of Houston (UH), specializing in Learning, Design, and Technology. With a B.S. in Electrical and Computer Engineering and an Ed.D. in Curriculum and Instruction (Instructional Technology emphasis) from UH, Dr. Dogan has been recognized with multiple awards, including the university-wide "Teaching Excellence Award for Innovation in Instructional Technology" (2024) and the college-wide "Teaching Excellence Award" (2018). As founder and director of ITECH-STEM at UH, he oversees STEM-focused programs, including summer camps and initiatives like Girls Coding Academy. His research interests encompass STEM education, digital storytelling, AI, gamification, virtual reality, coding, and 3D printing.

Amani Itani is a PhD student at the University of Houston, studying Curriculum and Instruction with a focus on Learning Design and Technology. Amani's research journey is centered around the intersection of education and technology, with a particular focus on integrating Artificial Intelligence into educational contexts (AIEd) such as digital storytelling projects, digital educational escape rooms, and ethics in AIEd. Her explorations also encompass curriculum planning methodologies and the cultivation of 21st-century skills within modern educational landscapes.

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9349

Integrating CT/CS into a Teacher Education Program: A Year-Long PD

Irene A. Bal, Karen Terrell, Hoang Bui, and Stacy Williams, Loyola University Maryland

Overview

This grant-funded, year-long professional development (PD) for higher education faculty focused on integrating computational thinking (CT) and computer science (CS) in preservice teacher education courses. A full-day PD was followed-up by four 15-minute PD sessions held during the Teacher Education Department (TED) meetings, supporting all TED faculty. A PD spinoff initiative, "Focused Faculty," supported five TED faculty in planning and implementing a CT/CS lesson in one of their courses in Spring 2024. Multiple materials and activities were utilized in this PD including Scratch Jr., Scratch, Code.org, micro:bits, unplugged activities, machine learning, makerspaces, and literacy integration.

Topics: Computational Thinking, Computer Science, Faculty, Preservice Teachers, Professional Development

Time: Faculty PD: Five sessions totaling 7 hours over the year (one six-hour session and four 15-minute follow-up sessions). Focused Faculty: Approximately 13 hours over a semester

Setup

The PD sessions were a mix of face-to-face and online which facilitated different learning environments and materials. For the face-to-face sessions, ensure there is a room large enough for all faculty/attendees. Most of the rooms utilized in this PD included shared tables that were either circle or rectangle to allow faculty communication and collaboration. A specific setup is provided for each PD session. Although specific tools/materials are showcased, this PD can be facilitated with any CT/CS tools and activities available.

Context-at-a-Glance

Setting
Professional development (PD) for a teacher education department in an urban, private institute of higher education in Maryland, U.S.

Modality
Face-to-face with two online sessions

PD Structure
This was a grant-funded, year-long PD focused on computational thinking (CT) and computer science (CS) integration into preservice education courses. A total of five sessions occurred with one full-day session followed by four, 15-minute sessions.

Organizational Norms
Faculty were paid a stipend for the full-day session. The follow-up PDs occurred during the Teacher Education Department (TED) meetings which faculty were expected to attend.

Learner Characteristics
This PD supported TED faculty with a range of prior CT/CS experiences and knowledge.

Instructor Characteristics
Three education faculty and one CS faculty with various prior CT/CS experience and knowledge.

Development Rationale
This PD was developed to support faculty in integrating CT/CS in their courses, supporting future PK-12 teachers in the integration of CT/CS. This work supports the Maryland General Assembly (2020) bill *Securing the Future: Computer Science Education for All*.

Design Framework
Social Constructivist Design Theory; Adult Learning Theory

Materials

- Session 1:
  - Session 1 PD slides
  - iPads/personal mobile devices
  - Computers
  - Scratch Jr.
  - Unplugged activities
  - Scratch accounts for participants
  - Code.org accounts for participants
  - micro:bits
  - Read aloud book (e.g., Brown Bear)
  - Exit Ticket
- Session 2:
  - Session 2 PD slides
  - *Cats, Dogs & Machine Learning* Lesson (MIT Media Lab, n.d.)
- Session 3:
  - Physical makerspace materials (e.g., Legos, popsicle sticks, cups/containers, straws, string, tape, playdoh)
- Session 4:
  - Poem Art lesson by CODE (n.d.-d)

Standards

The Computer Science Teachers Association ([CSTA], 2020) standards for computer science teachers and the International Society for Technology in Education ([ISTE], 2019) computational thinking competences were utilized in the design and implementation of this PD.

CSTA Standards for Computer Science Teachers

- Standard 1: CS Knowledge and Skills
- Standard 2: Equity and Inclusion
- Standard 3: Professional Growth and Identity
- Standard 4: Instructional Design
- Standard 5: Classroom Practice

ISTE Computational Thinking Competencies

- Standard 1: Computational Thinking (Learner)
- Standard 2: Equity Leader (Leader)
- Standard 3: Collaborating Around Computing (Collaborator)
- Standard 4: Creativity & Design (Designer)
- Standard 5: Integrating Computational Thinking (Facilitator)

Context and Setting

In 2018, the Maryland General Assembly (2020) passed a bill entitled, *Securing the Future: Computer Science Education for All*. This legislation established the Maryland Center for Computing Education (MCCE), with the mission "to identify ways to expand access to high-quality computer science education, strengthen the skills of educators, and increase the number of computer science teachers" (para. 5). Concurrently, the Maryland State Department of Education (2018) approved new computer science standards for grades K-12, requiring the creation of more opportunities for students to acquire CS-related skills in all grades.

MCCE holds professional development throughout the year for both PK-12 local school systems (LSS) and institutes of higher education (IHE) including workshops, courses, micro-credentials, the Maryland Computing Education Summit, and an annual Preservice CS Education Intensive Workshop. These professional learning sessions are a mix of online (i.e., Zoom) and face-to-face and include state-wide representatives from PK-12 LSS and IHEs. The Maryland Higher Education Summer CS Intensive Workshop is a three-day annual event with various state IHEs designed to equip attendees with CT/CS concepts and skills to implement in their teacher-preparation courses and programs. In Summer 2023, the authors, representing three departments from one IHE (Teacher Education, Education Specialties, and Computer Science), attended the workshop with the support of a small grant. During the conference, the group decided to move beyond individual course integration and, instead, created a plan, which included writing a second grant, to support TED faculty in the learning and planning of CT/CS and its full integration throughout the Teacher Education Program.

The Teacher Education Program is situated in a School of Education in the Teacher Education Department (TED) which encompasses programs for PK-12 teacher licensure at the undergraduate and graduate levels. In 2023-2024, the TED department included 12 faculty who had a range of prior CT and

CS knowledge. Most faculty were new to CT/CS and thought CT/CS primarily aligned with math concepts.

In the 2023-2024 academic year, it was the authors' goal to engage the TED faculty in approximately 8 hours of CT/CS PD and integrate CT/CS in some of the Teacher Education Program courses for a 2024-2025 implementation.

Instructor Backgrounds

Four faculty collaborated on this year-long PD representing three different departments in the same IHE: (a) the Teacher Education Department, (b) the Education Specialties Department, and (c) the Computer Science Department. Each of their CT/CS background knowledge and learning during the 2023 Maryland Higher Education Summer CS Intensive Conference are presented to demonstrate their diverse knowledge, skills, and perspectives when designing and implementing this PD.

Teacher Education Department: I am currently the Teacher Education Department Chair and have been a full-time clinical faculty member in an educator preparation program for over 16 years. I hold a teaching license in biology, grades 7 - 12. I taught general methods, elementary and secondary science methods, and supervised teacher candidates in all subjects and all levels. I also serve as the coordinator of partnerships and clinical experiences, so I work closely with local schools/school systems to ensure our programs are aligned to state and local initiatives.

The 2023 Maryland Higher Education Summer CS Intensive Conference provided excellent information related to state initiatives and gave me practical ideas to prepare preservice teachers with the CT/CS skills they will need. I took away conceptual ideas for our overall curriculum and practical activities to implement into coursework. The variety of plugged and unplugged strategies that were modeled allowed me to see the ways CT/CS can be integrated at all grade levels, all ability levels, and in all school settings. I was able to relate CT concepts to concepts we are already teaching throughout our programs, and the conference sessions helped me to consider ways faculty can make explicit connections among different content areas and CT concepts. Finally, the conference provided time for our team to work together to develop a plan of action for our programs, including developing a full-day workshop with key follow-up points over the year.

Teacher Education Department: I am in my second year in the School of Education as an Assistant Professor of Mathematics Education. I have been in education for twenty-five years, beginning as a high school teacher and moving into middle school, later coaching and consulting. I attended the 2023 Maryland Higher Education Summer CS Intensive Conference as I was starting my tenure, and I was pleasantly surprised that CT/CS initiatives were so prevalent. I was already intending to incorporate technology such as the Desmos' Activity Builder portal in my courses; however, the MCCE workshop provided a theoretical foundation for integration, as well as additional technologies to consider. Through the workshop I encountered micro:bits and block coding – technology that I had previously heard about but had never experienced. I have added both to my courses through activities such as coding the micro:bits as environmental sensors and as dice via Microsoft MakeCode. I have also been able to incorporate various activities wherein I have students create operations games through Code.org's game lab.

Another technology that I have been able to incorporate in my courses has been Texas Instruments' NSpire. I was familiar with their 83s and 84s from my days as a K-12 teacher, but the workshop introduced me to the expanded capabilities of the NSpire which includes Python coding. This exposure had long-term effects, establishing an ongoing partnership between my department and Texas Instruments (TI) that has expanded into our school serving as a pilot site for the company's Rising Teachers induction-years mentoring program for our graduating seniors, as well as multiple grant collaborations. Because of the MCCE workshop, students are gaining exposure to technology that goes far beyond computations on calculators and offers authentic uses that enhance pupil engagement in mathematics and other content areas.

Education Specialties Department: I am a certified PK-12 music teacher who currently teaches learning design, educational technology, research, and music at an IHE. Since 2018, I participated in MCCE's regular state-wide meetings for higher education and PK-12 institutions, attended their annual Preservice CS Education Intensive Workshop from 2022-2024, and was the principal investigator on four grants from MCCE.

In the 2023 Preservice CS Education Intensive Workshop, I was excited to have a team from my institution attend as I had been the only person from my institution engaged in MCCE for many years. During this workshop, I led the team on integrating CT/CS in their courses and the Teacher Education Program. After the workshop, I continued leading the team through the second grant and leading the plan to integrate CS into the Teacher Education Program at our institution.

Computer Science Department: Before attending the 2023 Preservice CS Education Intensive Workshop, I taught computer science for nearly nine years but could hardly recall encountering the concept of computational thinking. During the conference, I was introduced to CT and realized that I had been practicing it not only in my teaching but also outside the classroom. I came to understand that CT is not merely about using computers to solve problems; it represents a broader problem-solving approach rooted in computer science concepts such as divide and conquer, pattern recognition, abstraction, and algorithms. Additionally, I learned about the benefits of exposing young children to CT through unplugged activities, which can enhance critical thinking and analytical skills. This early exposure can also help children apply CT to solve everyday problems systematically and creatively. Since this workshop, I am continuing to collaborate with these authors in multiple grants, CT course design, and PD sessions.

CT/CS Four-Year Plan

During the summer of 2023, the faculty who attended the 2023 Maryland Higher Education Summer CS Intensive Conference developed a four-year plan to incorporate CT/CS as a permanent aspect in the Teacher Education Program. Figure 1 outlines the gradual rollout of this plan, beginning with department-level PD and small-scale course pilots, building into whole-scale CT/CS incorporation across the program, and formal assessment of preservice teachers' CT/CS planning and instruction.

To aid with CT/CS instruction, TED used grant funding from MCCE to purchase a class set (N = 20) of micro:bits (n.d.-c), as well as a Cubetto (PRIMO, n.d.), six Code & Go Mice (Learning Resources, n.d.), and some unplugged resources (e.g., Legos). The department invested in a set (N = 10) of TI Nspire calculators (Texas Instruments, n.d.). The activities that were developed and implemented are described further in the Learning Representation section.

As the team developed this multi-year outline, their planning and thinking were guided by constructivist and democratic lenses of instruction. In describing his views about democratic education, Dewey (1930) describes the lack of variety and choice in educative experiences as a kind of slavery, wherein "men are engaged in activity, which is socially serviceable, but whose service they do not understand and have no personal interest in" (p.98). As such, the authors were adamant in the implementation design that if faculty were going to be asked to engage in CT/CS instruction in their courses, not only should they be exposed to a variety of CT/CS exercises, but they should also be given choice as to when the pilot of these activities should occur in their classes, and which task(s) to incorporate. Further, Piaget's (1952) notion of learning (i.e., constructivism) dictated the fundamental interconnectedness of the mental life (i.e., assimilation) with the experiential (i.e., accommodation), "experience, accordingly, is not reception but progressive action and construction…" (p.365) and that "presupposes an organizing or structuring activity" (p.369). As such, the team was intentional in its selection of tasks to present to the faculty – ones that built upon each professor's content knowledge and used CT/CS activities to expand their conceptions and instruction of various topics.

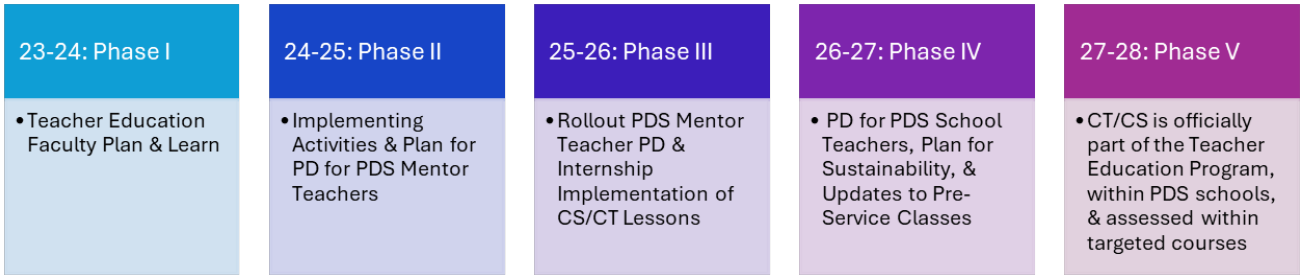


Figure 1. Original TED CT/CS training plan from 2023.

This article details the 2023-2024 Phase 1 plan to support TED faculty in the Learning Representation (Phase 1) section. The plan supported TED faculty in the learning, planning, and implementation of CT/CS lessons in their preservice courses. This was completed through a full-day PD, four 15-minute follow-up PDs, and a spinoff initiative called "Focused Faculty" where five faculty were paid to plan and implement CT/CS lessons in a Spring 2024 course.

Learning Representation (Phase 1)

The year-long plan to support TED faculty in the planning and learning of CT/CS began with a full-day PD session for all TED faculty. During this full-day PD session, faculty identified if they were interested and able to integrate CT/CS in one of their courses with planning and a pilot/attempt at activities in Spring 2024. This was a pilot with the goal of faculty trying things out to see what worked with a full implementation occurring in the 2024-2025 academic year. Five faculty signed up to integrate CT/CS in their courses and were called "Focused Faculty."

After the full-day PD, faculty learning continued throughout the year by presenting different CT/CS integration ideas and activities in the 2023-2024 monthly TED meetings. Three activities and discussions were implemented in department meetings. The showcase of the five Focused Faculty lesson(s), reflections, and purchased materials were presented in the last department meeting of the year (see Focused Faculty section; Table 1).

During this lesson, italic text identifies questions or prompts for the learners.

TED Faculty PD

Session 1: 11/3/23 Full-Day PD

The first PD session was for all TED faculty in a one-day, six-hour session. Within the session, nine faculty members were introduced to digital and unplugged CT/CS activities, CT/CS standards for K-12 teachers and students, and given time to collaboratively brainstorm how CT/CS can be integrated into classes (Figure 2; Slide 2 in Session 1 PPT). Time provided for each activity is included; these can be adjusted based on prior knowledge, skills, and availability.

Table 1
Department Meeting Dates and Activities

TED Meeting Date	Format	CT/CS Integration Follow-Up Topic/Activity
Session 1: November 3, 2023 (6 hours)	Face-to-Face	An overview of CT/CS with unplugged and digital hands-on activities and discussions on integrating ideas in classes.
Session 2: November 17, 2023 (15 minutes)	Face-to-Face	Follow-up from the 11/3/23 full-day PD including Unplugged Machine Learning Activity, MD CS K-12 Student Standards, Discussion on how CT/CS currently fits in the School of Education & Teacher Education Vision statements.
Session 3: January 25, 2024 (15 minutes)	Face-to-Face	Integration of computational thinking to support the Whole-Part-Whole Instructional Framework
Session 4: February 22, 2024 (15 minutes)	Online	Integration of CT/CS to support social-emotional learning.
Session 5: April 26, 2024 (1 hour)	Online	Showcase of what the "Focused Faculty" completed for the year.

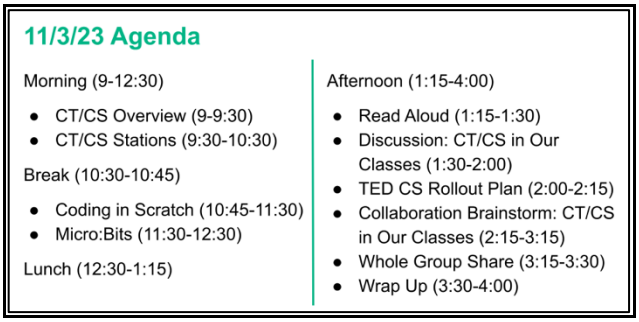


Figure 2. Agenda of full day PD for TED faculty.

Set-Up and Materials

This full-day session introduced faculty to multiple types of digital and unplugged coding and CT/CS activities. This session was facilitated in a large room with rectangle tables that sat four to six people. For the morning CT/CS stations, a second room was secured for the unplugged activities. This room was checked out for three hours to ensure the authors could set up and break down the unplugged activities. Lunch was provided to faculty (department funded) which was held in the main PD room. The authors hung posters in the main PD room before the session started showcasing the CT core practices, the Standards for CS teachers, and other printables provided by the CSTA.

The materials needed for this PD included:

- Session 1 PD Slides (PPT)
- iPads/Personal Mobile Devices
- Computers
- Scratch Jr.
- Unplugged Activities
- Scratch accounts for participants
- Code.org accounts for participants
- micro:bits
- Read aloud book (e.g., Brown Bear)
- Exit Ticket (DOC)

The iPads/Personal Mobile Devices were used for the Scratch Jr. activity, which is app based. The computers were used for the Scratch, Code.org, and micro:bit activities. The unplugged activities included (a) a maze that was laid out on the floor, (b) the Game with No Rules, a Code.org activity, and a (c) Pixels & Instagram activity. These unplugged activities are further described below.

The read aloud book for this PD was *Brown Bear, Brown Bear* (Martin, 1967). Any book that has repetition and similar phrases can be used for this activity. A read aloud was selected to model how CT/CS connects to literacy as there are multiple literacy faculty in the department.

Professional Development

CT/CS Overview (30 minutes): The full-day session began with an overview of CT/CS which started with a 30-minute explanation of Computing Education, encompassing Computational Thinking and Computer Science, the differences between CT and CS, and definitions (Digital Promise, n.d.; K–12 Computer Science Framework, 2016; Wing, 2006; Slides 3-5 in Session 1 PD PPT). The CT core practices from the K–12 Computer Science Framework (2016) were then presented (Figure 3; Slide 6 in Session 1 PD PPT).

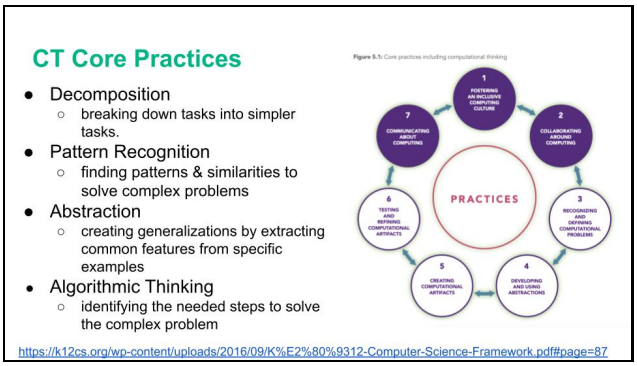


Figure 3. CT core practices from K-12 Computer Science Framework (2016), licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0).

Following this information, data and statements on CT/CS teacher preparation were presented (Slides 7-9 in Session 1 PD PPT). This information was important to present as the TED faculty are teaching future teachers and the CS teacher preparation data showcases that 24% of PK-12 CS teachers surveyed did not have college-level CS classes (Koshy et al., 2022). After this data, the Standards for CS teachers were presented with a focus on "effective K-12 CS instruction" (CSTA, 2020, para. 1; Figure 4; Slide 10 in Session 1 PD PPT).

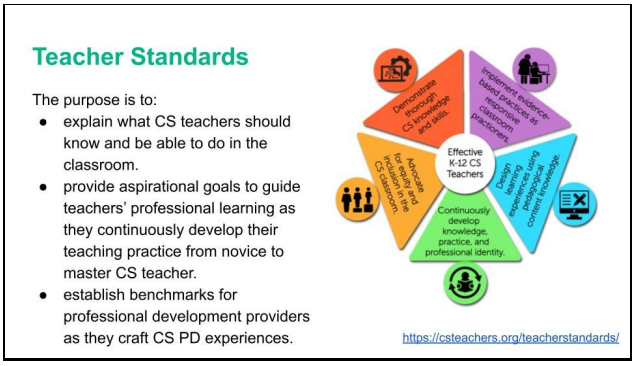


Figure 4. Slide presentation of standards for CS teachers, licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0).

CT/CS Stations (1 hour): After the overview of CT/CS, three CT/CS stations were implemented (Figure 5;

Slide 11 in Session 1 PD PPT). The stations were developed to be an overview of different CT/CS options and tools available to faculty with the end goal of getting faculty interested in joining the Focused Faculty initiative and help those faculty determine the CT/CS activities/tools they could integrate in their classes. Two of these stations were digital CT/CS stations using iPads/personal devices and computers and one of the stations was unplugged with a few activities for faculty to choose from. These CT/CS stations were set up before the full-day session began. The two digital stations were in the main room with the full-day session and the unplugged station was in a separate room to allow for multiple floor and table-top activities.

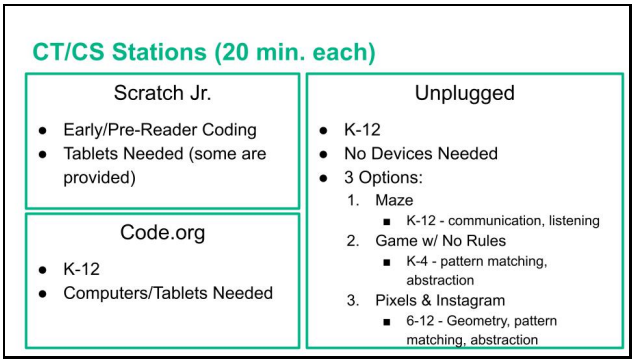


Figure 5. CT/CS stations during the full-day PD.

Participants broke into three different groups and cycled through stations for 20 minutes each. These stations were led by the authors. The stations included:

1. Scratch Jr. on iPads (or personal devices)
2. Code.org on computers
3. Unplugged activities

Scratch Jr. (n.d.) is a tool that can be used to interactively introduce young learners (PK-2) to coding concepts. The Scratch Jr. activity was geared towards early/pre-reader coding and was an exploratory activity where faculty did basic block coding to make a character move through a background and interact with other characters. iPads were available for faculty to use but faculty were also able to download the application on their personal devices if they preferred.

Code.org is a platform that provides self-guided lessons across PK-12 grade levels (CODE, n.d.-a). The activity used was Code with Anna and Elsa lesson (CODE, n.d.-b) focusing on the integration of math and coding for upper elementary/early secondary learners. Faculty used their own computers and joined a Code.org classroom section, created by one of the authors, to engage with the activity while also demonstrating the student and teacher side of Code.org for future implementation.

Unplugged activities are non-digital CT and CS activities that can be used in classrooms with and without devices. These types of activities can include games, puzzles, and other problem-solving activities that use physical tools such as crayons/markers, dice, and mazes to support CT/CS concepts like pattern recognition, abstraction, and algorithms (CS Unplugged, n.d.). Unplugged activities are typically utilized to first introduce CT/CS concepts without coding and many educators scaffold CT/CS lessons with unplugged activities first, followed by digital CT/CS work (Sawyer, 2022). Three options were provided for the faculty to choose from:

1. A maze that was laid out on the floor with directional placards for faculty to code their way through a maze. The directional placards were available in arrows for pre/early readers and in text directions (Figure 6).
2. The Game with No Rules is a Code.org activity where learners have to determine how to play a game by deconstructing provided phrases with pattern matching and abstraction, determine what the algorithm is, and create their own phrases based on their analysis (CODE, n.d.-e). Dice, markers, pens/pencils, and the printed worksheets were available for the faculty to use.
3. The Pixels & Instagram activity focuses on how images are displayed on digital devices. This activity begins with a Code.org (2015) video, Images, Pixels and RGB which features how images work on digital devices with the example of Instagram. Then, using the Colour by Numbers worksheet (CS Unplugged, 2021) faculty first create images based on the provided codes on the "Worksheet Activity: Kid Fax" page of the CS Unplugged (2021) Colour by Numbers worksheet (Figure 7). Then faculty create their own images and write the codes. At the end of the activity, faculty are provided a blank gridded page to rewrite the codes to their own images and swap with one another. They then attempt to create each other's images, based on provided codes.

After the stations, a short break was given to faculty and then a whole group lesson using Scratch and micro:bits was completed.

Figure 6. Faculty organize the maze directional placards.

Figure 7. Faculty creating images based on the provided codes in the Colour by Numbers activity.

Coding in Scratch (45 minutes): An introduction to Scratch was the next activity the faculty engaged in as a whole group (Slide 12 in Session 1 PD PPT). Scratch is "the world's largest coding community for children and a coding language with a simple visual interface that allows young people to create digital stories, games, and animations" (Scratch, n.d., para. 1). The faculty engaged with the Create a Story activity in Scratch where they had to create a short story with a beginning, middle, and end which they then created in Scratch (Scratch in Practice, n.d.). Although there is a tutorial available, one of the authors guided the faculty through the brainstorming and development of the story while the other authors supported faculty individually as needed. Forty-five minutes was provided for this activity and most faculty were able to brainstorm and create a short story in Scratch in that time. Here are a few of the faculty-created Scratch story projects:

- Kristina's Dream: https://scratch.mit.edu/projects/918668441
- Obi at the beach: https://scratch.mit.edu/projects/918667925
- Play Fetch with Beau: https://scratch.mit.edu/projects/918667967

micro:bits (1 hour): The Scratch Jr., Code.org, and Scratch activities allowed faculty to engage in coding in a scaffolded manner, slowly easing them into more advanced coding and more choices in what they code/create. The final digital coding activity for the day was coding in micro:bits (Slides 13-15 in Session 1 PD PPT). micro:bits are physical programming devices that allow learners to digitally code a machine that they can then interact with in a hands-on manner (micro:bit, n.d.-c; Figure 8). In a one-hour time slot, faculty engaged in two micro:bit activities. The first activity was a warm-up of creating dice. The dice micro:bit activity is a beginner-level activity that allows you to create your micro:bit into a die using the micro:bit block coding (this activity can also be completed in Python). Although there is a Dice Lesson Tutorial on how to teach/code the micro:bit (micro:bit, n.d.-a), one of the authors led the activity while the other authors supported faculty individually as needed. This dice activity allowed faculty to block code on the micro:bit site (i.e., learning another interface) and interact with the micro:bit before being introduced to more complex coding and options.

The second activity was directly aligned to a science lesson where faculty coded their micro:bits to measure the temperature, sound, and light for different locations around campus. The Environmental Exploration activity (micro:bit, n.d.-b)

includes a tutorial on how to complete the activity, including a data recording sheet, which allows easy implementation for teachers/faculty. This activity was led by the same author who led the micro:bit dice activity with the other authors supporting faculty as needed. Once faculty had their micro:bits programmed for the Environmental Exploration, they used the data recording sheet and worked in small groups to measure the temperature, sound, and light in various places on campus. The faculty were given 15 minutes to measure with their micro:bits and upon returning to the room, a brief discussion occurred on what they found and how the micro:bits could be used in their courses. Lunch followed this discussion.

Figure 8. Faculty using micro:bits during the full-day PD.

Read Aloud (15 minutes): The afternoon started with a read aloud activity with the book *Brown Bear, Brown Bear* (Martin, 1967), modeling how CT/CS can be integrated in all subjects, including early literacy read aloud activities (Slide 16 in Session 1 PD PPT). This activity was completed as a whole group as people were still finishing lunch. In this activity, the book *Brown Bear, Brown Bear* was read to the faculty by one of the authors. Then, as a whole group, the faculty analyzed the story through the CT core practices. The CT terms were presented in English and Spanish to showcase how multilingual learners can be supported in CT/CS activities. First, the faculty decomposed the story (in Spanish, la decomposición), answering the question: *How many animals/people were in the story?* They then identified the patterns (los patrones) by answer the following prompts:

1. *What repeated in the story?*
2. *Let's make a prediction.*
   A. *What if the brown bear saw a Teal Goat?*
   B. *What would the story say?*
   C. *What would the teal goat see?*

Then, the faculty abstracted (las abstracciones) the story, by answering the questions:

1. *What are the important parts of the story?*
2. *What lines/details are not needed?*

Finally, ending with the algorithm (el algoritmo), the recreation of the sequence of events, by answering the prompt, *what is the order of the things that occur in this story?*

Finally, in an unplugged coding activity, the faculty block coded a picture of a bear to each of the animals in order as presented in the book. This was completed with a big piece of chart paper with images of the different animals (e.g., yellow duck, blue horse) and block codes with arrows, best for early/pre-readers. This activity could be made for early/novice readers by choosing a more advanced/aged appropriate book and using text block codes instead of arrows.

CT/CS Discussion (1.5 hours): Following the *Brown Bear* activity, the faculty discussed how to integrate CT/CS in the preservice courses (Slide 17 in Session 1 PD PPT). Faculty were divided into small groups and used poster paper to showcase their ideas. These ideas were then reviewed and the potential need for funding to implement the integration ideas was indicated (Figure 9). This integration idea and identification of funding activity helped the authors in identifying how teacher education faculty were thinking of integrating CT/CS, where more PD support was needed, and where funding supports may be needed (e.g., time for planning, materials/technical tools, more professional learning, etc.). This activity helped the authors in writing the next set of grants to fund their multi-year plan.

Whole Group Share (15 minutes): After this activity, the day was concluded with a whole group share of what faculty learned and were excited about (Slide 18 in Session 1 PD PPT). This whole group share lasted about 15 minutes which was followed by some next

steps from the authors. These next steps included information on the Focused Faculty initiative, where five faculty could sign up to integrate CT/CS in their courses in the spring semester. Following that information a 3-2-1 exit ticket was given to faculty so they could reflect on three things they learned, two things they were excited to try, and one question they still had (Slide 19 in Session 1 PD PPT). In the 3-2-1 exit ticket, faculty were also asked if they were interested in participating in the Focused Faculty pilot. These 3-2-1 exit tickets were used by the authors to determine future PD topics in the upcoming TED meetings and who was interested in the Focused Faculty pilot.

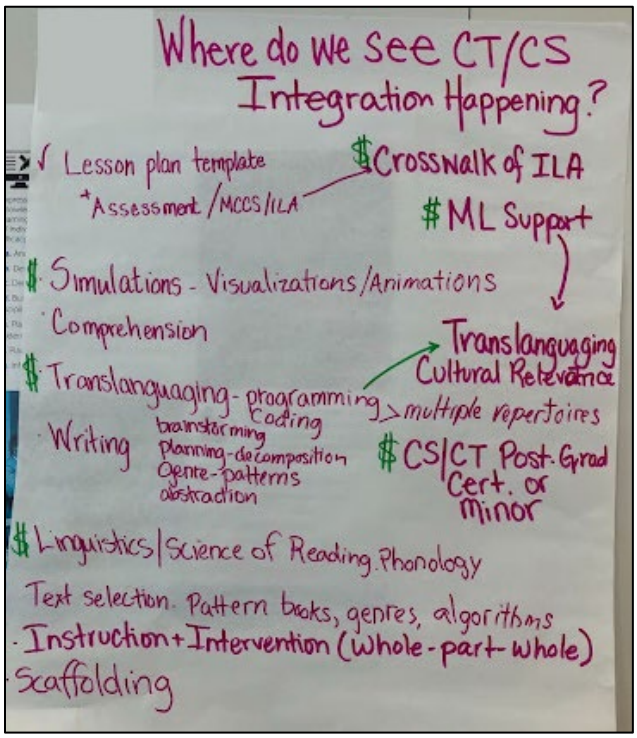


Figure 9. Faculty integration ideas from full-day PD analyzed for funding needs.

Session 2: 11/17/23 Follow-Up PD

On November 17, 2023, a follow-up meeting was held during the TED meeting. The authors were given 15-20 minutes to facilitate one activity, answer some questions from the Session 1 Exit Tickets, and announce the Focused Faculty.

Set-Up and Materials

This session was facilitated face-to-face in a room with round tables that seated six to eight people for communication and collaboration. The materials needed for this PD were:

- Session 2 PD Slides (PPT)
- MIT Media Lab (n.d.) *Cats, Dogs & Machine Learning* Lesson (adapted to be unplugged)

Professional Development

The 15-minute activity was an unplugged activity on machine learning led by one of the authors. Machine learning was selected as a follow-up topic because multiple faculty asked how artificial intelligence (AI) worked on the Session 1 Exit Ticket. The MIT Media Lab (n.d.) Cats, Dogs & Machine Learning lesson was adapted for this activity. This lesson is geared to fourth through eighth grade levels. Due to the limited amount of time provided for this second session, the activity was adapted to be unplugged and completed in around eight minutes.

To start this activity, a description of machine learning and AI was provided. This description was connected to the *Brown Bear, Brown Bear* book, read during Session 1, to feature recognizable labels for each animal (e.g., blue horse) and discuss how labels are used in technology. To introduce the information on machine learning and AI, speaking prompts were created along with a slide that included the aligning middle school standards (Slide 2 in Session 2 PD PPT).

> Prompt: *We are going to learn how supervised machine learning can be trained to classify complex datasets based on labeled data and make predictions about new pictures and live video feeds! AI is trying to predict something in the future or something that the data says. In a supervised machine learning system, a computer learns by example. From our Brown Bear Book last time, we see that each object has a label. We classify and label things in technology all the time such as: junk mail, face detection for snapchat filters, library databases, hashtags, etc. These labels and classifications help us make predictions on data and information that is not labeled.*

After the basic information was presented, a slide of three cats and three dogs was shown and faculty provided three characters that were similar for the three cats and for the three dogs (Slide 3 in Session 2 PD PPT). Faculty were then given multiple color printed images of cats and dogs and a big sheet of paper with Cats and Dogs written at the top. They were asked to use the three characteristics for cats and three characteristics for dogs to classify the images of the dogs and cats (Figure 10). After a few minutes of sorting, faculty discussed what occurred specifically asking the questions:

- *Did all the cats & dogs get classified correctly?*
- *What group had the higher percentage of correct identifications (cats or dogs)?*
- *What could we have done in our training data (the images on previous slide) to make it more accurate?*

> Prompt: *We are going to build a very simple cat/dog classifier like a CS person would for a computer. Based on these 3 pictures of cats, give me 3 similarities between them. Based on these 3 pictures of dogs, give me 3 similarities between them. Now at your tables, use these similarities as classifiers and classify the pictures into the cat and dog categories. Be sure to use the similarities on the board to classify the pictures. So, what happened?*

Finally, the activity concluded with a short discussion on algorithm bias and how the images used to identify similar characteristics were biased for and against types of dogs and cats (Slide 4 in Session 2 PD PPT).

> Prompt: *When algorithms, specifically artificial intelligence systems, have outcomes that are unfair in a systematic way, we call that algorithmic bias. We would say that our cat-dog classifier shows algorithmic bias and that it is biased towards _____ since it works really well for them and biased against _____ since it doesn't work as well for them. For computer systems, when we code and classify data, the data, labels, and categories we create impact what the computer learns.*

Following this activity, one of the faculty lead team members showcased the MD CS K-12 Student Standards and led a discussion on how CT/CS currently fits in the School of Education and TED vision statements. This first follow-up meeting concluded with an announcement of the five Focused Faculty and celebration for their willingness to participate.

Session 3: 1/25/24 PD

On January 25, 2024, the third CT/CS PD session occurred during the TED faculty meeting. In this meeting, the authors were given 15-minutes to conduct an activity.

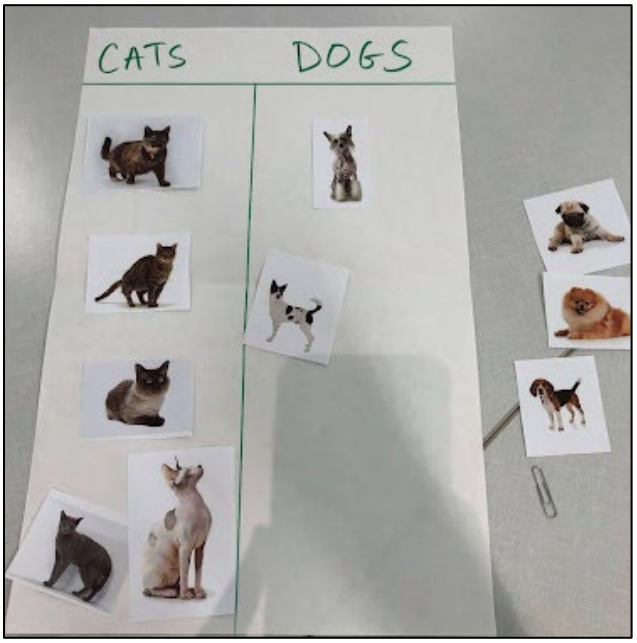


Figure 10. Image of faculty sort of cats & dogs based on identified similar characteristics.

Set-Up and Materials

This session was facilitated face-to-face in a room with round tables that seated six to eight people for communication and collaboration. This PD focused on a STEM/makerspace activity called "Build the Tallest Structure" to support the Whole-Part-Whole learning model (Swanson & Law, 1993). The materials needed for this PD were:

- Maker materials (e.g., Legos, popsicle sticks, cups/containers, straws, string, tape, playdoh)

Professional Development

One of the Focused Faculty requested support in connecting CT/CS concepts to the Whole-Part-Whole

learning model around the time of this PD session. Instead of providing a direct answer to this Focused Faculty member, it was determined to have all the TED faculty connect CT/CS to the Whole-Part-Whole learning model and provide some starter ideas. This activity was created by one of the authors with the help of OpenAI (2024). Open AI was used to brainstorm some STEM/makerspace activities that connected to CT/CS. The author who led the activity had a physical makerspace with multiple items and regularly integrated STEAM into their classes.

Before the PD, the author who led the activity gathered the materials and sorted materials for four groups. The materials provided to faculty for the activity were Legos, popsicle sticks, cups/containers, straws, string, tape, and playdoh, but any kind of physical materials would suffice as long as there is a mix of adhesive/bonding and building/stacking items.

During the PD, faculty were placed in teams and challenged to make the tallest structure using the provided materials. Faculty were given 8 minutes to build a structure (Figure 11). After the activity, a discussion occurred on how this activity connected back to the CT Core Concepts and how it connected to the Whole-Part-Whole learning model with the following prompts:

- *What was the task?*
- *How did you decompose the task?*
- *What types of patterns did you realize?*
- *How did you use abstraction in the task?*
- *What steps did you take to complete the task?*
- *Did you create the tallest structure? What could you have done differently if not?*
- *How does this connect to Whole-Part-Whole learning model?*

Session 4: 2/22/24 PD

The fourth PD session occurred on February 22, 2024, during the TED meeting. Another 15-minute time slot was provided, and the focus was on the connection of CT/CS to social emotional learning (SEL).

Set-Up and Materials

This session was facilitated online via Zoom. Breakout rooms were used for the paired reflection. Needed materials included participants using their mics to speak (or chat functions) and access to the Poem Art lesson by CODE (n.d.-d). A slide deck was used but no slide deck is provided as it only included the prompts which are provided below.

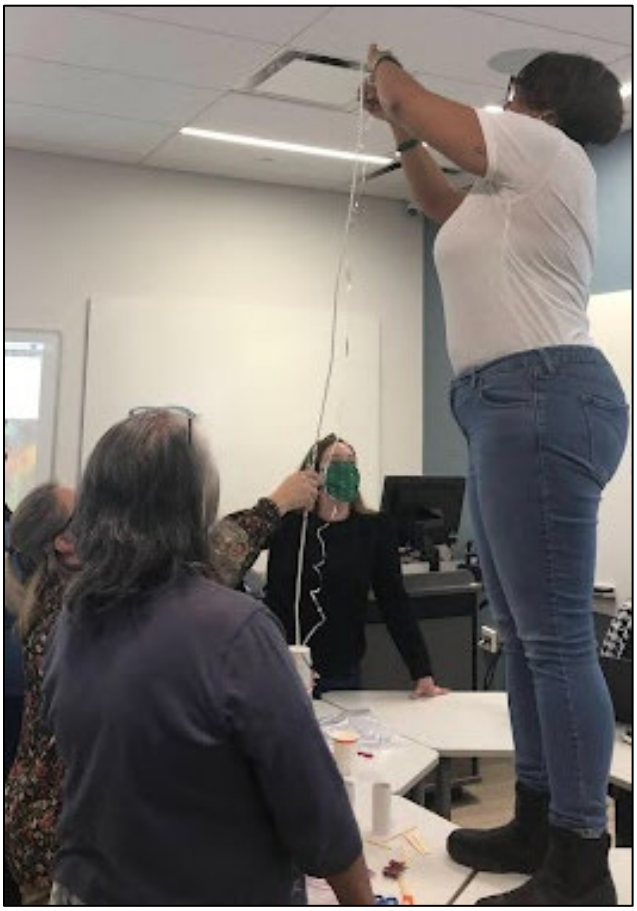


Figure 11. Faculty creating the tallest structure in the January 25, 2024, department meeting.

Professional Development

This session topic was request by another Focused Faculty member who was trying to create a CT/CS integrated lesson on SEL. This follow-up was online and began with the Poem Art lesson (CODE, n.d.-d). Faculty were given about eight minutes to try to get to the second or third progression (i.e., level) of the lesson. Faculty did not have to login to Code.org to complete this progression. After faculty were given time for the poem art, they reflected (individually and in pairs) about what was easy and difficult in the activity, how they reacted to difficult items, and how they demonstrated resilience. Four minutes was given for this reflection. The prompts included:

- *What was easy for you?*
- *What was difficult for you?*
- *When you hit a difficult part of the game, what did you do? (give up - low resilience, or keep trying until you solved it - high resilience)*
- *Why do you think you demonstrated low or high resilience when faced with a difficult part?*

Following the reflection, a brief discussion was held on how the code.org activity, both the activity of Poem Art & reflections, supported Social Emotional Learning. The CASEL framework was provided as a foundation for this discussion (CASEL, n.d.).

Session 5: 4/26/24 Wrap Up Showcase

Finally, on the last follow-up PD, the Focused Faculty showcased their created lessons, and the authors showcased the grant-purchased materials.

Set-Up and Materials

This session was facilitated online via Zoom. No breakout rooms were needed. Needed materials included a slide deck, created collaboratively by the Focused Faculty and authors, that showcased their CT/CS lessons and reflections, and the grant-purchased materials. No slide deck is provided as this is specific to implementation.

Professional Development

Each of the five Focused Faculty members showcased what they integrated in their classrooms and reflected on how it went and what they would change for the following academic year. Part of the funding from the grants that supported this work was used for reusable materials. Each of the Focused Faculty were able to purchase something for their lessons which can be used throughout the TED program. These items were highlighted for all of the TED faculty to consider integrating in the 2024-2025 academic year.

Focused Faculty

From the Session 1 full-day PD, five faculty volunteered to become the Focused Faculty that continued working with the authors in integrating CT/CS into their courses. To select the Focused Faculty, the authors decided to first see who was interested and able, as opposed to asking/requiring faculty to participate. From the initial request in the first PD session, five faculty volunteered representing these courses:

- Elementary and Secondary Math Methods
- Early and Elementary Literacy/Reading
- Elementary Education Capstone (a course taken in conjunction with internship)

The five Focused Faculty were teamed with one of the authors as a mentor and worked on creating and implementing at least one CT/CS activity, lesson, and/or assessment into one of their courses for the spring semester. The minimum requirement for this work was 10 hours. This time included approximately 3 hours of whole group meetings (the five Focused Faculty and the authors), 5 hours of co-planning (where each Focused Faculty member met with their mentor), and 5 hours of individual planning/preparation of the activities. The co-planning hours allowed the mentors to coach their assigned Focused Faculty member and support them in co-planning their integration and activities. It was encouraged for the Focused Faculty to have their faculty lead team member observe the CT/CS integration and be available for the lesson to support as needed. This informal observation also served as a component of the Focused Faculty annual evaluation in TED, allowing the Focused Faculty member to check that requirement off while also getting formative feedback on and/or support with their CT/CS lesson.

In the initial Focused Faculty meeting, the stipulations and payment information was presented, and guiding questions were provided to help the Focused Faculty choose the course to integrate CT/CS into and how to work with their mentor. Prior to this meeting, the authors met to select their Focused Faculty mentees. These were selected based on content area and course time (e.g., one of the authors teamed with a faculty member in a course they were already team teaching). The guiding questions used in this initial meeting were:

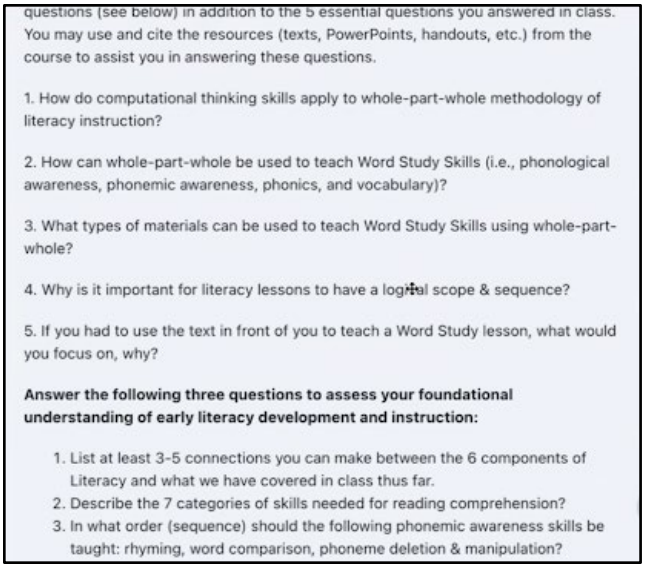
- *What class are you thinking of integrating CT/CS?*
- *What day/time/location does the class meet?*
- *When are you planning your syllabus/class?*
- *Set up 1st co-planning time for Spring 2024.*
- *Set up other co-planning times (2-3)*
- *Set up informal peer observation & debrief time (1)*

During the co-planning sessions between the Focused Faculty and their mentor, the Focused Faculty could ask questions, brainstorm ideas for CT/CS integration, and get more support on CT/CS skills. These co-planning sessions were scheduled between the Focused Faculty member and the mentor that best fit their schedules and the timing for the CT/CS lesson. Within these co-planning sessions, questions arose such as how to teach the Whole-Part-Whole instructional framework with CT/CS (see Session 3: 1/25/24 PD section) and Social Emotional Learning with CT/CS activities (see Session 4: 2/22/24 PD section). Since the Focused Faculty were teaching preservice teachers, the integration had to be a mix of content instruction and modeling, so the preservice students can learn the content and also see and/or demonstrate the application of that content in a classroom setting (K-12 spaces and in the course classroom). This meant that these CT/CS lessons needed to support the intended lesson content (e.g., Whole-Part-Whole Framework, SEL) while also demonstrating how the content is applied in a classroom setting. Some of these integration questions from the Focused Faculty became the focus of the CT/CS PD sessions in the TED meetings.

An example of how the Focused Faculty integration questions became a whole TED faculty PD is when one of the Focused Faculty members wanted to teach the Whole-Part-Whole instructional framework with CT/CS integration (see Session 3: 1/25/24 PD). First, they aligned the Whole-Part-Whole instructional framework to CT terminology by introducing the concept and the vocabulary. Students then had to answer a few questions in a discussion forum to show their understanding of the Whole-Part-Whole framework and how CT skills are applicable within the framework (Figure 12). The Focused Faculty member reflected on student answers stating the students could connect CT vocabulary to the Whole-Part-Whole framework, and when listing materials to teach Word Study Skills (question 3 in forum), students listed some of the CT/CS unplugged materials they used in the activities from the lessons.

Focused Faculty Integrated Lessons

Finally, near the end of the year, the Focused Faculty were asked to present their integrated lesson to the other TED faculty in a department meeting. The following integration activities, by course, were created and implemented by the Focused Faculty.


questions (see below) in addition to the 5 essential questions you answered in class. You may use and cite the resources (texts, PowerPoints, handouts, etc.) from the course to assist you in answering these questions.

1. How do computational thinking skills apply to whole-part-whole methodology of literacy instruction?

2. How can whole-part-whole be used to teach Word Study Skills (i.e., phonological awareness, phonemic awareness, phonics, and vocabulary)?

3. What types of materials can be used to teach Word Study Skills using whole-part-whole?

4. Why is it important for literacy lessons to have a logical scope & sequence?

5. If you had to use the text in front of you to teach a Word Study lesson, what would you focus on, why?

Answer the following three questions to assess your foundational understanding of early literacy development and instruction:

1. List at least 3-5 connections you can make between the 6 components of Literacy and what we have covered in class thus far.
2. Describe the 7 categories of skills needed for reading comprehension?
3. In what order (sequence) should the following phonemic awareness skills be taught: rhyming, word comparison, phoneme deletion & manipulation?

Figure 12. Discussion forum on how CT supports the Whole-Part-Whole Instructional Framework.

Course: Materials for Teaching Reading

- Introducing Whole-Part-Whole Framework Using CT Skills
- Intro to Genre-Based Writing
  - Three Stations:
    - Narrative Writing–Plugged activity using Scratch Jr.
    - Persuasive Writing–Unplugged activity using research
    - Procedural Writing–Unplugged activity using human robot in a maze
- Review: Integrating CT & Literacy
  - Three Stations using the Cubetto and two Code & Go Mice

Course: Early Literacy Assessment and Instruction

- Introduced CT concepts and vocabulary: logic, decomposition, patterns, algorithms, abstraction, and evaluation
  - Unplugged Activities
- Comprehension Retelling using *We're Going on a Bear Hunt* with unplugged activity map and code and go mouse with story map
- Students used Scratch Jr for narrative writing and reflected on how the plugged activity would be useful to get early elementary students motivated to write

Course: Assessments & Instruction in Reading I

- Introduced the six concepts of computational thinking: Logic, Evaluation, Algorithms, Patterns, Decomposition and Abstraction
- Introduced five computational practices: tinkering, creating, debugging, persevering and collaborating
- Students charted the results on the connections between literacy, reading comprehension/meaning-making and computational thinking
- Three Unplugged Stations:
  - Maze Activity
  - Three Word Story
  - Oral Reflection Sentence Frame

Course: Elementary Education Capstone

- Step 1:
  - Provided overview of computational thinking concepts and approaches
  - Used Scratch Jr.
- Step 2:
  - Read Monster at the End of the Book
  - Students used Scratch to create their own monster
- Step 3:
  - Read If You Give a Mouse a Cookie
  - Students used Code and Go Mice to sequence events in the book

Course: Elementary Math Methods

- Tech Day: TI NSpire used to program micro:bits and hubs
- Introduced CT & Code.org: Code with Anna and Elsa lesson (CODE, n.d.-b)

Course: The Teaching of Mathematics

- Desmos: teacher- & student created
- Tech Day: TI NSpire programming
- TI NSpire programing with micro:bits
- Code.org: Coding a Geometric Star Quilt (CODE, n.d.-c)

Course: Math for Elementary Teachers-Geometric

- Tech Day: TI NSpire with micro:bits
- Extra Credit → Code.org: Coding a Geometric Star Quilt (CODE, n.d.-c)

MATERIALS

In addition to the planning and co-planning support, Focused Faculty were given $100 each to spend for supplies for their CT/CS integration. This funding was part of the grants and allowed the Focused Faculty to purchase materials needed to implement the CT/CS lessons. They were encouraged to purchase materials that were not one-time use and could be utilized in other courses and by other faculty. The items purchased by the faculty included one Cubetto (PRIMO, n.d.), two additional adventure maps for the Cubetto, six Code & Go Mice (Learning Resources, n.d.), and some materials for unplugged activities. In addition to these materials, TED had access to 20 micro:bits (n.d.-c), purchased with the first grant, and 10 TI Nspire calculators (Texas Instruments, n.d.). These tools were utilized in the Focused Faculty courses in Spring 2024.

MICRO:BITS

The micro:bits were utilized during the first PD session and heavily utilized in the math courses. The math Focused Faculty member stated,

> I used [the micro:bits] in my mathematics methods courses (both elementary and secondary) to introduce block chain coding with Microsoft MakeCode's (n.d.) Flashing Heart Activity. Then we used them as dice for probability exercises. My secondary mathematics methods course also did some activities interfacing the micro:bits with the Texas Instruments Nspire calculators. Also, I used the micro:bits with my secondary methods course this past spring to do the Environment Exploration activity. I will use them again for this activity in the early summer course.

This faculty member integrated CT/CS in multiple of their courses going beyond the one lesson in one course minimum.

CUBETTO

The Cubetto was a purchase for one of the Focused Faculty to use in their literacy course (see Figure 13). The faculty provided a statement on how they used the materials for their course. This is what the faculty member stated about the Cubetto:

After scaffolding the integration of CT/CS principles using both plugged and unplugged activities into two lessons, I was able to use the Cubetto programming robot and coding mice to help solidify students' understanding and application of CT/CS integration into literacy content instruction. Students were able to apply and synthesize CT/CS principles with their understanding of how the brain processes and performs literacy practices and determine the implication of such in relation to literacy instruction in the elementary and secondary classroom. They had to read the texts provided with the Cubetto and follow the instructions to move either the Cubetto robot or the coding mice to the locations on the map as directed. Through this activity students were using CT/CS skills in real-time in collaboration with their peers and discussed how this and similar activities would be done in their future classrooms.

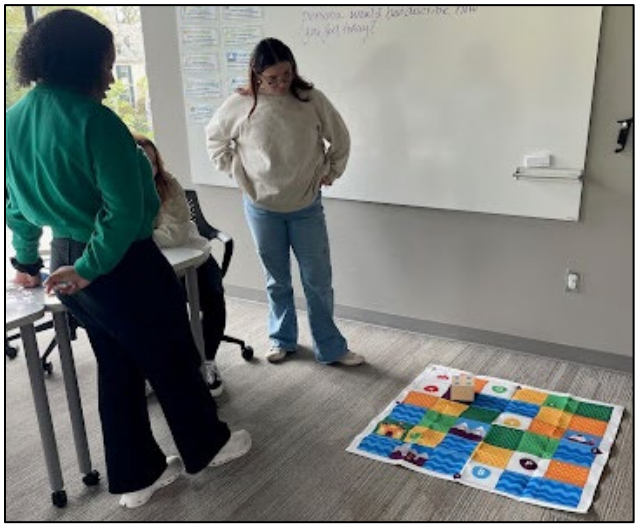


Figure 13. Undergraduate students using the Cubetto in a literacy class.

Code & Go Mice

The Code & Go Mice were purchased for two classes. First, in Early Literacy Assessment, the faculty member had students complete a comprehension retelling using *We're Going on a Bear Hunt* with an unplugged activity map and the Code & Go Mice with a story map (Figure 14). The students were placed in three groups to work with the Code & Go Mice, and sequenced retelling of *We're Going on a Bear Hunt* with story map using large grid paper.

Another faculty member completed a similar activity with the Code & Go Mice in the Elementary Education Capstone course with the book *If you Give a Mouse a Cookie*. Since two faculty completed similar activities in two different classes, this prompted a discussion amongst the authors on when activities should appear and how to scaffold the activities to ensure they become more complex in computing and integration for students. This led to a larger mapping of the current integration of CT/CS and the course sequence for preservice students.

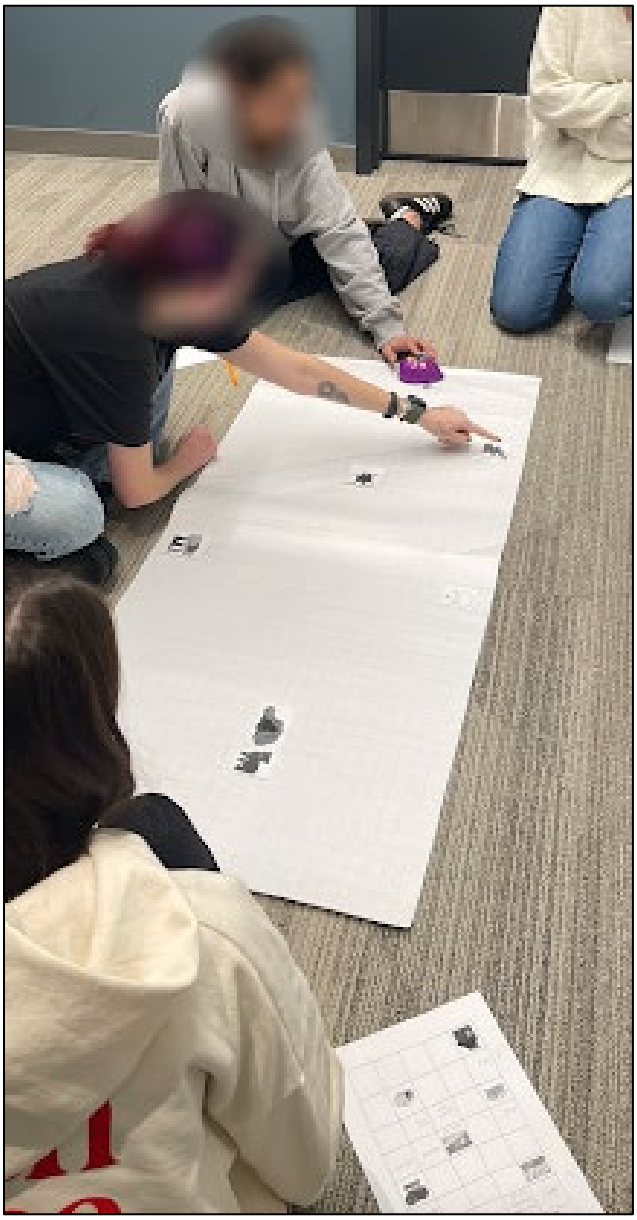


Figure 14. Literacy students using a story map with Code & Go Mice for We're Going on a Bear Hunt.

UNPLUGGED

One of the literacy faculty created unplugged activities in their Assessments & Instruction in Reading course stating,

> I utilized chart paper, 4x6 multicolored index cards, paper copy instructions and reflections (both on paper and oral) to complete the CT/CS unplugged activities. Students struggled at first but started to make connections between the 6 principles and 5 pedagogical practices and the practice of teaching literacy and the relationships to literacy concepts. The students all enjoyed the activities and after completing the activities could see the connections more readily (Figure 15).

Figure 15. Image of literacy students completing unplugged activities.

Students in this class also connected five computational practices to literacy, reading comprehension/meaning-making (Figure 16). The authors observed that many of the Focused Faculty were integrating CT/CS at multiple moments in their courses which they reflected on for future considerations.

CRITICAL REFLECTION

These sessions were implemented during the TED meetings which were already established meetings set in the calendar and communicated to faculty. It was expected for all TED faculty to attend these meetings. When planning whole-group PD, it is recommended to find time during an already-established meeting instead of creating more meetings.

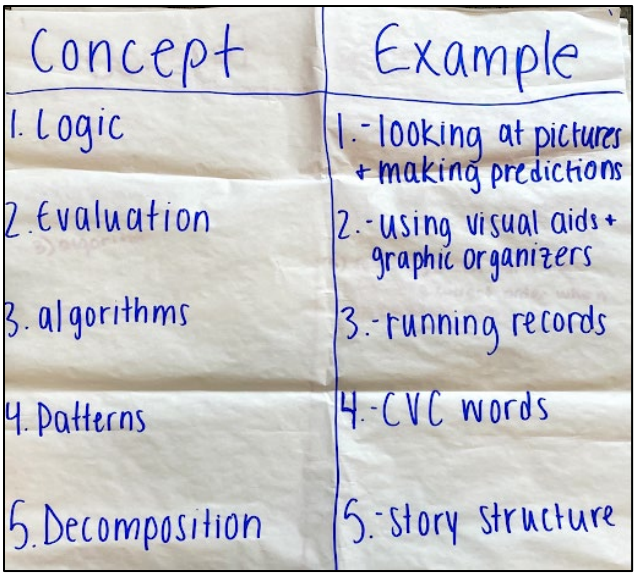


Figure 16. Images of computational practices to literacy, reading comprehension/meaning-making.

This was grant-funded which allowed for the payment of faculty for their time during the full-day, Session 1 PD, the payment of the Focused Faculty for 10 hours to plan and implement their CT/CS lessons, the purchasing of materials, and the payment of the authors for implementing this PD. This external financial motivation may have increased the number of faculty who joined in the full-day, Session 1 PD (although it was required) and the willingness of the faculty to join the Focused Faculty. Although internal motivating factors are the goal for any adult learning, it is recommended to support faculty time to learn, design, and implement activities with internal or external funding. There were limitations, however, as the funding only allowed five faculty to join the Focused Faculty in this first year.

The grant supported 10 hours of work for five Focused Faculty, but we asked the faculty to document the number of hours they spent on meeting, learning, and designing their CT/CS lessons. They each spent an average of 13 hours on CT/CS integration for their courses in the Spring 2024 semester with one faculty member spending 19 hours (Table 2). The requirement for the Focused Faculty was to integrate one lesson into one course. A few focused faculty members integrated CT/CS in

their course's multiple times, and one focused faculty member integrated CT/CS in all of their mathematics courses.

As the Focused Faculty created their CT/CS implementation activities, it was noticed that many of the activities completed in the Session 1 full-day PD and subsequent PD sessions were used in their courses (e.g., the unplugged maze activity). This speaks to the need for modeling for faculty so they can implement it in their courses. We focused on a hands-on, constructivist approach with all of the PD sessions, allowing faculty to experience and play with CT/CS activities and lessons instead of just talking about them.

Table 2
Faculty Focused Hours

Descriptive Statistic	Number of Hours
Minimum	10.5 Hours
Maximum	19 Hours
Mean	13 Hours
Median	11.5 Hours
Mode	11 Hours

This use of modeling and hands-on approaches is best captured in the Exit Tickets for the Session 1full-day PD. Faculty listed the *Brown Bear, Brown Bear* activity as one of their favorites stating, "you can use CT/CS in all content areas. Love *Brown Bear*," and "I can use CT/CS & coding with literacy (*Brown Bear*)," and "seeing there is room for CS/CT in ESOL courses." When implementing CT/CS with faculty, it is recommended to take a constructivist, hands-on approach to model to faculty how they should be implementing in their classrooms (similar to what we expect from preservice and inservice teachers).

There were a few challenges in our year-long PD. First, there was a limitation of the number of times the authors could join the TED faculty meetings and how much time they were given during those meetings. The authors were only able to join in four TED faculty meetings over the year. There were other faculty meetings in which the authors were given time to implement an activity, but none of the authors were available due to travel and conference schedules. In each meeting, the authors were given about 15 minutes to complete an activity, discussion, and follow-up from previous sessions. This time was limited, so many of the activities were introductory to spark ideas and conversation about integrating CT/CS in classes. The limited amount of time in each session was especially impactful in Session 2 about AI and machine learning. More time for faculty to sort the cats and dogs based on their initial descriptions and discuss was needed. Session 4 was also virtual, so the activities had to be adjusted from a face-to-face facilitation to an online facilitation. The use of code.org activities supported this online facilitation but did not allow the authors to view how many levels the faculty were able to accomplish nor if faculty were actually participating in the activity. More time in meetings would allow for deeper learning of CT/CS but may also disrupt the business and other items completed during the meetings.

During the Focused Faculty lessons, implementation issues did occur with the tools and faculty ability to fully describe the CT/CS components. These specific issues were not documented in this article as they are part of the Focused Faculty lessons and should be further discussed by those faculty.

We were able to secure another grant to send four faculty to the 2024 Maryland Higher Education Summer CS Intensive Conference. Two of the authors and two of literacy Focused Faculty attended this conference. The two Focused Faculty presented some of the CT/CS activities they created in their literacy courses at this conference with the help of one of the authors (their mentor).

The activities presented were the Three Word Story to introduce computational literacy vocabulary, and connecting CT/CS to the skill of reading through genre-based writing and the Cubetto and adventure maps. For the Three Word Story game, students are trying to get their partner or someone in their group to say a hidden word without actually using it themselves. First, students are put into pairs or small groups. One student picks up a word card and keeps it hidden from the rest of the group. That student then says three words to begin the story. The next person in the group says another three words to continue the story. This process is repeated until the hidden word is said, or time limit is reached. Check out Three-Word Stories with Benedict Cumberbatch for an example (The Tonight Show Starring Jimmy Fallon, 2014). For the Cubetto activity, students first start by brainstorming and listing the connections between the skill of reading and CT/CS concepts.

Then each groups reads their Cubetto adventure story listing the character(s), setting, goal of the character(s), problem, and resolution. Then groups coded their Cubetto robots (or Code & Go Mice as they were used with the Cubetto adventure maps) to follow the story. Groups then reflected and discussed the CT/CL concepts they used when reading and analyzing the story and coding the robots. This literacy focused CT/CS session was widely attended and provided an opportunity for the literacy Focused Faculty to showcase their CT/CS integration activities beyond their classroom and TED.

Department Chair Reflection

As department chair, I lead continuous improvement efforts that include curricular revisions to keep our programs current. As we implemented the plan over the year, it was rewarding to see teacher education faculty embrace CT/CS. Their enthusiasm for the PD activities we planned was evident. Many faculty members went beyond the expectations of the plan and implemented multiple activities with their teacher candidates. They were excited by the teacher candidates' engagement in the lessons and invited other faculty to their classes to observe and/or participate. Faculty enjoyed collaborating with each other during department meetings and continue to volunteer their own time to support each other with the integration of CT/CS.

Next Steps

During the 2024 Maryland Higher Education Summer CS Intensive Conference, the attending faculty reflected upon the 2023-2024 work and further developed the department's expansion plan (see Table 3). The plan continues into the elementary courses, resulting in more options for assignments in the final Capstone Seminar. In addition, consideration was given to exposing students to various technology or activities in at least two classes prior to the Seminar. For example, preservice teachers who work with micro:bits in Mathematics for Elementary Teachers-Geometric will also encounter micro:bits in Science Methods before being required to incorporate this technology in their lesson planning during the final semester of the program. This structure of revisiting the tools and concepts in multiple courses will ensure that the TED teacher candidates are prepared to implement CT/CS activities in their Capstone/Internship classrooms and future classrooms as practicing teachers.

We also continue to seek external grants to help fund this initiative and have acquired department and institutional funding to continue the work, including designing a cross-listed course for students in the teacher education and computer science departments that introduces the basics of CT and CS concepts for PK-8 classrooms.

Table 3
TED Revised Expansion Plan

Year	Course & CT/CS Tools/Activities (if known)
2024-2025	• Elementary Social Studies Methods: Code & Go mice (timelines) • Elementary Science Methods: micro:bits/makecode (environment) • Secondary Science Methods: Code.org: Outbreak Simulator • TI Rising Teachers Program • Mathematics for Elementary Teachers (Algebra): unplugged, Scratch Jr. • Capstone Seminar: Scratch
2025-2026	• Introduction to Special Education • Classroom Management • Elementary Mathematics Methods & Management • Elementary Social Studies Methods & Management • PD for PDS/Partner School Mentors • Assessment and Evaluation for Special Education • Collaboration and Consultation for Students with Special Needs • Principles of Behavior Management for Special Education • Foundations of Education
2026-2027	• Curriculum and Instruction for Special Education • Algebraic Concepts • Integration Science I • Improving Access to the General Curriculum for all Learners • PD for PDS/partner schools • Integration Science II • Methods of Teaching English • Methods of Teaching Social Studies • Methods of Teaching Art

References

CASEL. (n.d.). *What is the CASEL framework?* Retrieved June 4, 2025, from https://casel.org/fundamentals-of-sel/what-is-the-casel-framework/

CODE. (n.d.-a). *Anyone can learn Computer Science.* Retrieved June 4, 2025, from https://code.org/students

CODE. (n.d.-b). *Code with Anna and Elsa*. Retrieved June 4, 2025, from https://studio.code.org/s/frozen#

CODE. (n.d.-c). *Coding a geometric star quit.* Retrieved June 4, 2025, from https://studio.code.org/s/csc-starquilts-2023?viewAs=Instructor

CODE. (n.d.-d). *Poem art.* Retrieved June 4, 2025, from https://studio.code.org/s/poem-art-2021

CODE. (n.d.-e). *Unplugged: Computational Thinking*. Retrieved June 4, 2025, from https://code.org/curriculum/course3/1/Activity1-ComputationalThinking.pdf

Code.org. (2015, March 11). *Images, pixels and RGB*. YouTube. https://youtu.be/15aqFQQVBWU?si=Ws4sOC_qo1JuOtJh

Computer Science Teachers Association (2020). *Standards for CS teachers*. https://csteachers.org/teacherstandards

CS Unplugged. (n.d.). *About*. Retrieved June 4, 2025, from https://www.csunplugged.org/en/about/

CS Unplugged. (2021, September 26). *Image representation*. Classic Computer Science Unplugged. https://classic.csunplugged.org/activities/image-representation/

Dewey, J. (1930). *Democracy and education: An introduction to the philosophy of education*. The Macmillan Company.

Digital Promise. (n.d.). *What is computational thinking?* Retrieved June 4, 2025, from https://digitalpromise.org/initiative/computational-thinking/about/

International Society for Technology in Education. (2019). *Computational thinking competencies*. https://iste.org/standards/computational-thinking-competencies

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

Koshy, S., Twarek, B., Bashir, D., Glass, S., Goins, R., Novohatski, L. C., & Scott, A. (2022, December). *Moving towards a vision of equitable computer science: Results of a landscape survey of PreK-12 CS teachers in the United States.* https://landscape.csteachers.org.

Learning Resources. (n.d.). *Code & Go robot mouse.* Retrieved June 4, 2025, from https://www.learningresources.com/item-stem-robot-mouse

Martin, Jr., B. (1967). *Brown bear, brown bear, what do you see?* Henry Holt and Company.

Maryland General Assembly. (2020). *Legislation, session: 2018 regular session*. https://mgaleg.maryland.gov/mgawebsite/Legislation/Details/hb0281?ys=2018RS&search=True

micro:bit. (n.d.-a). *Dice*. Retrieved June 4, 2025, from https://microbit.org/projects/make-it-code-it/dice/?editor=python

micro:bit. (n.d.-b). *Environmental exploration.* https://microbit.org/projects/make-it-code-it/environment-exploration/

micro:bit. (n.d.-c). *What is the micro:bit?* https://microbit.org/get-started/what-is-the-microbit/#computing-made-physical

Microsoft Makecode. (n.d.). Flashing heart activity. Retrieved June 4, 2025, from https://makecode.microbit.org/projects/flashing-heart

MIT Media Lab. (n.d.). Cats, dogs & machine learning. *Canada Learning Code*. Retrieved June 4, 2025, from https://www.canadalearningcode.ca/lessons/cats-dogs-machine-learning/

OpenAI. (2024). *ChatGPT 3.5* (January 10 version) [Large language model]. https://chat.openai.com/chat

Piaget, J. (1952). *The origins of intelligence in children*. International Universities Press, Inc.

PRIMO. (n.d.). *Meet Cubetto.* Retrieved June 4, 2025, from https://www.primotoys.com/

Sawyer, H. (2022, November 20). *Lesson sequence- Unplugged to plugged.* Learning Beside My Learners. https://hsawyeresl.edublogs.org/2022/11/20/lesson-sequence-unplugged-to-plugged/

Scratch. (n.d.) *About Scratch.* Retrieved June 4, 2025, from https://scratch.mit.edu/about

Scratch in Practice. (n.d.). *Create a story activity guide.* Retrieved June 4, 2025, from https://sip.scratch.mit.edu/guides/story/

Scratch Jr. (n.d.). *Home.* Retrieved June 4, 2025, from https://www.scratchjr.org/

Swanson, R. A., & Law, B. D. (1993). Whole-part-whole learning model. *Performance Improvement Quarterly, 6*(1), 45-43. https://doi.org/10.1111/j.1937-8327.1993.tb00572.x

Texas Instruments. (n.d.). *TI-Nspire™ CX CAS graphing calculator.* Retrieved June 4, 2025, from *https://education.ti.com/en/products/calculators/graphing-calculators/ti-nspire-cx-cas?category=overview*

The Tonight Show Starring Jimmy Fallon. (November 18, 2014). *Three-word stories with Benedict Cumberbatch* [YouTube]. https://www.youtube.com/watch?v=DCataNWjw-Q

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3). https://doi.org/10.1145/1118178.1118215

Support Materials

- Code with Anna and Elsa lesson (CODE, n.d.-b)
- The Game with No Rules (CODE, n.d.-e)
- Images, Pixels and RGB (Code.org, 2015)
- Colour by Numbers (CS Unplugged, 2021)
- Scratch Create a Story Activity (Scratch in Practice, n.d.)
- micro:bit Dice Activity (micro:bit, n.d.-a)
- micro:bit Environmental Exploration Activity (n.d.-b)
- Cats, Dogs & Machine Learning Lesson (MIT Media Lab, n.d.)
- Poem Art Lesson (CODE, n.d.-d)
- Coding a Geometric Star Quilt (CODE, n.d.-c)
- Microsoft MakeCode's (n.d.) Flashing Heart Activity
- Scratch Jr. (n.d.)

About the Authors

Irene A. Bal is an Assistant Teaching Professor of Learning Design & Technology at Loyola University Maryland with research interests in microlearning, micro-credentials, teacher professional development, computational thinking and computer science for preservice and inservice teachers, and innovative PK-12 classroom practices and technology.

Karen L. Terrell is the Assistant Professor of Mathematics Education at Loyola University Maryland, as well as a Faculty Associate for the Center for Equity, Leadership, and Social Justice in Education. Her specialties include content and language integration, mathematics education, technology, and assessment for diverse learners. Dr. Terrell is currently the Editor of MCTM's journal, The Banneker Banner; Junior Co-Chair of the Research of Mathematics Education SIG of AERA; and a member-at-large for the Association of Maryland Mathematics Teacher Educators. She also serves on the Newsletter Working Group of the Bilingual Education Research SIG of AERA and a member of the TLC4ME Consortium, a nationwide research group for multilingual learners.

Hoang Bui is an Associate Professor in the Computer Science Department at Loyola University Maryland. He received a B.S. and M.S. in Computer Science in 2004 and 2007 from Midwestern State University. He received his Ph.D. in Computer Science in 2012 from the University of Notre Dame. From 2012 to 2015, he worked as a post-doctoral researcher at Rutgers University. After that, he spent seven years at Western Illinois University, often teaching programming and system courses. In addition to teaching, his research interests are computing education, big data storage, high-performance computing, and parallel and distributed systems.

Stacy Williams is a Clinical Professor and the Teacher Education Department Chair at Loyola University Maryland. She began her career teaching high school biology and mentoring early career teachers and is now dedicated to preparing teachers who will advocate for equitable and anti-racist education for all.

Acknowledgements

This professional development was produced with the assistance of two grants from the Maryland Center for Computing Education within the University System of Maryland under the auspices of the Maryland Preservice Computer Science Teacher Education Grant Program.

Sharing & Modification Permissions

Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9345

Scratch Day: Hands-On Computational Thinking Activities for Youth and Adults

Ayanna Perkins[3], Elexis Allen[1,2], Kiyah Stokes[1], and Danielle Jones[4]
[1]CodeCrew, [2]Eastern University, [3]The University of Memphis, [4]University of Florida

Overview

This lesson engages K-12 students, educators, and parents in original Scratch Day activities developed for lower-level (K-5th grade), upper-level (6th-12th grade), and post-secondary (adult) audiences. These lessons targeted individuals with limited knowledge regarding computational thinking. Activities involved sequencing and algorithmic expressions using block-based coding on the ScratchJr and Scratch web apps. Participants engaged in collaborative conversation and problem-solving as they made creative design decisions and debugged when they encountered coding issues.

Topics: Block Coding, Scratch, Computational Thinking, ScratchJr, K-12 Computer Science

Time: 1 hour 30 minutes

Materials

General
- Projector/Screen

K-12
- Scratch.mit.edu
- ScratchJr
- Olympics Presentation
- Scratch Jr. Presentation
- Scratch JR for Desktop Website
- CodeJr.org Website

Parent and Educator
- CSEdu Resource List
- Scratch Day Presentation
- Gymnastics Background
- Track Background
- Student Username List

Context-at-a-Glance

Setting
A community day of coding hosted on a Saturday by a local educational non-profit organization in the southeastern United States

Modality
Face to Face

Class Structure
Learners were grouped based on their current education level: Elementary, Secondary, and Post-Secondary. Each group included between 5 - 20 learners. One instructor and 1 - 2 teaching assistants were assigned to each group. During a single session, learners worked in a computer lab as instructors monitored progress through each activity. All coding activities used the block-based coding platforms provided by the Scratch Foundation, e.g., Scratch and/or Scratch Jr.

Learner Characteristics
Participants of all ages and coding experience levels engaged in an informal learning experience and created artifacts through direct experience, observation, collaboration, and active participation.

Instructor Characteristics
Licensed K-12 teachers with Computer Science teachers with bachelor's degrees in computer science taught K-12 students. A software engineer and instructional designer taught adults.

Development Rationale
Scratch Day was funded in partnership with the Scratch Foundation. This experience provided an accessible introduction to coding tools.

Design Framework
Kolb's (1984) Experiential Learning Theory

Setup

The set-up should take between 30 minutes and 1 hour, depending on the speed of the computers in use. The learning environment should ideally include rows of chairs and tables that allow the instructors to see all screens from the back of the room. To encourage collaboration and support, learners should be arranged in seating where they are near at least one other learner.

Instructors should have Google Chrome browser tabs open displaying Scratch and ScratchJr on the projector screen before participants arrive. They can place a printed copy of the PowerPoint slides at each workstation.

Standards

The following are standards from the Computer Science Teachers Association (n.d.).

1. Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
2. Create programs that include sequences, events, loops, and conditionals.
3. Design projects that combine hardware and software components to collect and exchange data.
4. Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.

These standards represent the ISTE Computational Thinking Competencies for Educators (ISTE, 2020).

1. Develop resilience and perseverance when approaching CS and CT learning experiences, build comfort with ambiguity and open-ended problems, and see failure as an opportunity to learn and innovate.
2. Evaluate and use CS and CT curricula, resources, and tools that account for learner variability to meet the needs of all students.

Context and Setting

Currently, 42 states have adopted Computer Science (CS) educational standards for K -12 instruction, and a growing number of states – including Tennessee (TN Code § 49-1-232, 2021) -- have included Computer Science as a graduation requirement. During the rollout of Tennessee's CS educational requirements for all grades, parents and core-subject classroom educators were often not a targeted audience for early exposure and engagement activities. While formalized CS professional development was provided to those seeking licensing endorsement, general education classrooms and those post-endorsement received limited continual content-knowledge support, which is a common occurrence (Ni et al., 2021).

In 2015, realizing their communities "were on the wrong side of the digital skills gap", Memphis technology industry professionals Meka Egwuekwe, Audrey Willis (nee Jones), and Petya Grady founded CodeCrew, a non-profit organization to educate "underrepresented communities to be tech innovators and leaders through practical, hands-on computer science training" and ultimately "have an effective instruction program……[to] train teachers" (Code Crew, 2016, p. 35).

While best known for its annual competitive, multi-day hack-a-thon where K-12 students work collaboratively on a design challenge, CodeCrew primarily works directly with local educational agencies for school-based, after-school, and summer programming for students. The organization also provides an early career training program, CodeSchool, that targets students ages 18-30 to become certified entry-level software developers within a six-month course. Using a scaffolded approach for learners with limited experience and of various ages, the organization offers ongoing learning opportunities about various computing topics to community members, such as teachers, parents, business-owners, and others.

Setting

With sponsorship from the Scratch Foundation, CodeCrew hosted a Scratch Day on Saturday, August 24, 2024, in partnership with Southwest Community College on their Macon Campus (CodeCrew, 2024). The Scratch Foundation developed Scratch Day to engage communities in Scratch, "where people come together to celebrate creativity, coding, and learning through hands-on activities" ("Celebrate Scratch Day! — Scratch Foundation," n.d.). This community engagement event was designed as a free offering for local school-age youth to participate. Over 100

students signed up for the 3.5hr event and utilized the campus computer labs to engage in activities based on their grade level.

The organization also sought to offer local educators professional development training in computational thinking and the opportunity to build a community of practice (Ni et al., 2021 & Davis et al., 2018) by hosting a Scratch Day for Educators simultaneously. Parents were also targeted to increase parental awareness and support of CS education. Educator and parental enrollment were sought via social media advertisements, flyers shared with the local school system, and onsite day-of offerings.

Class Structure

Each session lasted 1.5 hours in a computer lab, with tables arranged in groups of four that encouraged collaboration. All monitors faced the back of the room so that instructors could monitor progress. One instructor was assigned to each group to lead them through their respective activities. The organization provided parents and educators with a separate room to complete two activities. Each activity included between 15 and 20 students. Educators and parents engaged in one K-5 lesson and one 6-12 lesson that mirrored the K-12 activities.

Activities

All activities were developed by CodeCrew using the popular Scratch coding platform. Scratch was used to develop students' computational thinking skills through block-based programming (Fagerlund et al. 2020). ScratchJr, a popular programming language and platform, is based upon the more advanced programming language and platform Scratch (Resnick et al., 2009). ScratchJr is tailored for younger audiences but still promotes the education of fundamental programming concepts.

Student Characteristics

Learners were students with limited experience in coding. The K-5 students varied in reading levels, with some students still learning to read. The 6-12 students varied in experience levels, from no coding experience to basic coding experience. All adult learners (parents and educators) had limited experience with CS programming and instruction.

Instructor Characteristics

The instructors included licensed K-12 teachers who hold bachelor's degrees in CS. These teachers have extensive experience instructing K-12 students and knowledge of CS concepts and skills. The instructors for the adult learners included a former teacher and instructional designer with limited experience in CS programming, and a software engineer with extensive programming experience. Experience with Scratch and ScratchJr is recommended to implement this lesson. However, any confident instructor can implement these activities using the walk-throughs provided in the slide decks.

Learning Representation

Learning Objectives

Scratch

Learners will:

- Program external hardware (i.e. Micro:bit) to control structures and functions in the game.
- Utilize math concepts for object placement and movement.

ScratchJr

Learners will:

- Engage in problem-solving activities using ScratchJr.
- Develop animations to tell a story using ScratchJr.

Lesson Structure

Step 1: Student Connection (15 minutes)

Scratch

During Scratch activities, the instructor can use coding and logic skills to connect with current world events. The Olympics had recently passed at the time the organization hosted the event. At the beginning of the lesson, students are encouraged to answer

questions involving sports: “Do you play sports?” or “Who is your favorite athlete?” This allows students to connect with the activity.

ScratchJr

The instructor introduces the hosting organization through a video that illustrates the activities that their peers participate in during CS-related events. This presentation gives students a peek at to expect and to bring excitement the atmosphere.

The instructor introduces ScratchJr and gives four key words to describe the program: video games, block-coding, storytelling, and programming language. Also, the instructor explains to students that ScratchJr is an application that is friendly to young scholars and that there is much to be learned within the application. Afterwards, students experience a digital gallery of ScratchJr projects created by previous students. The instructor creates the gallery in a PowerPoint presentation using four videos of student animations. The instructor then asks students what they like about the projects.

Step 2 Guided Instruction (45 minutes)

Scratch Micro:bit Integration

The instructor gives a simple explanation of the Micro:bit, its functions, and its extension inside of the Scratch platform. The instructor emphasizes that the Micro:bit is a convenient way to introduce hardware to beginners. The instructor explains that the Micro:bit's accelerometer detects movement and tilting, while its buttons allow for user input. Furthermore, the instructor details that the accelerometer can be explained using X, Y, and Z coordinates, an elementary math concept.

Activity 1: Track Running

The instructor first explains that students will be programming their character sprite to jump over a hurdle. The hurdle is created by the students, made of colored lines in the costume customization section. Students will navigate to the Choose a Sprite button located within the character staging section. Students will then hover over the Choose a Sprite button and select the Paint option available in the Choose a Sprite dropdown. Students will create a new sprite by drawing the lines to create the hurdle.

The Instructor will model this process and monitor student progress. Once the first hurdle is completed, it just needs to be duplicated to create additional hurdles. There must be at least two hurdles created but no more than four. The hurdles must be placed evenly on the track background. See Figure 1 for an example.

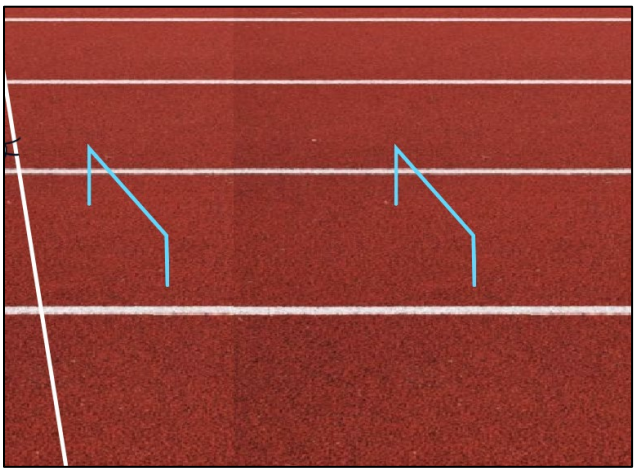


Figure 1: Hurdles

Adding the character sprite is the student’s next step. The instructor encourages students to find a sprite that they like from the Sprite library. The instructor then guides them in programming the jump action over the hurdle with their sprite. The instructor guides students to add the Micro:bit extension blocks. Once the blocks are imported, students need to set the tilt function to make their character move to the first hurdle by changing costumes. The goal is to get their character over the hurdle. The character is programmed to restart if it touches a hurdle. See Figure 2 for an example of the blocks.

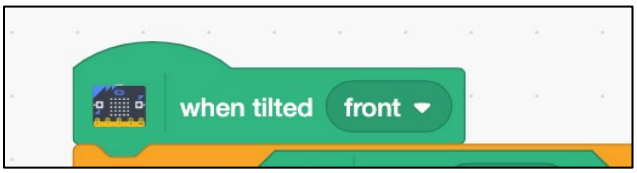


Figure 2: The “When Tilted” Block

The mathematical concepts in this activity are coordinates and simple algebraic expressions. The example the instructor gives requires students to determine how far the character should jump to not touch the hurdle. The instructor will model the next step: to add a block to change the background to the next scene, which is the Gymnastics portion.

Activity 2: Gymnastics Flipping

In the Scratch Gymnastics Activity, the instructor guides the students to flip their character sprite in the air. Due to Scratch's environment, there is a specific degree of rotation the sprite must follow to complete a full turn smoothly. The activity incorporates geometric math and loop concepts that challenge the students' ability to visually see what needs to be done to successfully rotate the sprite in a complete rotation (Olssen, 2022). The goal is to program the front tuck acrobatic motion.

The instructor provides the necessary backgrounds and sprites to complete the activity. After the sprites and backgrounds are loaded, students must use X and Y coordinates to determine the character sprite's landing spot. Referencing the solution from Activity 1 could assist students in programming the more difficult Activity 2 motion.

The final part of the activity is to create a function to perform the gymnastics event code based on input from the Micro:bit. The Micro:bit provides different sensors that can be implemented to detect input for this activity (Milić, 2018). The instructor has students verify the Micro:bit is still connected to the project. Students will program Button B to execute a function to make the character sprite jump before performing the front tuck motion over and over.

Next, the instructor asks students to add the Button-pressed function block. This is used to trigger the character sprite to begin flipping. Once the correct coding blocks are connected for the character sprite's tuck, they must be connected to a Micro:bit Button-pressed block for the board buttons to work inside the game. The goal here is to place the character sprite on the beam by jumping and flipping over. After functions are set accordingly, students must problem-solve to correct the X and Y coordinates and integers to complete the task smoothly.

The lesson is completed after Micro:bit integration in the gymnastics task. The instructor encourages students to take creative liberties for their programs after all tasks are completed.

ScratchJr

The instructor starts the lesson by explaining the key navigation features of the ScratchJr framework to the students. When ScratchJr launches, two option buttons are given to users: Home and Help. The instructor identifies the two options and instructs students to click the Home button. After clicking the Home button, the empty My Projects menu appears. The instructor explains to students that this space is where projects are saved. The instructor identifies the Create Project button located in the menu and instructs the students to click the empty canvas denoted with a plus sign in the center as seen in Figure 3.

Figure 3: The ScratchJr Project Menu

After creating a blank project, the instructor tasks students with exploring the interface and identifying symbols that may be familiar to them. Students identify various aspects, such as the green start flag for executing code and the paintbrush used to customize characters.

The instructor introduces the programming commands that students use to build their programs. Commands are grouped into categories based on function. The instructor identifies each category based on function and color. The instructor names the categories as follows: yellow blocks represent trigger blocks, blue blocks represent motion blocks, purple blocks represent look blocks, green blocks represent sound blocks, orange blocks represent control blocks, and red blocks represent end blocks.

After the introduction of block categories, the instructor navigates students through an interactive activity of using at least two blocks from each category to display their function. The instructor paces through each of the categories from left to right of the ScratchJr interface, starting with the trigger blocks.

The instructor then allows students to physically act out the actions of the blocks before running their programs. Students find this to be a nice connector to the lesson and identify the actions of the blocks consistently throughout the activity.

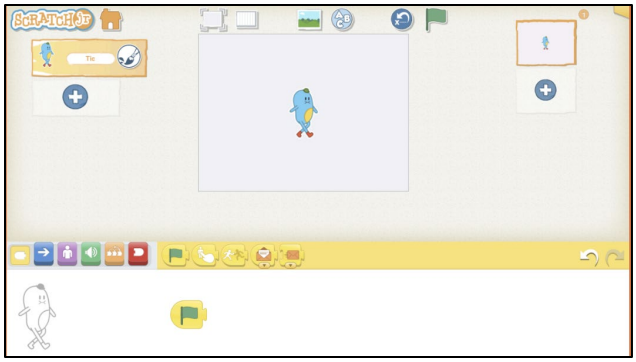


Figure 4: The Initial Project Interface of ScratchJr

After introducing the commands, the instructor models adding and removing characters. Before experimenting with the characters, the instructor introduces a synonym for character that is commonly used in the animation space: the sprite. The instructor refers to the characters as sprites for the remainder of the session. The instructor introduces the starting sprite, Tic, which displays when the project is created. Tic appears in the center of the project canvas (see Figure 4). The instructor asks the students if they would like to experiment with any sprite other than Tic. The instructor then gives directions to click the Add sprite button. The Add sprite button is referred to in Figure 5.

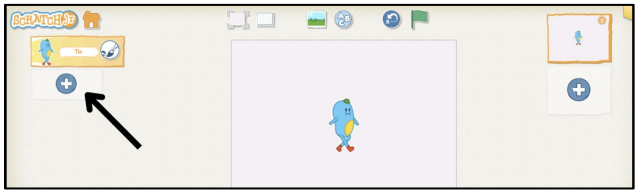


Figure 5: Add Sprite Button

The instructor gives students five minutes to look over the sprite selection and choose a sprite to be added to their project. After adding their chosen sprite, the instructor gives students guidance on how to remove Tic from the canvas. The instructor observes the canvas of each student, ensuring that only the chosen sprite remains on the canvas.

After the sprite section is complete, the instructor introduces a section centered around adding backgrounds. The instructor guides the students in clicking the Add Background button, which opens a database of backgrounds (see Figure 6). The instructor gives students the responsibility of choosing the background of their choice.

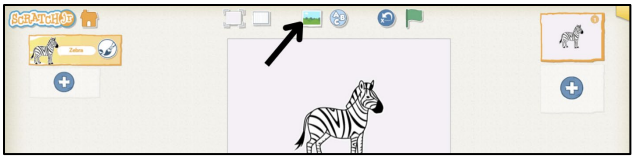


Figure 6: Add Background Button

After finishing the Background section, the instructor leads the students into creating their first program by prompting an activity with the objective of shrinking the sprite on the canvas. Students are instructed to add and connect a trigger block and a look block in their workspace, resulting in their sprite being shrunk. This program is demonstrated in Figure 7.

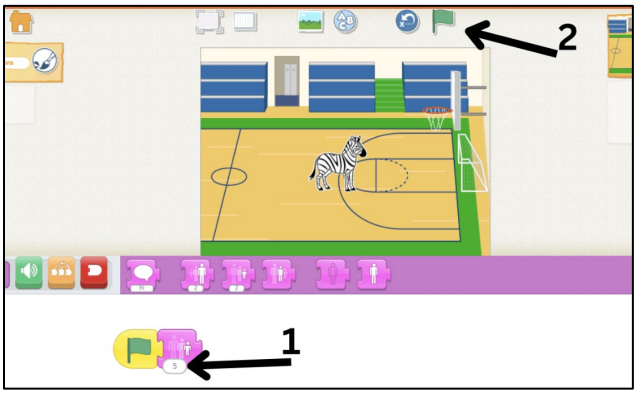


Figure 7: Sprite Shrink Program

After the sprite shrink section, the instructor focuses on creating a race using multiple characters and using commands within the control block that control the speed of characters. Figure 8 provides an example of the program for one character.

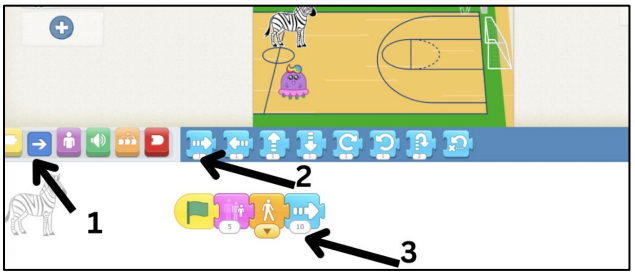


Figure 8: Sprite Race Program

After the sprite race program, the instructor allows students to use the remainder of time to create their own stories and optionally collaborate, encouraging students to experiment with building programs from blocks they haven't utilized yet.

Step 3 Problem Solving (30 minutes)

Scratch

There are numerous bugs and errors that arise during activities. The following errors are the most common:

Error 1 - Sprite Turns are not landing correctly in the Gymnastics portion: The common error is miscalculation in degrees or loop repeats. The instructor assists the students by decomposing the numbers used in the repeating loop.

Error 1 Solution: To fully complete the task, the instructor encourages students to use a simple algebraic expression to determine how far the sprite must turn in its rotation in the repeating loop. The instructor performs math equations to predict what numbers are needed. An example would be *Degree of turn = 360/Loop repeats.*

Error 2 - Sprite is constantly starting over in the Track portion: The common issue is the Y position is not gaining enough pixels to get high enough to miss the hurdle, causing the sprite to constantly hit the hurdle.

Error 2 Solution: The instructor points out the errors in the students' sprite coordinates. The sprite will need to move longer along the Y-axis and add more height to its path.

Error 2 Alternative Solution: The instructor points out the errors in the students' hurdle coordinates. The hurdles may be too large for the sprite to jump over correctly. The instructor tells students to reconfigure their hurdles by making them smaller or moving them lower.

ScratchJr

Although ScratchJr is designed for ease of use among early learners, navigating the platform becomes significantly harder as students begin to add more coding aspects to their canvas, as well as additional sprites.

Error 1- Programming the wrong sprite: A frequent issue that students encounter is programming the wrong sprite to complete a certain action.

Error 1 Solution: The ScratchJr platform allows users to visually identify which sprite is being programmed with an outline image of the character that is positioned to the left of the workspace. Figure 9 illustrates the placement of the sprite's outline.

The instructor guides students in correcting this mistake by instructing students to identify the character currently being programmed. Next, the instructor allows students time to review the program connected to the character, ensuring that the student understands that the program is specifically designed for that individual character. The instructor will guide the student to modify the original character's program to achieve its intended function if there is an error in character selection. Next, the instructor directs the student to add the intended character to the workspace and program the character to perform its intended function. Finally, the instructor guides the student to run the program and analyze the character's output animation. The instructor encourages the student to alter their program until the desired output is achieved.

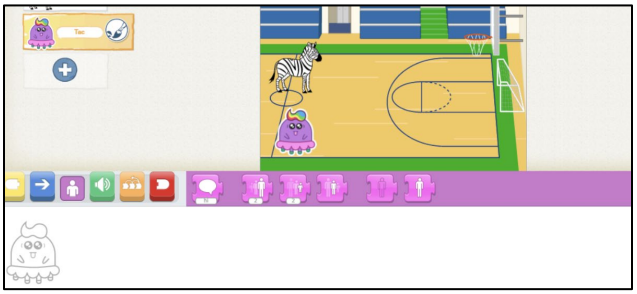


Figure 9: Outline of Selected Sprite

Error 2 - Deleting the Program: Students occasionally have issues with deleting a portion of their program and do not know the necessary steps to undo a program.

Error 2 Solution: The instructor leads the students in identifying which aspects of their program have changed since the deletion of their code. Next, the instructor paces the students through how to retrieve previous code using the previous and next buttons to the far right of the coding blocks. The instructor encourages students to compare the previous and current versions of the program. Additionally, the instructor encourages students to run their program between instances of using the previous and next buttons to introduce a visual comparison method for the students (see Figure 10).

Figure 10: The Previous and Next Buttons

Error 3 - Forgetting the Function of Blocks: Based on how various students' programs develop, the instructor may notice that some students focus on certain category blocks over others. An effect of this is students forgetting the function of blocks that they less frequently interact with.

Error 3 Solution: The instructor identifies the blocks that students have a difficult time memorizing. The instructor then provides printouts of the ScratchJr coding blocks and places them at the computer workstations. The instructor frequently references these blocks when students have questions about the functions of the blocks. Additionally, the instructor encourages students to test the functions of the blocks by experimenting within their programs. Finally, the instructor encourages students to analyze the visual function of the blocks, allowing students to become visually aware of the functions.

Differentiation Strategies

Instructors used various differentiation strategies.

Printed Walk-Throughs

One strategy included the use of physical copies of each presentation. Instructors were able to identify where participants struggled by comparing provided printed code to participants' typed code. Instructors also used printed copies of each presentation as a differentiation tool for all experience levels, allowing participants to work at their own pace.

Collaborative Learning

In each session, participants were encouraged to collaborate on design efforts after initial instruction. Participants collaboratively discussed designing characters, compared and contrasted programming methods, and reinforced their previous lesson by duplicating work to multiple pages. Through these efforts, participants assisted each other in reinforcing learned concepts and skills.

Interest-Based Learning

To engage participants, efforts to identify and apply the interests of the participants were implemented during each activity. Although tutorials included specific characters and backgrounds, participants were encouraged to change these to an object of their interest. These alterations are minor but foster an student-focused environment full of joy leading to long-term engagement (Purcell et al., 2021).

Creative Support Materials

For ScratchJr in particular, instructors also offered familiar stories and nursery rhymes that could be used by anyone unable to develop their own story. This method not only saved time but it also provided support for those who were experiencing creative difficulties.

Modalities

Learning can be improved and adapted to various learning preferences by using multiple modes of information, also known as multimodal learning (i.e., visual, kinesthetic, auditory, and reading) (Bouchey et al., 2021). The instructors designed the activities to support multimodal learning. This included providing a visual representation of the code, modeling each step, explaining the Scratch and ScratchJr site layout, and using consistent vocabulary across activities and age groups.

Critical Reflection

Computational thinking can have undeniable benefits, especially with early initiatives. However, there could be some difficulties when implementing it. Technical issues are common when programming with beginners across age groups. The learners are usually slow typers and unfamiliar with moving files across web pages, making steps more difficult to complete. Another challenge that may arise is engagement in general. Computational thinking can be tailored to any learning preference but as the lessons become more abstract, it can overwhelm beginners, hence losing focus and motivation. This is why the instructor should be able to gauge learners' preference and comprehension levels.

We conducted two learning representations, with two additional teachers leading another K-5 and 6-12 group during the Scratch Day event. Approximately 80 students participated in the session.

Approximately 5 parents, 3 teachers, and 1 teacher who also had a child to participate in a lesson, also attended.

K-5 Learning Representations

Both readers and emergent readers engaged with the buttons and explored each of the main features in ScratchJr. We understood that students came from various literacy backgrounds, so we chose ScratchJr as the preferred platform due to its flexibility. All pieces used in the coding aspect are visually stimulating, relying on color and symbols instead of words (de Ruiter & Bers, 2021).

Through repetition and modeling, students had multiple opportunities to make connections between using the blocks, sequencing blocks and testing the expected outcomes. By describing the icon on each of the blocks and physically demonstrating what the blocks would do, the instructor provides a physical representation that students can understand. Furthermore, students thought creatively to develop their own stories and imagined original scenes.

The instructor taught students CS terms, such as "sprites" to describe characters. This integration of vocabulary throughout the lesson helped students to make connections to more complex concepts. Students experienced debugging and troubleshooting as they developed their scenes.

The instructor also questioned students and modeled to demonstrate that sequencing matters. Questioning allowed students to think critically about their next action and test their hypotheses by running the programmed sequence.

The K-5 session's goals were for all participants to engage in problem-solving and develop animations to tell a story using ScratchJr. All participants achieved both objectives.

While the instructor covered all the material in 1.5 hours, the learning representations only represented an introductory level of content knowledge. A second session, covering more detailed loops and sequences, would have benefited students by building the foundational knowledge that they need to translate the coding skills gained in ScratchJr to text-based coding in the future. For instance, students started to debug and code as they created their stories; however, their experience with debugging varied based on the programs that they chose to create. A second session would provide opportunities for students to debug a pre-programmed project in a controlled environment.

6-12 Learning Representations

Students are often consumers of technology, but they do not think about the hardware that powers their favorite applications and technological components. By using the Micro:bit, students had the opportunity to make their own connections between the physical controller and the sprite. Integration of hardware and software allows students to make connections with both important components.

The 6th-12th grade sessions' goals were to use math concepts for object placement and movement and to program external hardware (i.e. Micro:bit) to control structures and functions within the game. All students met the session objectives to an extent, but most of the students were not able to complete the full activity. Students enjoyed the contextualization of sports, gaming (through the use of the Micro:bit as a controller), and computational thinking.

Two prevailing concepts across both 6-12 activities included pattern recognition and the use of algebraic equations. For instance, when the sprite jumped over the hurdles, students learned to recognize a pattern. The sprite also had to move a certain distance from one hurdle to clear the next hurdle, which also created a number pattern in terms of distance. Moreover, the gymnastics activity challenged students to use their mathematical knowledge to create an algebraic equation so that they could program the sprite to flip over the balance beam.

There were two activities given to students; however, due to time constraints, all students completed the Track activity but did not complete the programming for gymnastics. For future representations, we would only select one activity and provide the second activity as an option for advanced students. Since the gymnastics activity integrates math and requires more steps, we would focus the time on the gymnastics activity so that students have adequate time to problem-solve productively.

Students were able to collaborate verbally with other learners to initiate creative thoughts and debugging strategies for their individual programs. The

instructor encourages changes to be made after students develop confidence in their coding abilities.

Adult Learners

The learning representations for parents and educators worked better with the ScratchJr activity, where parents used sprites to tell a simple story. Adult learners felt more comfortable with the simple format and limited features of ScratchJr than they did with the more feature-rich Scratch platform. Overall, adult learners were successful in completing the Scratch Jr. activity. Adult learners completed each section of the activity without issue; no modifications were needed. Parents and teachers additionally expressed interest in sharing what they learned and feeling more confident engaging in future coding activities.

The timeframe provided for the completion of both activities presented a significant challenge. Parents and educators did not complete the Scratch Olympics exercise because it took more time to complete each step of the preceding ScratchJr activity. Due to a limited Micro:bit supply, the adult learners utilized the virtual Micro:bit extension available within the Scratch platform. The virtual Micro:bit functions just as a physical Micro:bit and is available for on-screen functionality. Parents and educators had difficulty with using the keyboard features for making their character move across the screen in Scratch. We modified this activity by demonstrating the actions that the sprite could do using the external feature.

Modifications

For the ScratchJr activity, we chose to use an emulator instance of the mobile application so that participants could use the larger, more familiar desktop workstation over the size limitations imposed with tablets and phones. Officially, ScratchJr is only supported as a mobile app, which requires it to be downloaded and installed onto a mobile device and all data is stored locally. We chose to use these alternative websites to protect the privacy of the learners, reduce the time needed to create an account during the event, and ensure that all participants could participate using the desktops at the community college. However, the sites mirror one another with slight differences in the sprite, or character, sections.

The varying attention spans of our K-5 students proved challenging due to their young age. Solutions included frequent breaks and impromptu games that incorporated the printed representation of the digital blocks. For instance, we encouraged students to quiz other students and the instructor by holding up different printed representations of the blocks and asking, “What does this block do?” This review kept engagement high and positive. Furthermore, oral instruction was reduced within each phase of the activity as it seemed students’ attention waned.

Our 6-12 students also struggled to maintain attention during instruction. Although their attention span is relatively better than K-5 learners, we had to make real-time modifications, like letting participants explore other characters to keep students engaged. 6-12 students were also allowed impromptu breaks from instruction. Instructions were simplified and shortened to prioritize and maximize student engagement, like the approach used in K-5 instruction. Overall, these real-time changes positively impacted the instruction and allowed for the facilitation of effective activities. With these modifications, the 6-12 students were able to stay more engaged with the content and were less likely to get distracted by personal devices.

References

Bouchey, B., Castek, J., Thygeson, J. (2021). Multimodal learning. In J. Ryoo, & K. Winkelmann (Eds.) *Innovative learning environments in STEM higher education: Opportunities, challenges, and looking forward* (pp. 35-54). Springer Briefs in Statistics. Springer. https://doi.org/10.1007/978-3-030-58948-6_3

Celebrate Scratch Day! — Scratch Foundation. (n.d.). https://www.scratchfoundation .org/scratch-day

CodeCrew. (2024, September 4). *Scratch Day with CodeCrew 2024 Recap* [Video file]. https://www.youtube.com/watch?v=ykrNsx9Y-x0

Code Crew. (2016, October/November). *Inside Business Memphis, XI*(1), 34-35. https://issuu.com/contemporarymedia/docs/imb_oct-nov_2016

Computer Science Teachers Association. (n.d.). *View the CSTA K-12 Standards*. https://csteachers .org/k12standards/interactive/

Davis, S., Ravitz, J., & Blazevski, J. (2018, February 21). *Evaluating computer science professional development models and educator outcomes to ensure equity*. [Paper presentation]. Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT), Baltimore, MD, USA. https://doi.org/10.1109/RESPECT.2018.8491716

de Ruiter, L. E., & Bers, M. U. (2021). The coding stages assessment: Development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education, 32*(4), 388–417. https://doi.org/10.1080/08993408.2021.1956216

Fagerlund, J. Häkkinen, P., Vesisenaho, M., & Viiri, J. (2020). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education, 33*(1), 28-61. https://doi.org/10.1002/cae.22255

Gray, S. (2024, July 26). Codecrew hosts "Hack-a-thon." Action News 5. https://www.actionnews5.com/2024/07/26/codecrew-hosts-hack-a-thon/

International Society for Technology in Education. (2020). *ISTE Computational Thinking Competencies for Educators.* Retrieved January 30, 2025, from https://iste.org/standards/computational-thinking-competencies

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development.* Prentice-Hall.

Mılıć, M., Kukuljan, D., & Krelja Kurelović, E. (2018). Micro:Bit implementation in ICT education. *The Eurasia Proceedings of Educational and Social Sciences, 11*, 128-133. https://dergipark.org.tr/en/pub/epess/issue/40408/491201

Ni, L., & Bausch, G., & Benjamin, R. (2021). Computer science teacher professional development and professional learning communities: A review of the research literature. *Computer Science Education, 33*(1), 29-60. https://doi.org/10.1080/08993408.2021.1993666

Olsson, J., & Granberg, C. (2022). Teacher-student interaction supporting students' creative mathematical reasoning during problem solving using Scratch. *Mathematical Thinking and Learning, 26*(3), 278–305. https://doi.org/10.1080/10986065.2022.2105567

Purcell, J. H., Burns, D. E., & Purcell, W. H. (2021, October 4). A blueprint for Interest-Based Learning. ASCD. https://www.ascd.org/el/articles/a-blueprint-for-interest-based-learning

Resnick M., Maloney J., Monroy-Hernández A., Rusk N., Eastmond E., Brennan K., Millner A., Rosenbaum E., Silver J., Silverman B., & Kafai Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Stamatios, P. (2024). Can preschoolers learn computational thinking and coding skills with ScratchJr? A systematic literature review. *International Journal of Educational Reform, 33*(1), 28-61. https://doi.org/10.1177/10567879221076077

TN Code § 49-1-232. (2021). *Chapter 979 of the Public Acts of 2022,* https://publications.tnsosfiles.com/acts/112/pub/pc0979.pdf

About the Authors

Dr. Ayanna Perkins is an Instructional Designer at Carle Health and a part-time instructor and consultant at The University of Memphis. Her career focuses on CS education, teacher development, and research, to enhance student learning and promote diversity and inclusion. Ayanna's previous roles include Teacher Professional Development Facilitator at CodeCrew, where she developed training materials and workshops for over 200 teachers at CodeCrew. Her research explores assistive technology in virtual learning and justice-centered computing in K-12 education. She has presented at various conferences and holds a Doctor of Education in Instructional Design and Technology.

Elexis Allen is a CS Instructor at CodeCrew in Memphis, TN, with a biomedical engineering and data science background. She integrates technology and medical device knowledge into K-12 programming and software development curricula. She also developed and led the Healthy Bytes program, which bridges computer science and physical therapy through hands-on applications in coding, biomechanics, and data analysis. Allen is pursuing a master's degree in data science at Eastern University,

specializing in computational modeling and machine learning.

Kiyah Stokes is a graduate of the University of Memphis, where she earned a bachelor's degree in Programming and Web Development from the College of Professional Studies. Stokes possesses extensive experience as an instructor and is proficient in a variety of programming languages, including Java, Python, C#, and C++. She has further honed her expertise through leadership roles in initiatives such as STEM from Dance and Tech Wearables. In recognition of her exceptional contributions and leadership, she was honored as a Class of 2023 Sphero Hero by the Sphero Education Ambassador Program.

Danielle L. Jones is an EdD student in the University of Florida Computer Science Education Ed.D. Program. She has a research interest in computing education within early education and pair programming. Danielle has 10+ years of experience in Software development and 7+ years of experience in computing education.

Sharing & Modification Permissions