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.

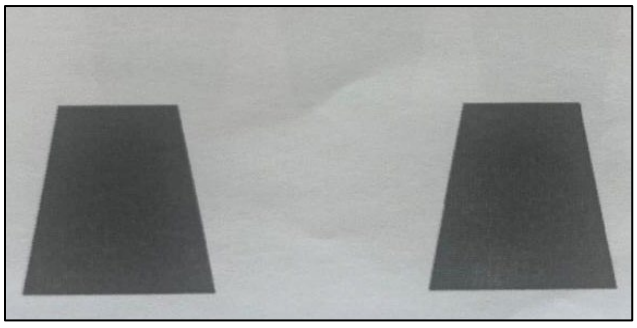

Figure 1. First cup-stack challenge. Note: Image reproduced with attribution under Creative Commons BY-NC-SA 4.0 license.

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.

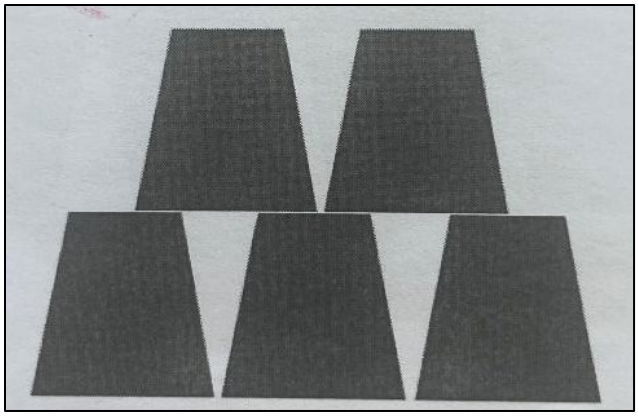

Figure 2. Second cup-stack challenge. *Note: Image reproduced with attribution under Creative Commons BY-NC-SA 4.0 license.*

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.

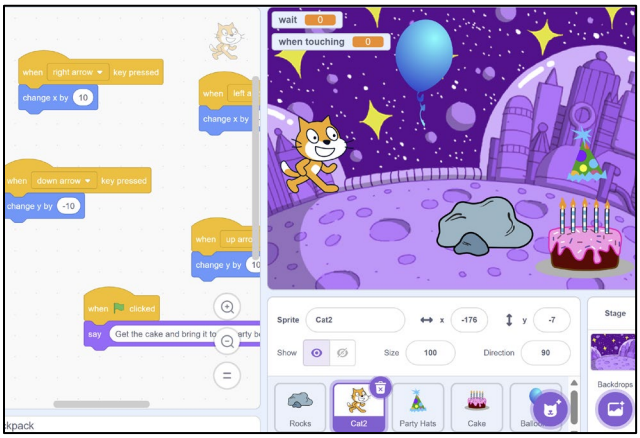

Figure 4. Example of student project.

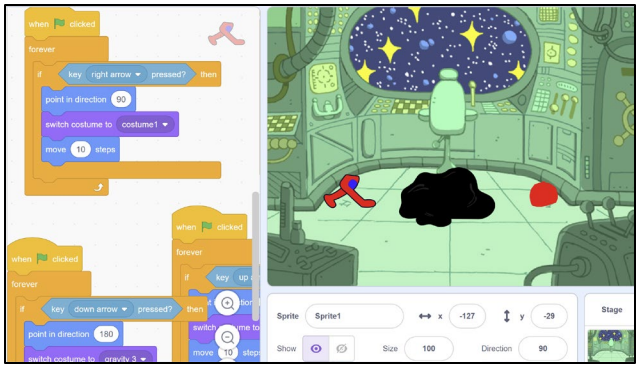

Figure 5. Example of student project.

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.

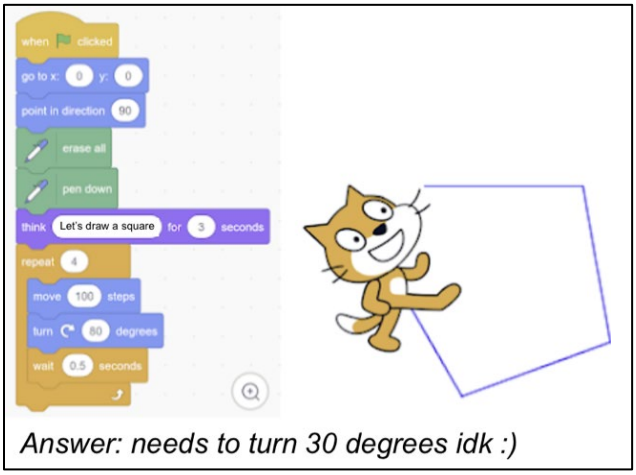

Figure 6. Student response to an open-ended question.

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