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.

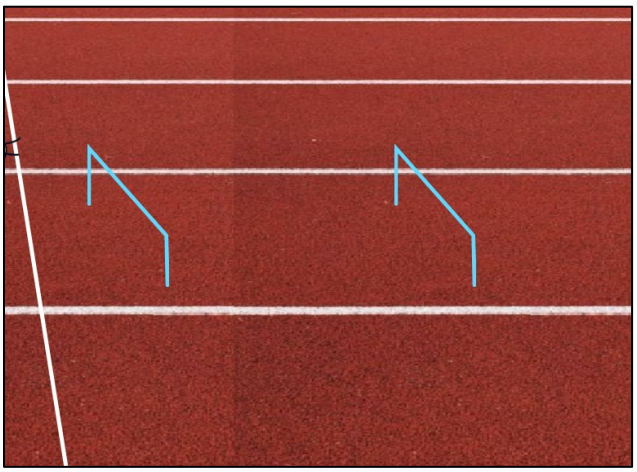

Figure 1: Hurdles

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.

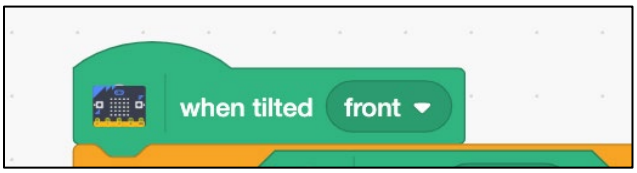

Figure 2: The “When Tilted” Block

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.

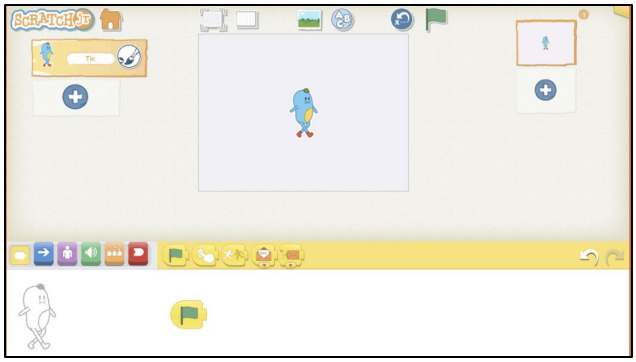

Figure 4: The Initial Project Interface of ScratchJr

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.

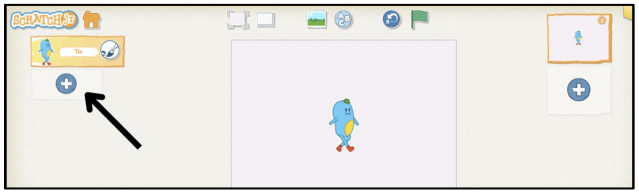

Figure 5: Add Sprite Button

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.

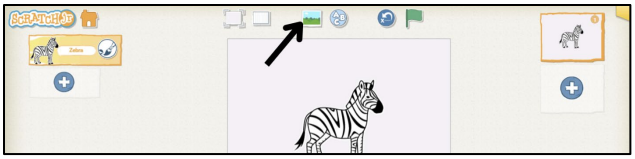

Figure 6: Add Background Button

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.

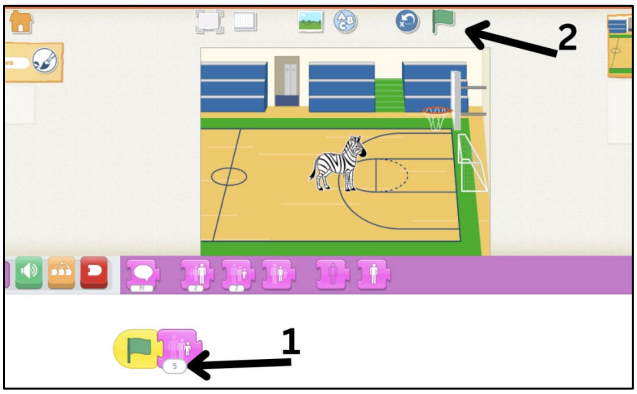

Figure 7: Sprite Shrink Program

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.

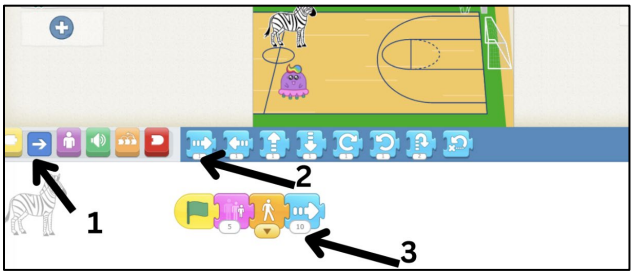

Figure 8: Sprite Race Program

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.

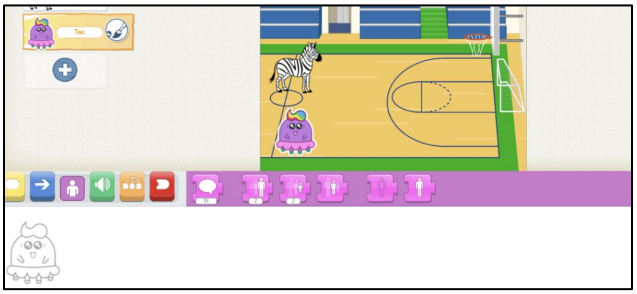

Figure 9: Outline of Selected Sprite

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