2.2

Coding

Coding in the K-12 Classroom

Jennifer Alexiou-Ray, Cassie Raulston, Diana Fenton, & Sherri Johnston

STEM Coding Programming

Learning Objectives

- Understand why coding is important in K-12 classrooms;
- Recognize the relationship between coding and important standards;
- Complete challenges and access resources associated with coding.

Computers are a part of almost every industry and are changing the way we live. Equally, this is true in education because computers and technology are used to deliver curriculum and offer opportunities for students to interact with technology. To better understand the role of computing in the world, we need to dive deeper and educate students on the basics of computer science.

Most students are taught only how to use computer and web-based software, rather than creating ways for them to use computational thinking and computer programming to solve authentic problems. One dimension of computer science that can easily be integrated into the curriculum is teaching students to code.

This chapter will provide preservice teachers with an overview of coding and how it is being taught in the K-12 setting.

Key Terms

Algorithm
a set of steps that are used to complete a task

Algorithmic Thinking
breaking problems down into smaller workable parts

Block-based Coding
coding with a programming language where the instructions are represented in blocks

Coding
a language that a computer can use to complete a task or a set of instructions

Computational Thinking
a problem solving process; typically broken down into decomposition, pattern recognition, abstraction, and algorithm design

Computer Language
structured commands written for a computer to process; some of the most common include JavaScript, Python, Structured Query Language (SQL), C, C++

Debugging
the process of testing, finding, and solving errors in computer programs

Pair-programming
two students sit at one computer, one is the “navigator” and one is the “driver”

Pseudocode
informal or simplified programming language that can be used to represent algorithms outside the computing environment

Unplugged
a coding lesson that does not require a computer

Why is coding in the K-12 classroom important?

Computers are a part of almost every industry and are changing the way we live. Equally, this is true in education because computers and technology are used to deliver curriculum and offer opportunities for students to interact with technology. To better understand the role of computing in the world, we need to dive deeper and educate students on the basics of computer science.

Most students are taught only how to use computer and web-based software, rather than creating ways for them to use computational thinking and computer programming to solve authentic problems. One dimension of computer science that can easily be integrated into the curriculum is teaching students to code.

Coding is simply a language that a computer can use to complete a task. It is a set of instructions given to a computer. “Coding is a new literacy. To thrive in tomorrow’s society, young people must learn to design, create, and express themselves with digital technologies” (Berkman Klein Center, 2014). According to the Institute of Museum and Library Services (2014), 65 percent of scientists with advanced degrees started being interested in the sciences started before

middle school. Adams and Mowers (2013) also indicate that coding skills are needed at all stages in life because it is a global language and more commonly used than spoken languages like English, Chinese or Spanish.

Research shows a growing need for coding in the job market. Computing jobs are growing at twice the national average while only 2.4 percent of college graduates with a degree in Computer Science (Adams & Mowers, 2013). As a result of this growing need, coding is now recognized as a critical literacy skill in the CS field, which people use to solve problems. Even outside of the CS field, many new jobs require some knowledge of computing and/or coding. Coding requires complex thinking and has evolved from solely technical skills to organize and communicate ideas (New Media Consortium, 2015). States such as Texas and Florida are allowing coding classes to count as a foreign language credit in high school (Zinth, 2015).

Offering computational thinking strategies with coding as part of the curriculum in elementary schools, K-5 students will have the exposure to 21st century skills that enable them to be successful in their future careers. In addition, research indicates that coding provides cognitive advantages to learning.“Learning a system of signs, symbols, and rules to communicate - that is, language study - improves thinking by challenging the brain to recognize, negotiate meaning and master different language patterns.” (Adams & Mower, 2013).

Coding involves problem-solving, perseverance, collaboration, mathematical logic, and reasoning skills. Could anyone argue that these are not skills that we want for all students? It can be easy to get caught up in the common barriers of why not to code, such as lack of time, lack of knowledge, or lack of resources. But coding can be integrated into any curriculum and the learning can take place along with the students. There are many low or no-cost solutions to teaching students to code.

Educators need to first start with the belief that coding is for everyone and everyone needs to learn coding. Teaching coding is more than teaching the language for computing. It is not a language like ours, with vocabularies or alphabets, but special commands and abbreviations that are used to write computer software. Some of the most common coding languages include JavaScript, Python, Structured Query Language (SQL), C, C++. Coding can be thought of as a small part of the computer science field, which also involves physical systems, networks, storage, and collection of data or coding can be taught to reach more overarching goals conceptualizing the student as a global citizen and learner.

Coding and its relationship to ISTE Standards

Coding touches on most of the ISTE (International Society for Technology in Education) Standards for Students (ISTE, 2016). There are seven ISTE standards for students of a digital world. They are *Empowered Learner, Digital Citizen, Knowledge Constructor, Innovative Designer, Global Collaborator, Computational Thinker, and Creative Communicator.* These standards have been adopted across the country and world. Coding is a powerful tool we can use to cover more than just ISTE standards. Sylvia Duckworth, a world renown sketchnoter, summed up the ten reasons to teach coding from Aspinall (2015) perfectly.

Empowered Learner

When students code they become *empowered learners* . When a program is being developed, students set goals for program completion, develop strategies to “debug” or find mistakes in their program, and reflect on their work in order to make their program better by adding additional code. They also are able to share and receive feedback on their work. During the coding process students troubleshoot and use the knowledge gained to explore other emerging technologies as a result of understanding the logic behind how code is organized and executed.

Digital Citizen

Many times during a coding project students need assistance to troubleshoot their program. Students will turn to other programs or online documentation to assist them with their problem and program completion. In this case, students

must respect the intellectual property rights of others by giving proper credit. This is following the “Digital Golden Rule” and demonstrating digital citizenship.

Knowledge Constructor

Many students will code an app in order to solve a real-world problem. They will use research strategies and evaluate the accuracy of the information they acquire. Soon after, they will develop an idea and use their coding to seek a solution. These students are *knowledge constructors*. They are creating meaningful learning experiences for themselves.

Creative Communicator and Innovative Designer

As students create their programs they take into account the objectives of their product and work on expressing their ideas clearly and effectively. These coders go through the design process for generating ideas a number of times, test their theories, and refine their prototypes. As they accomplish their task, they use a variety of digital objects to create games, visualizations, models, or simulations. These students are blossoming into *creative communicators and innovative designers* .

Computational Thinker

The last ISTE standard coding meets is the one that is most obvious. That is the standard of *computational thinker*. When a child codes they are using algorithmic thinking, breaking problems down into smaller workable parts, and becoming adept at developing sequences of detailed steps to create and test their work. As students code, they begin to understand how computers execute their programs and what is involved in the production of the apps and they use daily.

What coding looks like in the classroom

Basic Coding Concepts

While the idea of teaching coding may seem overwhelming and complex, basic coding lessons do not even need to start with a computer. Coding lessons can progress from simple to complex. A lesson that does not require a computer is often termed “unplugged” because computer technology is not needed. A basic unplugged lesson might include teaching the students the vocabulary of coding. For example, writing an algorithm is one of the fundamental concepts in coding. An algorithm is a set of steps that are used to complete a task. Using the “unplugged” philosophy, students can write algorithms of their routine at school to learn this concept.

Challenge #1

Write an algorithm that consists of the steps you take to get ready each morning. What do you do first, second, and third? You can write this algorithm as if you were talking to another person, no need to use any “code” for now!

Progression from “unplugged” to “plugged” lessons with computers might start with simple coding programs. Simple coding programs for students are in friendly, graphical format with the more complex language hidden. For example, Blockly is a visual coding editor where students move around blocks to build code. The hidden language in Blockly is JavaScript. Blocks can be dragged and dropped and edited for simple programming in a puzzle like appearance.

When teaching “plugged” coding, computer availability might be a concern. To reduce the number of computers needed, a strategy that is considered a best practice when coding is the use of paired programming, which “is a proven method for both enhancing learning and writing better code” (Kraus and Prottsman, 2017). Two students sit at one computer, one is the “navigator” and one is the “driver.” Then, at some point during the activity the two roles switch, which allows students experience with both roles and hands-on computer time.

Challenge #2

Try a basic Blockly app.

For many coding games or apps, once the blocks are arranged, a program can be run to execute various commands. Many block type programs that are free are available to educators.

Challenge #3

Now try Blockly again. This time you will be navigating your way through a maze.

As you can see with this previous challenge, lessons or time writing code can be spread out over multiple class periods with the recognition that some students might advance faster than others and multiple levels of programming should be available.

One resource that allows for this progression is Code.Org. Code.Org not only offers free training for educators but also plugged and unplugged lessons that educators can take and immediately use in the classroom. The free, web-based platform is where students can utilize Blockly to progress from simple to more complex coding concepts and teachers can track student proficiency and progress. It is designed with multiple courses that cater to different student age groups and reading proficiencies that help teach coding at a developmentally appropriate level.

Challenge #4

Visit the Code.org Course Catalog to see what options are available for the students you are planning to teach.

Coding with Scratch

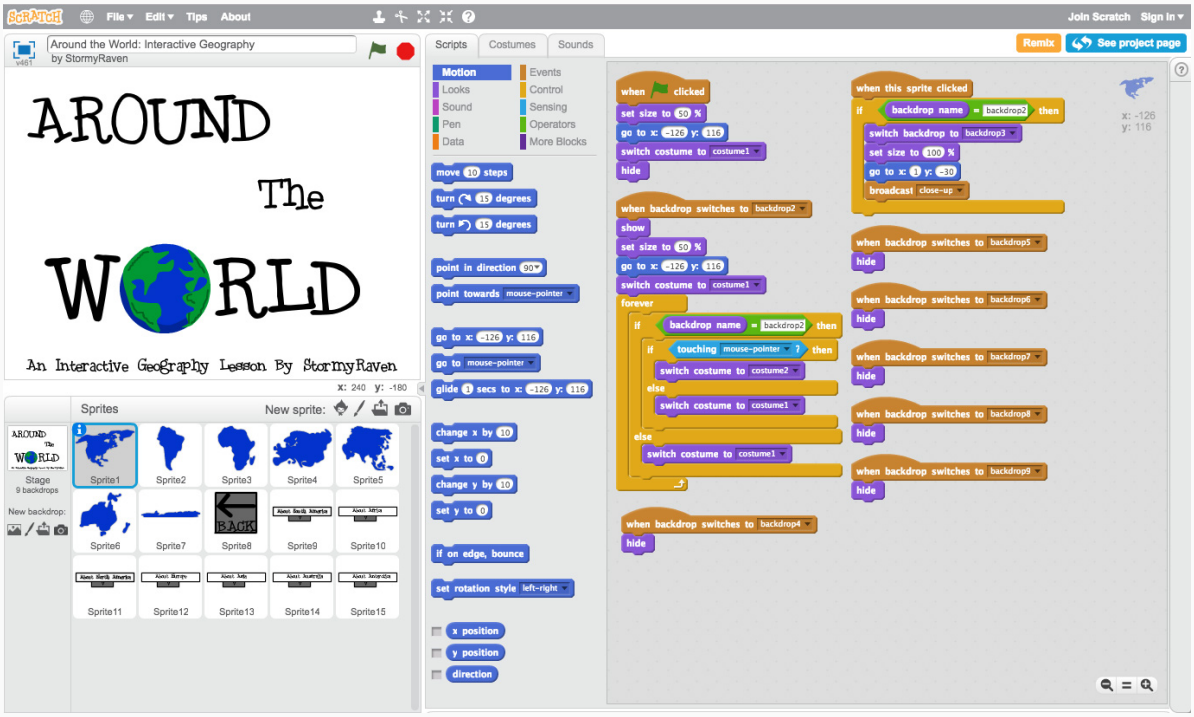
Watch on YouTube

Challenge #5

Visit Scratch and recreate some of these subject-based projects.

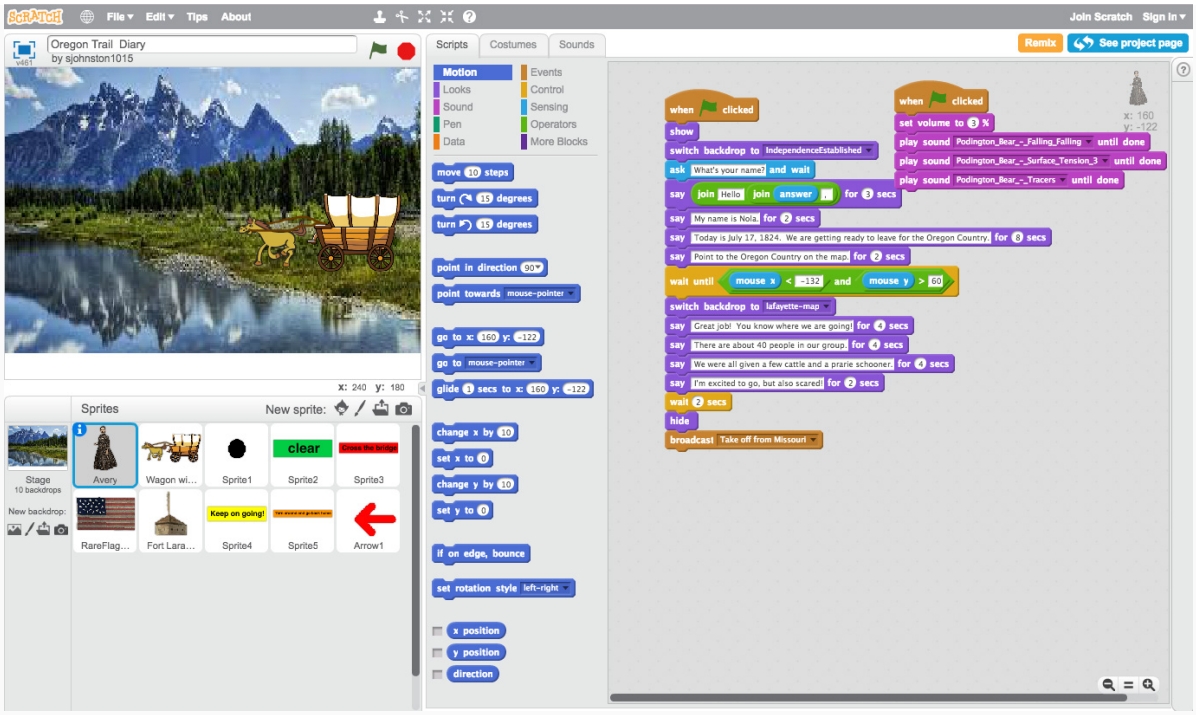
Geography

[Around the World Interactive Geography]



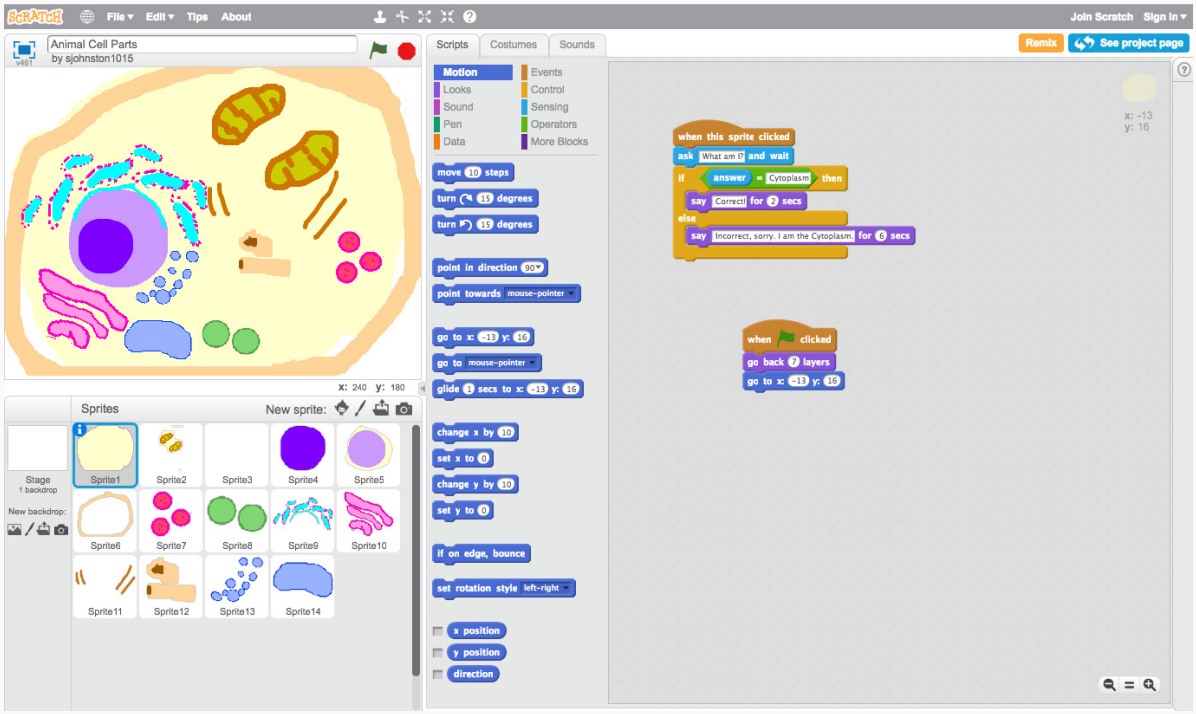
Social Studies

[Oregon Trail Interactive Diary]



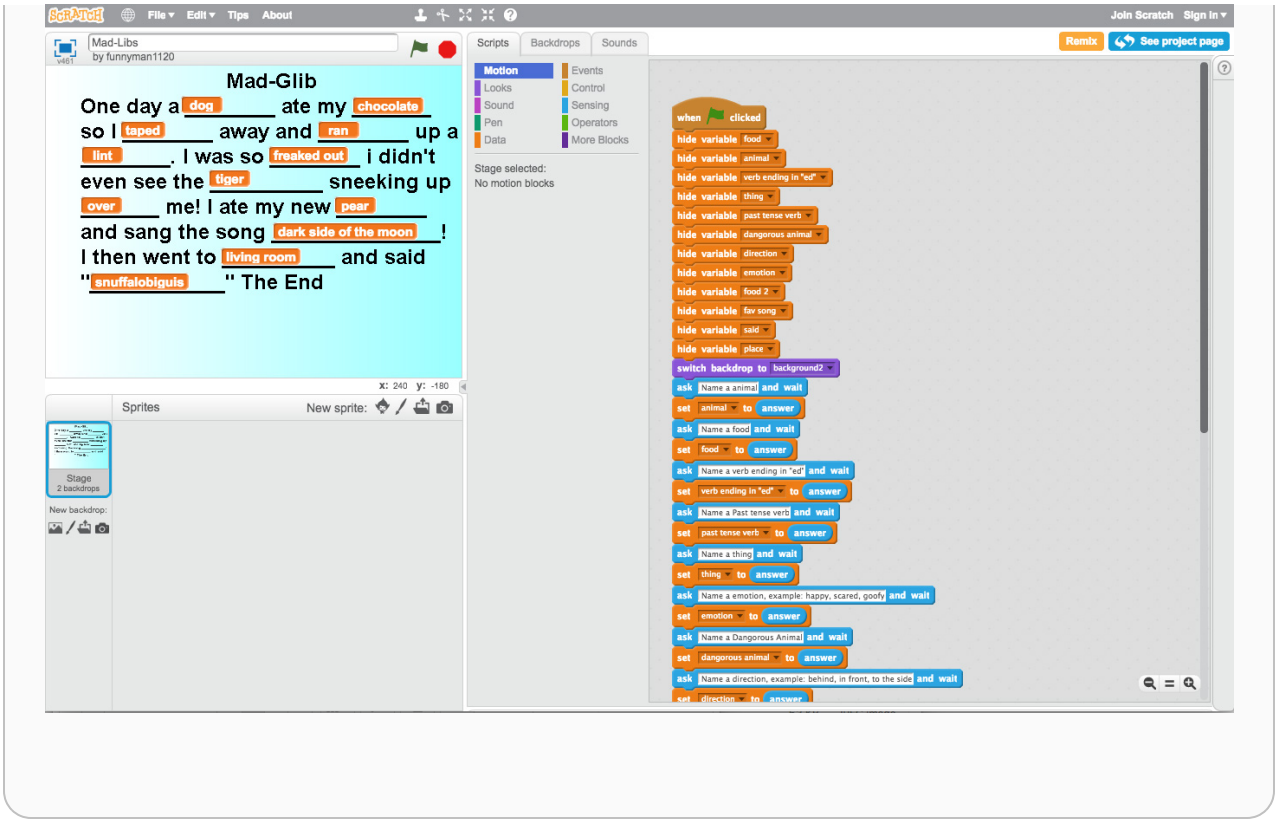
Science

[Animal Cell Parts]



Language Arts

[Mad Libs Story]



Coding resources and additional study

In addition to the resources you have explored in this chapter, there is an ever-growing list of tools and resources a teacher can use to aide in the integration of coding across the curriculum. Much of what is currently being used in education focuses on teaching students coding concepts in a block-based coding environment, like Blockly. However, some tools give students the option of learning simple coding languages or even beginning with block-based coding and moving to a coding language.

Low or No-Cost Coding Tools for the Classroom

- Computer Science Classes by Code.Org (free, ages 5-15)
- Scratch Coding Cards
- 20 Makey Makey Projects for the Evil Genius
- Scratch - (free, ages 8-14)
- The Foos by codeSpark -free, ages 4-8)
- Kodable - (free, ages 5-18)
- Littlecodr - (card game)
- Code Monkey Island - (board game)

Children's Books to Encourage a Love of Coding

- Computational Fairy Tales
- Hello Ruby: Adventures in Coding
- A is for Array
- Kids Get Coding: Learn to Program
- Coding for Kids by McCue

Computer or Mobile Device Assisted Games

- Bloxels - (tactile video game design)
- Makey Makey -(making/programming)
- Osmo Coding Awbie
- Osmo Coding Jam

Web-based Software for Coding

- Tynker - (fee-based ages 8-13)
- Code Monkey - (online--paid solution to transition to writing code)

Robots for Hands-on Coding Application

- Edison - (programmable robot)
- Dash & Dot - (programmable robots)
- Hummingbird Robotics
- Sphero - (programmable robot)
- Ozobots -(programmable robot)

Coding Resources for Professional Study

- Code.org Workshops
- Code: The Hidden Language of Computer Hardware and Software by Petzold
- Programming in the Primary Grades: Beyond the Hour of Code by Patterson
- Code in Every Class by Brookhouser and Megnin
- Girls Who Code: Learn to Code and Change the World by Saujani
- No Fear Coding: Computational Thinking Across the K-5 Curriculum by Williams
- Invent to Learn: Making, Tinkering, and Engineering in the Classroom by Martinez & Stager
- Computational Thinking and Coding for Every Student: The Teachers Getting Started Guide by Krauss & Prottsman
- Guide to Teaching Computer Science: An Activity-Based Approach by Hazzan, Lapidot, & Ragonis

Additional Readings and Resources for Professional Study

- Most Likely to Succeed: Preparing Our Kids for the Innovation Era
- Drive the Surprising Truth about what Motivates Us
- Mindset: The New Psychology of Success

References

Adams, A. & Mowers, H. (2013, October 3). Should coding be the "new foreign language" requirement? [Blog post]. Retrieved from http://www.edutopia.org/blog/coding-new-foreign-language-requirement-helen-mowers

Aspinall, B. (2015, May 26). 10 reasons to teach coding [Blog post]. Retrieved from http://brianaspinall.com/10-reasons-to-teach-coding-sketchnote-by-sylviaduckworth/

Berkman Klein Center. (2014, July 10). 21st century literacy: New initiative makes the case that learning to code is for everyone [Blog post]. Retrieved from https://cyber.harvard.edu/node/95731

Institute of Museums and Library Services. (2014, June). Talking points: Museums, libraries, and makerspaces. Retrieved from https://www.imls.gov/assets/1/AssetManager/ Makerspaces.pdf

International Society for Technology in Education. (n.d.). ISTE standards for students. Retrieved from http://www.iste.org/standards/for-students

New Media Consortium. (2015). NMC horizon report: 2015 K-12 edition. Retrieved from http://cdn.nmc.org/media/2015-nmc-horizon-report-k12-EN.pdf

Shrock, K. (2017, August 12). Computational and design thinking. Retrieved from http://www.schrockguide.net/computational-thinking.html

Zinth, J. (2015, April). Computer science in high school graduation requirements. Retrieved from https://www.ecs.org/clearinghouse/01/18/29/11829.pdf

Jennifer Alexiou-Ray

University of Montevallo

Dr. Jennifer Alexiou-Ray is an Associate Professor and Program Coordinator for Leadership and Technology Programs at the University of Montevallo. She has degrees in Educational/Instructional Leadership and Instructional Technology. Dr. Ray's research interests include coding/robotics, social media in the classroom, flipped learning, professional development, and mobile learning. Recently, she received the prestigious International Society for Technology in Education's "Making it Happen" award for excellence and leadership in Educational Technology from the Alabama Leaders for Educational Technology. Additionally, she works with K-12 students throughout the year helping to introduce coding/robotics into school curriculum and through summer camp programs.

Cassie Raulston

University of Montevallo

Dr. Cassie Raulston is an Associate Professor for the Teaching, Leadership, and Technology programs at the University of Montevallo. She has degrees in Elementary/Special Education, and Instructional Leadership/Technology. Dr. Raulston's research interests include mobile learning, technology integration, coding, and gifted education. She presented a 3-hour hands-on workshop "Using Scratch and Robotics to Support Multiple Literacies in the Classroom" at the International Society for Technology in Education conference in Chicago in 2018. In addition, she has the privilege of collaborating with educators to develop and direct curriculum and multiple summer Coding Camps for students.

Diana Fenton

St. John's University

Dr. Diana Fenton is an Assistant Professor in the Education Department at the College of St. Benedict and St. John's University. She has degrees in science and education leadership. Dr. Fenton's research interests include science pedagogy, technology integration with coding, robotics and 3D printing. She works closely with pre-service teachers to provide opportunities to implement lessons with coding and computational thinking with K-12 students. Recently, she presented "An Algorithm to Successful Computer Science Elementary Implementation" at International Society for Technology in Education conference in Chicago in 2018.

Sherri Johnston

Clovis Unified School District

Sherri Johnston is the Coordinator of Educational Technology and Professional Development for the Clovis Unified School District. She has over 20 years experience in education and holds degrees in Elementary Education and Curriculum and Instruction/Educational Technology. She has extensive experience with computer science in the elementary and Jr. high school environments. Sherri is a CODE.org Computer Science Discoveries Facilitator, a Google for Education Certified Trainer, Microsoft Innovative Educator Trainer, Common Sense Digital Citizenship Ambassador. She has presented at EdTech Summits, National CUE Conferences, and the International Society for Technology in Education conference in San Antonio in 2017.