feature article

recommendations to support computational thinking in the elementary classroom

Computational thinking is an important and necessary way of thinking for computer programmers and other professionals in STEM.

Mobile applications intended to provide exposure to the concepts of computer programming and coding, referred to as coding apps, are becoming increasingly recognized as useful tools for classroom instruction (Hutchison, Nadolny, & Estapa). For example, the ScratchJr app provides opportunities for users to create stories, games, and animations through visual coding and, as a result, experience what it is like to be a computer programmer. These programming apps can be used to expose students not only to computer programming or coding, but they also teach mathematics concepts and the broader skills associated with computational thinking by asking students to engage in tasks that require them to do things such as group variables, apply conditional logic, develop algorithmic functions, calculate angles within geometric shapes, and more. Computational thinking is described as a problem-solving process and can be defined as follows:

> Formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implement-

by
Anne Estapa,
Amy Hutchison,
and
Larysa Nadolny

Figure 1. Example of snap coding from ScratchJr.

> ing possible solutions with the goal of achieving the most efficient and effective combinations of steps and resources; and generalizing and transferring this problem-solving process to a wide variety of problems (Society of Technology in Education [ISTE] and the Computer Science Teachers Association. [CSTA]) (Israel, Pearson, Tapia, Wherfel, & Reese, 2015, p. 263).

Computational thinking is an important and necessary way of thinking for computer programmers and other professionals in science, technology, engineering, and mathematics (STEM). Research on emerging practices around computational thinking that is developed through coding initiatives in schools reports that elementary children typically learn how to operate technologies rather than learn how to develop new technologies (Israel, et al., 2015). As a result, students in elementary schools experience only the receiving end of technology (Burke & Kafai, 2014). This lack of production potentially limits the effectiveness of technology integration since early experiences with computational thinking as a means of problem solving in abstract ways has the potential to improve attitudes, engage students, and enhance programing skills (Israel, et al., 2015). Thus, it is important to provide students with early exposure to computational thinking. Yet, with so many apps and so little guidance, it can be difficult to know how to integrate these apps into classroom instruction. Therefore, the purpose of this article is to provide recommendations for teachers, drawn from research, on how to select apps and begin practices that support computational thinking.

Recommendation #1: *Select Computationally Rich Coding Apps*

To ensure that an app is appropriate for all learners within the classroom, the idea of "low ceiling, high ceiling" should be a guiding principle. Grover and Pea (2013) explain that computational thinking tools (coding apps) for elementary students should be easy for beginners to start an activity and create programs or codes (low ceiling). However, the tool should also be powerful and extensive enough to satisfy the attention and learning of more experienced or advanced programmers (high ceiling). Apps with this principle in mind often follow a use-modify-create progression to allow a learner to experience each stage to support learning and engagement. Therefore, in reviewing apps for implementation within the classroom, the authors recommend that teachers select apps that allow students to increase their engagement and production with the app as their skill increases.

Grover and Pea (2013) highlight the following apps as examples that allow early experiences to focus on designing and creating: Scratch, Alice, Kodu, and Greenfoot. Many of the apps provided use a visual programming language, which allows programmers to snap visual programming codes together to control actors on the screen. This format supports computational thinking and provides students with the opportunity to create their own digital media products. Yet, it is simple enough that beginning users can be successful with the apps. The authors highlight this process in Figure 1, with an example from ScratchJr. In this example, the student selects a series of commands and places them in a logical sequence to make the animals move around the barn. Further, this app allows for the addition of a recorded speech response (represented by the microphone) that plays as the movement on the screen occurs. This example shows how simple it is to navigate a coding app such as ScratchJr and apply computational thinking skills (low ceiling). Yet, the app also provides opportunities for students to develop and apply more complex computational thinking by creating original characters, developing and connecting multiple scenes, changing colors and words, etc. (high ceiling).

Recommendation #2: *Become a Learner*

For teachers to effectively integrate coding apps into mathematics instruction it may be helpful to first engage with these apps as a learner. Some teachers may believe that coding is too difficult to learn or too far outside the realm of their expertise. However, coding apps and many coding initiatives are designed for beginners and require no previous coding experience. Many apps are designed with a game-like format or simple tutorials that teach the user what he or she needs to know to engage in the activities presented within the app. By engaging with coding apps as a learner, teachers can gain experience with the apps while also determining the specific concepts that can be taught through the app.

There are many popular apps and websites with which users can try to gain a better understanding of the function and purposes of coding apps. The authors recommend that teachers get started with Scratch or ScratchJr, depending on their own skill level and the grade level they teach. Scratch and ScratchJr (scratch.mit.edu) are both free and allow users to create animations, art, games, stories, or more. Scratch is targeted at ages 8-16 and allows users to program their own content, but also has an online community in which teachers can engage to get resources and ideas for integrating Scratch into their classrooms.

ScratchJr is targeted at younger students, ages 5-7, and is a great tool for those who are inexperienced with coding apps. Both apps teach computational thinking, as they require students to apply conditional logic and solve problems to get the outcome they want—creation of a game, image, animation, etc. Similarly, many coding apps require the application of mathematics skills such as group variables, applying conditional logic, developing algorithmic functions, and calculating angles within geometric shapes. Teachers can consider how these skills can be taught through the apps as they explore them for themselves.

Recommendation #3: *Use Apps for Active Learning*

Student content creation within a coding app can meet the needs of learners in several ways. The 2016 National Education Technology Plan (U.S. Department of Education, 2016) asks all educators to consider equity in the use of classroom technology, particularly considering differences in passive or active learning through technology. Differences in the way technology is used in the classroom for either more active creation with digital content and tools or more passive consumption of information from digital devices has been termed the "digital use divide" (U.S. Department of Education, 2016). By engaging all students in the active, creative use of coding apps, the teacher is helping to bridge the digital use divide. In addition to classroom activities, the authors recommend providing students, parents, and/or guardians with additional online resources to encourage engagement with groups underrepresented in the STEM fields. For example, Black Girls Code (www.blackgirlscode.com/) was created by Kimberly Bryant in partnership with major corporations in the fields of technology and finance. Students can attend workshops, join afterschool communities, and participate in hackathons across the nation. Girls who Code (https://girlswhocode.com/) hosts summer camps and afterschool clubs. If online resources do not meet the needs of students, teachers can consider starting a coding club using the free resources at Code Academy (www.codecademy.com/schools/curriculum/resources) or the ready-made lesson plans for afterschool clubs using the Tynker app (see Figure 2).

Example of a coding app.

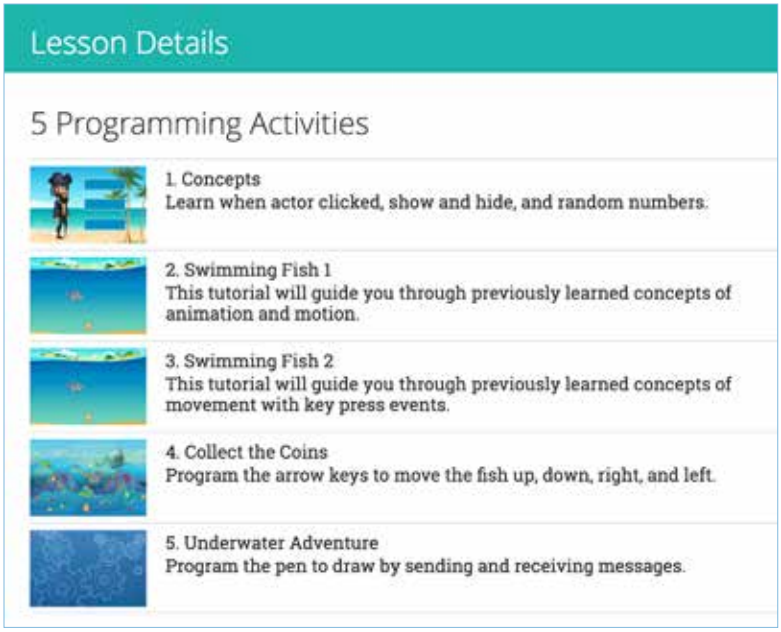

Figure 2. Sample lesson activities from Tynker.com.

Recommendation #4: *Bridge Learning Across the Disciplines*

Recently, several researchers have illustrated how concepts of computational thinking can be aligned with other content areas to provide authentic learning experiences (e.g., Jona, et al., 2014; Sengupta, et al., 2013; Weintrop, et al., 2014). Advocates of computational thinking contend that computational thinking is at the core of all STEM disciplines (Henderson, Cortina, Hazzan, & Wing, 2007) and has the potential to bridge learning within and across discipline areas. Importantly, coding apps can be used to help students begin thinking like scientists, mathematicians, or engineers. For example, coding apps can be used to develop what Lucas and Hanson (2014) refer to as Engineering Habits of Mind (EHOM), which include: (1) systems thinking, (2) adapting, (3) problem-finding, (4) creative problem solving, (5) visualizing, and (6) improving. For instance, as part of a science lesson, teachers could ask students to create an animated demonstration of the life cycle of a butterfly using a coding app or explore the topic of adaptations (Figure 3). As part of that process, teachers could also teach and integrate engineering habits of mind such as creative problem-solving

Butterfly Coding Challenge

National Science Education Standards: K-4 The Characteristics of Organisms: Each plant or animal has different structures that serve different functions in growth, survival, and reproduction.

Now that you are familiar with how some butterflies use camouflage or a disguise to hide themselves from predators, it is time to help your own butterflies survive!

1. Choose two butterflies from a botanical garden website, such as http://rgbutterflyapp.com/ or www.missouribotanicalgarden.org/
2. Download images of your selected butterflies to your iPad.
3. Follow the same steps above to find and download a picture of a predator of butterflies.
4. Create a background in your coding app that will help hide those butterflies.
5. Using the sequence and looping tools in your coding app (control and motion in Scratch), move the butterflies and the predator, showing how a butterfly can survive by using its adaptations.

Extension

Turn your story into a survival game! Use controls and variables to allow the player to earn points when the predator touches the butterfly. For example, when the space bar is clicked, the wasp will move four steps in a random direction until it touches the butterfly.

Figure 3. Buterfly Coding Challenge

(EHOM 4) by having students generate coding and design solutions together and then by adapting (EHOM 2) their code and design to improve (EHOM 6) their demonstration.

Further, engaging with coding apps can also help students develop digital literacy skills and exposure to disciplinary vocabulary by introducing them to specialized language and opportunites to create and produce new information in digital contexts (Hutchison, Nadolny, & Estapa, 2016). Through the use of coding apps students can learn coding skills ranging from basic to complex, can learn how to devise and communicate effective messages through a combination of images, text, and color. Further, students will gain experience that will support their development towards proficiency with the International Society for Technology in Education's *ISTE Standards for Students* (2016), such as becoming computational thinkers and creative communicators.

To maximize learning when implementing coding apps into the classroom, teachers should begin by connecting the mathematical content learning within the app to one other discipline, building connections one content area at time. This will ensure that efforts are purposeful and that students will be shown the connection among the STEM disciplines. For example, when working on an app focused on computational thinking goals, through problem solving and representing data using graphs and/or tables (Mathematics) students could also engage in conversations around patterns in coding (Technology), create stories to provide context for what is happening on the screen (Literacy), or recreate a code using classroom materials to design and redesign paths given specific criteria (Engineering). In this way, the learning experience connects student understanding within and across STEM disciplines as recommended within *Next Generation Science Standards* (*NGSS*). In Table 1, the authors highlight how a computational thinking coding experience might align with *NGSS* (NGSS Lead States, 2013).

Through the integration process, the lesson or activity implemented supports student learning within and across STEM content areas.

Conclusion

The authors support claims that early access to and experiences with computational thinking will strengthen elementary students' computational thinking abilities while enhancing their understanding of mathematics and the connection of mathematics to other disciplines. In defining computational thinking as a way for

Table 1. *NGSS* K-2 Engineering Design Standards

Performance Expectation	Ask questions, make observations, and gather information about a situation people want to change to define a simple problem that can be solved through the development of a new or improved object or tool.
Science and Engineering Practices	Ask questions based on observations to find more information about the natural and/or designed world(s). (K-2-ETS1-1)
Disciplinary Core Idea	A situation that people want to change or create can be approached as a problem to be solved through engineering. (K-2-ETS1-1) Before beginning to design a solution, it is important to clearly understand the problem. (K-2-ETS1-1)

students to not only use computers to solve problems but also as a means to create and represent model solution strategies, student learning reaches beyond programming. As teachers explore options and purposefully integrate apps into their classroom following the recommendations in this article, students will be provided with the opportunities and tools they need to learn. The interest generated from such experiences has the potential to prime students for success within the classroom and in future computational-thinking-based opportunities.

References

Burke, Q. & Kafai, Y. B. (2014). Decade of game making for learning: From tools to communities. *Handbook of Digital Games,* pp. 689-709.

Code.org. (2014). *2014 annual report.* Retrieved from http://code.org/about/2014

Grover, S. & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Henderson, P. B., Cortina, T. J., & Wing, J. M. (2007, March). Computational thinking. In ACM *SIGCSE Bulletin, 39*(1), 195-196.

Hutchison, A., Nadolny, L., & Estapa, A. (2015). Using coding apps to support literacy instruction and develop coding literacy. *The Reading Teacher*, 1-11.

International Society for Technology in Education. (2016). *ISTE standards for students.* Retrieved from www.iste.org/standards/standards/for-students-2016

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education, 82*, 263-279.

Jona, K., Wilensky, U., Trouille, L., Horn, M., Orton, K., Weintrop, D., et al. (2014). *Embedding computational thinking in science, technology, engineering, and math* (CT-STEM). Presented at the Future Directions in Computer Science Education Summit Meeting, Orlando, FL.

Lee, Y. J. (2011). Empowering teachers to create educational software: A constructivist approach utilizing Etoys, pair programming and cognitive apprenticeship. *Computers & Education, 56*(2), 527e538. Retrieved from http://dx.doi.org/10.1016/j.compedu.2010.09.018

Lucas, B. & Hanson, J. (2014). Thinking like an engineer: Using engineering habits of mind and signature pedagogies to redesign engineering education. *International Journal of Engineering Pedagogy, 6*(2), 4-13.

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior, 41*, 51-61.

NGSS Lead States. (2013). *Next generation science standards: For states, by states.* Washington, DC: National Academies Press.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*, 351e380. Retrieved from http://dx.doi.org/10.1007/s10639-012-9240-

U.S. Department of Education, Office of Educational Technology. (2016). *Future ready learning: Reimagining the role of technology in education.* Washington, DC. Retrieved from http://tech.ed.gov/files/2015/12/NETP16.pdf

Weintrop, D., Beheshti, E., Horn, M. S., Orton, K., Jona, K., Trouille, L., et al. (2014). *Defining computational thinking for science, technology, engineering, and math.* Poster presented at the Annual Meeting of the American Educational Research Association (AERA 2014), Philadelphia, PA.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-36.

Anne Estapa *is an assistant professor in the School of Education at Iowa State University. Her research focuses on the facilitation of teacher noticing and learning through teaching practices, technologies, and professional development opportunities. She can be reached at aestapa@iastate.edu.*

Amy Hutchison *is an associate professor at George Mason University. Through her scholarship she examines how digital technology can be used equitably to support diverse learners and ways of supporting the development of STEM literacy among underrepresented students.*

Larysa Nadolny *is an assistant professor in the School of Education at Iowa State University. Her research interests include examining factors in student motivation and achievement within digital environments.*

This is a refereed article.

Ad Index