Girls Build Excitement for Math

To increase digital literacy and mathematical fluency, students electronically piece together details of computer coding by creating and animating sprites.

Julie M. Amador and Terence Soule

Before this unit about computer programming, I didn't know what it meant to code and I didn't know it required math. Now, I may consider computer coding when I go to college.

Marley had just completed a weeklong program for middle school girls that focused on computer coding, with an aim toward increasing both mathematical understanding and conceptual reasoning through the integration of mathematics and technology. Middle school girls, including Marley, spent five days learning how to design, create, and animate sprites, which are computer graphics manipulated as a single entity (see **fig. 1**). They learned about vectors as a way to describe the velocity of falling objects and were able to work collaboratively to manipulate animated characters in specific directions as they interacted with other sprites in a virtual realm. Marley's work is one of several creations made by middle school girls over the course of a thirty-hour computer-coding unit focused on advancing mathematical understanding and spatial reasoning. A library of these computer coding projects can be found at http://www2.cs.uidaho.edu/~tsoule/codecamppages/.

THE IMPORTANCE OF CODING

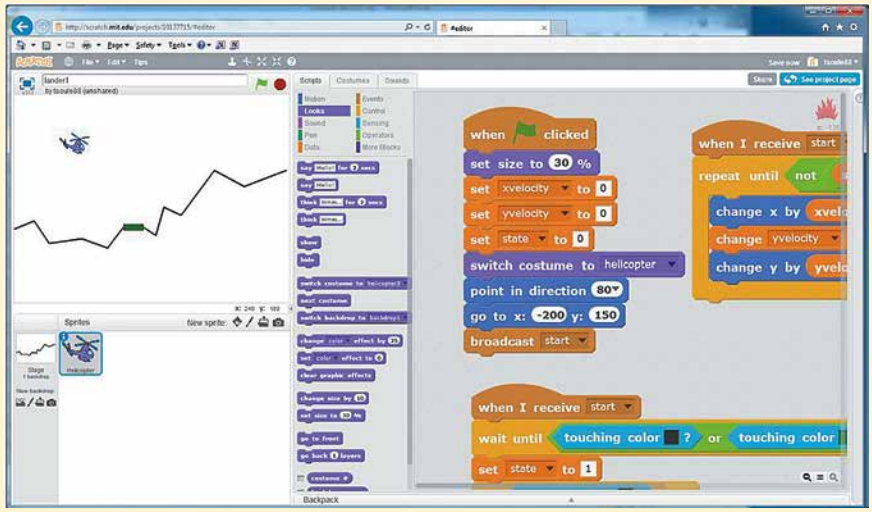
By 2020, five of the top ten in-demand jobs in the United States will be in information technology (Moeller 2012). Companies across the nation are seeking a new type of employee: one who is computer savvy and who is familiar with computer coding, data,

from Scratch

HIGHWAYSTARZ-PHOTOGRAPHY/THINKSTOCK

Fig. 1 This illustration is one example of a sprite in motion.

Fig. 2 In Scratch, the code "scripts" are created on the right by dragging code elements from the center list. The window on the top left shows the results of the program.



Source: Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. http://scratch.mit.edu

mathematics, and augmented reality (Leber 2013). Recent reports indicate that, although students are initially interested in jobs in science, technology, engineering, and math (STEM) fields, positive attitudes toward mathematics drastically decline between seventh grade and tenth grade, and interest in careers involving mathematics decrease as well (University of Idaho 2013).

Findings indicate that girls' attitudes diminished at a far more substantial rate than that of boys, highlighting the need to focus on girls and STEM disciplines (University of Idaho 2013). As a result, we designed and implemented a computer-coding unit, specifically for girls in grades 6–8, focused on increasing their technological and mathematical understanding. This article describes our approach and provides directions for integrating technology with mathematics as students learn to create virtual games and programs.

CODING: INTEGRATING MATHEMATICS, SCIENCE, AND TECHNOLOGY

The computer-coding unit was designed to increase digital literacies and mathematical fluency; it also promotes technological careers for girls as they become familiar with hardware and software and learn how to program. Programming activities focus on Scratch, a free programming environment developed at MIT Media Labs to introduce students to computer programming (Resnick et al. 2009). Scratch employs a graphical two-dimensional, drag-and-drop programming approach, with easy-to-understand graphical icons representing basic programming elements (see **fig. 2**).

Scratch accounts give users the ability to create and run Scratch programs, save programs in their own online folders, and share programs with the Scratch community. User can copy and "remix" any shared Scratch program and can track how many times their shared projects have been viewed or remixed. This element can be motivating for students. Scratch has a large online community, with a strong emphasis on sharing, which situates programming as a cooperative, social activity that allows opportunities to share, discuss, and modify programs.

CLASSROOM IMPLEMENTATION

We wanted to increase students' awareness about using technology in society and careers while addressing mathematical concepts in an engaging way. To address these goals, we began by developing a studio of sample programs with scripts to illustrate introductory programming concepts in Scratch. These scripts served as potential starting points for the students' own projects. (The studio of samples is available online at http://scratch.mit.edu/studios/211999/.) Early lessons addressed how to create a new project, add and modify sprites, move sprites around the screen, and have sprites talk via word bubbles.

UNIT PROGRESSION

The general progression of lessons in the unit, including the coding emphasis and mathematical concepts introduced, are outlined in **table 1**. To introduce new concepts, each session began with ten to twenty minutes of discussion about the concepts for the lesson followed by ten to twenty minutes of student programming

Table 1 The lessons in the unit included both coding and mathematical concepts.

Lessons	Code Topic/Function
1. Sprite movement, including Cartesian coordinates, and absolute and relative motion	**Go To, Glide, Move** Compare and contrast two ways to move sprites: 1. Absolute movement in the Cartesian plane (*go to, glide*) 2. Relative movement (*forward*) based on the sprite's current position and heading
2. Sprite interaction, including how different events can be controlled to occur at specific times	**Wait and Broadcast/Receive** Two ways to coordinate sprites: 1. Sprites alternatively wait for fixed periods (*wait*) 2. Sprites wait to receive a message before continuing (*broadcast* and *receive*)
3. Special effects, animation, graphical effects, and sounds, including percentages and scaling	**Effects, Loops, Costumes Runtime** Create animation loops (*repeat* loop) Understand animation as frames per second, combining animation and movement Manipulate directionality and speed Use special effects via the *change effect X by Y* command Explore Loops to iterate through the effect values **Bitmaps and Vector Graphics** Understand sprite size and rotation
4. Sprite control, including conditional, absolute, and relative movement	**Sensing, Conditionals, Position, Movement** Understand the *if* command and the use of conditions to control behaviors and movement Introduce variables to control velocity Discuss movement in the coordinate plane more in depth
5. Physical movement, including velocity, acceleration, gravity, and vectors	**Variables, Operators, Logic in Conditionals, Vectors, Equations of Motion** Understand and state variables Calculate the *X* and *Y* motion of a sprite independently Understand equations of motion Incorporate levels of backgrounds Use angular movement of sprites (i.e., the tipping of a helicopter)

time. During the discussion time, the instructor wrote or presented a program and described its key features: for example, the use of vectors, Cartesian coordinates, or a particular programming construct. Students were encouraged to program along with the instructor and develop their own ideas, asking questions and engaging in conversations with other students during the programming process.

Instructor: What would you like to do?
Student: I was thinking I want to do something like the hopping game. But instead of . . . oooh, we could do a dolphin swimming game.
Instructor: That would work too.
Student: Yeah, the sprites are all there.
Instructor: If you started with the helicopter game, and then you know how the velocity goes down?
Student: Negative.
Instructor: You could make that positive, so the dolphin is floating to the surface.
Student: OK.
Instructor: And you could say, I don't want to float all the way to the surface.
Student: I'm going to try that one first.

During the programming time, students worked in small groups,

usually pairs, to experiment with the newly introduced concepts and to include them in their ongoing projects.

MATHEMATICS USED IN SCRATCH PROJECTS

Lessons in this program contained two student projects: a scene with animated, interacting sprites; and a game involving user-controlled objects. These projects were designed to emphasize the role of mathematics and science in computing and technology. The lessons focused on three mathematical topics: mathematical descriptions of position and movement; scaling and percentages; and mathematical abstraction of physics concepts, such as acceleration and velocity, with vectors.

POSITION AND MOVEMENT

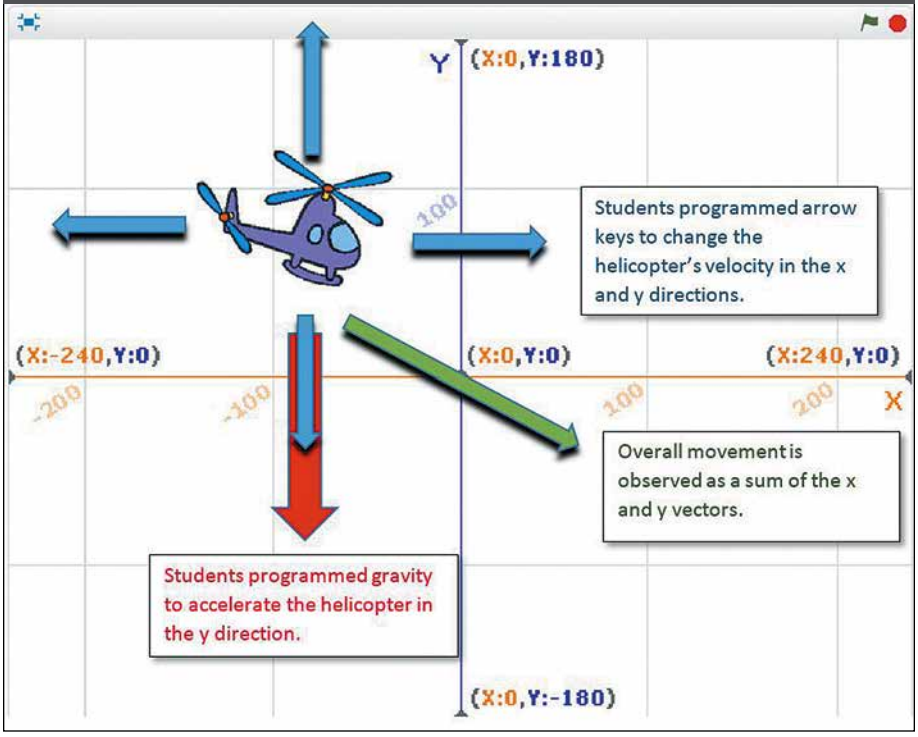
The lessons on movement were incorporated into a project that required students to create and animate scenes in which multiple sprites moved and interacted. Student scenes ranged from humorous, with two sprites telling jokes; to socially educational, with two sprites making friends searching for a lost dog; to topical, with a sprite introducing photographs taken during the code camp itself. Thus, although understanding the mathematics was critical to making scenes work, it was incorporated into much broader, student-motivated projects.

Scratch supports two basic movement styles: absolute movement within a predefined Cartesian plane (e.g., move to (25, 25)); and relative movement based on current position (e.g., turn 20 degrees, move forward 10 steps). To compare and contrast these approaches, specific lessons used both sets of commands on a Scratch background with Cartesian axes (see **fig. 3**).

During the whole-group discussion time, the students and instructor worked together to move sprites to specific locations. Following this, the students worked in teams to manipulate sprite movement on individual computer screens. Students began by deciding where they wanted their sprites to be or to move and then estimated or calculated the distance, direction, or absolute location that needed to be coded to get the desired movement. By running their programs, they could get immediate feedback regarding their estimates and calculations. These lessons demonstrated how relative and absolute movement changes are related and introduced the idea of vectors, even vector addition, as multiple small steps that resulted in one large step.

Students compared absolute movement in a Cartesian coordinate system, including go to (*X*, *Y*); to relative movement, turn (number of degrees); followed by forward (number of steps). The girls had to decide which approach to use, absolute or relative movement, to control the sprites (such as the helicopter in **fig. 3**). They then had to use mathematics to program the correct angles, trajectories, speed, and direction of their sprites. This content was introduced from a real-life scenario that related to the girls' experience. Similarly, **figure 4** shows a picture of two female sprites shortly after one moved over to the other. The students had to write computer code to make the girl on the left walk horizontally to the girl on the right.

Fig. 3 A helicopter sprite is maneuvered on a Cartesian coordinate plane.



SCALING AND PERCENTAGE

Scale factors and percentages were also introduced to control features of the sprites such as size and transparency, which are defined as percentages. Scratch incorporates special effects to control some sprite qualities, such as size, color, and transparency, which are determined by an adjustable value representing a percent effect. During the teacher demonstration, the different effects were presented, and the students observed how different values changed a sprite's appearance. Students then had the opportunity

to add similar effects to their own projects. They began by deciding on the desired effect and calculating the required changes to the special-effect value. For example, to have a sprite slowly fade out of the scene, a student might increase the transparency by 5 percent per time step, whereas to have it quickly fade, he or she might increase transparency by 10 percent per time step.

The girls participating in the computer-coding unit increased their mathematical vocabulary and comfort with mathematical concepts as they designed the contexts for their sprites and designated certain opacities to given features in their virtual worlds. For example, the girls began manipulating their sprites and the surrounding scenes to create a more realistic environment, such as sheer curtains on windows. With this example, they had to adjust the opacity on the objects by considering the relationship between changing the percentages and the effect on the objects. This often resulted in conversations about the difference between 20 percent opacity versus 60 percent opacity and how those differences changed their scenes.

Running the program gave students immediate feedback on how well their choices matched the desired effects and allowed them to experiment with the effect of changing values by different amounts. Students were encouraged to pursue their own goals as they manipulated their sprites and engaged in the related mathematics. One student combined motion with successive size changes to create sprites that appeared to move forward from the background and retreat into the background; this action was a fairly sophisticated application of relative size and perspective.

PHYSICS CONCEPTS

Later lessons focused on creating simple physics-based games, which included landing a flying vehicle, catching falling or bouncing objects, and moving through a maze. Simple versions of all three games were presented in separate lessons. Students created their own games by modifying and building on the sample games, which were used to motivate lessons on vectors and equations of motion.

Vectors were formally introduced to describe the velocity of falling and bouncing objects, including the idea of separate vectors for horizontal and vertical movement, combining those vectors to create movement in arbitrary directions, reversing velocity vectors to "bounce" objects, and decreasing velocity during a bounce to model loss of momentum. Simple equations of motion were introduced to model the movement of falling objects and flying vehicles, for example:

$$\text{position}_{\text{new}} = \text{position}_{\text{old}} + \text{velocity} \times \text{time}$$

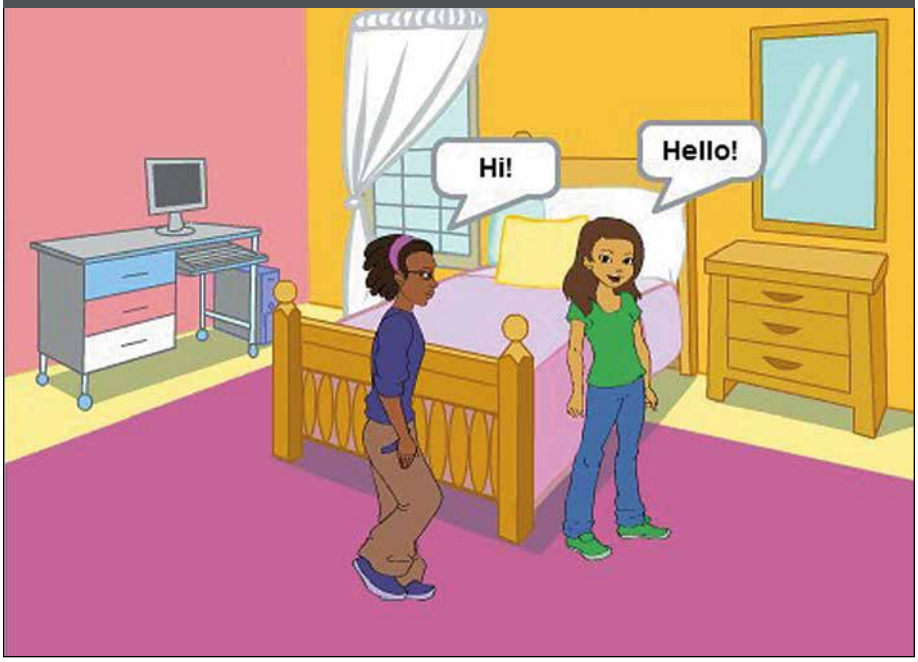
$$\text{velocity}_{\text{new}} = \text{velocity}_{\text{old}} + \text{acceleration} \times \text{time}$$

Physical concepts, including velocity, acceleration, gravity, and friction, were incorporated into the projects using mathematical relationships. The velocity of a falling helicopter was defined as

$$V_{\text{new}} = V_{\text{old}} + \Delta V,$$

and vectors were introduced to control objects moving in a plane (see **fig. 3** and the helicopter sprite on the Cartesian coordinate background). Students observed how movement in arbitrary directions could be defined using the sum of perpendicular vectors. Students programmed the helicopter to move based on thrust along the x- and y-axes and the influence of gravity. The project was used to teach about vectors and basic equations of motion: position, velocity, and acceleration.

Fig. 4 Students used basic equations to move their sprites.



GAMES, MOTIVATION, AND MATH

We found that a significant advantage of Scratch is that students can easily modify the appearance and thus

"If you get it wrong on there, sometimes it just crashes and burns!"

the theme of projects. For example, by simply changing the background image, the sprite, and even the sound effects used in a game, students could change a game involving landing a helicopter in the mountains into a game about flying a Pegasus to a cloud castle. This allowed students to feel an almost instant sense of ownership for their projects, which significantly strengthened their motivation to improve them.

The instructor helped the girls make the connection between their real-life experiences and the mathematics by such statements as these:

> It's surprising, and you may have seen this in video games before, if the character, or sprite, or thing you're controlling doesn't really follow semi-real rules of physics, it's really hard to play, right?

When the girls agreed, the instructor went on to explain:

> And it works like this. I'll write it out this way: X new, the new position sideways, is equal to the old X plus the X velocity times time ($X_{new} = X_{old} + X_{velocity} \times$ time). So this is the basic equation—a basic equation—of motion. Your new position is your old position plus however fast you're going times however long it takes that you're measuring that time for.

The girls then engaged in changing the X velocity and Y velocity and found that their helicopters would immediately crash if they did not correctly manipulate the velocity. After experimenting with the numbers, one girl said, "If you get it wrong on there, sometimes it just crashes and burns!" This process of manipulating the mathematics and seeing the visual effect helped the girls make a connection between the involvement of mathematics and the distance and timing of moving objects.

Project data support the idea that a critical factor in the educational process was that mathematical concepts were introduced specifically to further the students' own goals. Students wanted to create games with more realistic motion and consequently were highly motivated to understand these mathematical concepts for their own purposes. Some of the mathematics presented, such as the content related to vectors, may initially seem challenging for middle school students. However, the context of computer programming provided support for the girls to begin learning the mathematical concepts. Therefore, the context of the computer coding application gave new relevance to the mathematics for the girls.

PROGRAMMING IN YOUR CLASS

Computer coding with Scratch can be easily incorporated into one or two hours a week in a middle school classroom. These opportunities will engage students in all STEM disciplines as they learn to program sprites, focus on the mathematics involved, and create their own virtual realities. By following the aforementioned steps and exploring the programs, teachers can find ways to support student learning through this creative venue.

Math teachers could design a multidisciplinary coding unit with science, technology, and mathematics. For example, students could work on a unit on ecosystems and use Scratch to create a virtual ecosystem with different animals moving in their habitats. Programming these animals to move would involve mathematics, and students would have to make decisions about the fauna and flora in their environment and how they interact with each other. This project encourages interdisciplinary work and can be an engaging tool for encouraging students to learn mathematics while learning how to code.

GETTING EXCITED ABOUT CODING

The girls who took part in the computer-coding unit were excited about learning to code and using mathematics to accomplish their personal coding goals. They appreciated the opportunity to create their own virtual worlds. One participant, Jennie, commented, "I learned about coding, computer programming, teamwork, and that technology and math are easier than they seem." This opportunity presented a new way for the girls to consider mathematics through a context they could personalize. They were able to advance their understanding of mathematics and technology as they built animated scenes from scratch.

ACKNOWLEDGMENTS

The authors would like to acknowledge the following entities that contributed to funding this project: Micron Foundation, Verizon Foundation, iShoutOut, The National Aeronautics and Space Administration, Hewlett-Packard Development Company, and the University of Idaho, Coeur d'Alene.

Ed. note: Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See http://scratch.mit.edu.

BIBLIOGRAPHY

Common Core State Standards Initiative (CCSSI). 2010. Common Core State Standards for Mathematics. Washington, DC: National Governors Association Center for Best Practices and the Council of Chief State School Officers. http://www.corestandards.org/wp-content/uploads/Math_Standards.pdf

Leber, Jessica. 2013. "In a Data Deluge, Companies Seek to Fill a New Role." *MIT Technology Review*: *News and Views*. http://www.technologyreview.com/news/513866/in-a-data-deluge-companies-seek-to-fill-a-new-role/

Moeller, Philip. 2012. "Where the Jobs Will Be in 2020." *U.S. News and World Report*. http://money.usnews.com/money/careers/articles/2012/09/10/where-the-jobs-will-be-in-2020

Resnick, Mitchel, John Maloney, Andres M. Hernandez, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. "Scratch: Programming for Everyone." *Communications of the ACDM* 52 (11): 60. doi:http://dx.doi.org/10.1145/1592761.1592779

Rodger, Susan, Maggie Bashford, Lana Dyck, Jenna Hayes, Liz Liang, Deborah Nelson, and Henry Quin. 2010. "Enhancing K–12 Education with Alice Programming Adventures." In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, edited by Reyyon Ayfer, John Impagliazzo, and Cary Laxer, pp. 234–38. Ankara, Turkey: ACM Special Interest Group on Computer Science Education.

University of Idaho. 2013. "Investigating Influences in Idaho STEM Education." http://www.uidaho.edu/research/STEM/stem-micron/micronstemed/project-reports

Any thoughts on this article? Send an email to ***mtms@nctm.org.***—*Ed*.

Julie M. Amador, jamador@uidaho.edu, is an assistant professor of mathematics education at the University of Idaho, Coeur d'Alene. She directs a regional mathematics center and is interested in how preservice and in-service teachers professionally notice students' mathematical thinking. **Terence Soule**, tsoule@cs.uidaho.edu, is an associate professor of computer science at the University of Idaho. His primary research interest is machine learning and robotics, and he deeply enjoys teaching students of all ages and abilities.