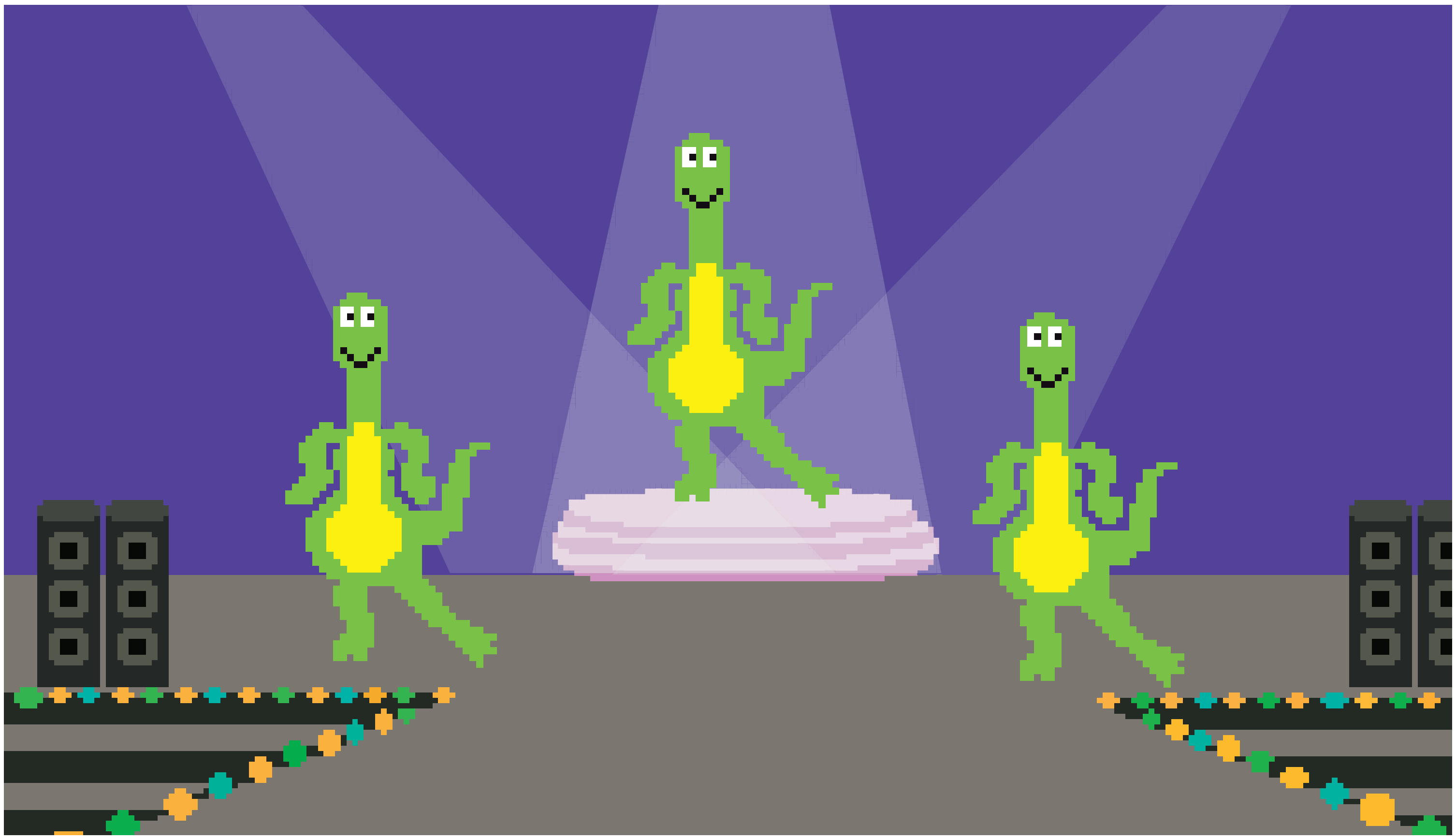
teach coding with SCRATCH™

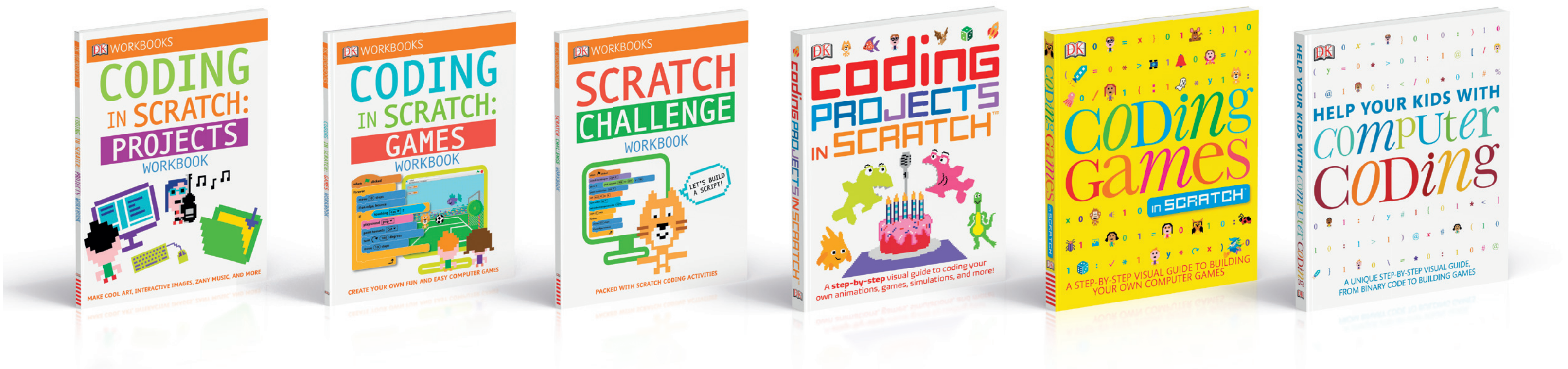
A **MIDDLE SCHOOL** Teacher's Guide



DK A WORLD OF IDEAS: **SEE ALL THERE IS TO KNOW**

A **MIDDLE SCHOOL** Teacher's Guide prepared by Grant Smith

To complete the projects outlined in this guide, your students will need the following DK publications:



Note to Educators:

Coding with Scratch Workbook, ***Coding in Scratch: Games Workbook***, ***Scratch Challenge Workbook***, ***Coding Projects in Scratch***, and ***Coding Games in Scratch*** serve as workbooks and guides that are useful in helping young learners understand and create simple computer programs. The books contain project walk-throughs that will engage all types of learners. Many projects also include suggested "hacks and tweaks" that are perfect for differentiated learning.

The workbooks (***Coding with Scratch Workbook***, ***Coding in Scratch: Games Workbook***, and ***Scratch Challenge Workbook***) provide a scaffolded learning environment that can be used to build student knowledge. These workbooks also contain short quizzes that can serve as formative or summative assessments.

The larger guides (***Coding Projects in Scratch*** and ***Coding Games in Scratch***) promote student choice by outlining projects in many fields of interest including art, music, games, simulations, and more. These projects are perfect for aligning your lessons to another content area.

Seymour Papert

In Mindstorms (1980), Papert wrote: *"One might say the computer is being used to program the child. In my vision, the child programs the computer, and in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intense contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building."*

Middle school is when many teachers transition their students from block-based coding to written programming languages like Python. The ***Help Your Kids with Computer Coding*** book will be especially useful for these teachers because it outlines programming in both Scratch and Python. You should also look forward to a new book, ***Coding Projects in Python***, that will be published in June 2017.

Scratch builds on the constructionism learning theory. Seymour Papert developed this theory as a branch of Jean Piaget's description of constructivism. However, the difference is that Papert focused on the social aspects of learning. The idea is that students learn best when they use the knowledge they have to build an artifact and share it. Scratch was intentionally developed not only to provide an open and creative coding sandbox, but also to allow users to build and share projects on a global scale.

Papert explained that "the role of the teacher is to create the conditions for invention rather than provide ready-made knowledge." This teacher guide has been developed with the goal to assist teachers (even those with little to no computer science content knowledge) in effectively creating the ideal "conditions for invention" for their students. At this point you may be asking yourself the following questions:

- ***Will I know the answer to every question that my students will have?***
- ***Will I feel well-rested, prepared, and in control at all times?***
- ***Will every class run without a hitch?***

Let me answer those questions for you:
1) No. 2) You wish. 3) In your dreams!

Will it be worth it? You better believe it!
Now let's make it happen!

GETTING STARTED

All the projects in this teacher guide will be made by your students using Scratch. Scratch was developed by the Lifelong Kindergarten group at the MIT Media Lab. It costs nothing and takes student privacy seriously. Scratch is a coding environment where students snap blocks together to create computer programs. The advantage to learning in a block-based environment is that your students won't have to worry about complicated language syntax. Additionally, Scratch provides a learning environment that is easy to get started with, allows a wide variety of types of projects, and is powerful enough to create fairly complex programs.

Before you get started with your students, you will need to set up Scratch. There are online, offline, and iPad versions of Scratch. To choose which one is the best for your class, you will need to consider the following (and it may be a good idea to get help from your technology department):

- ***Does your school/district have policies against creating online student accounts?***
- ***Are you able to download programs/apps onto your devices?***
- ***What kinds of devices do you have?***

Pages 24–25 of *Help your Kids with Computer Coding* provide an overview of how to sign up for or download Scratch. If you only have access to iPads, you can use Pyonkee (based off Scratch 1.4, not officially made by MIT). If your district has policies against students under 13 years old creating online accounts, consider applying for a Scratch Educator account.

Before your first lesson, you should complete the student projects found in this teacher guide. Completing the activities yourself will help you anticipate the needs of your own students. Also, your completed projects will serve as models to show students what they will be making.

INTRODUCTION TO PROGRAMMING:

If you are unfamiliar with computer programming, prepare to introduce coding to your students by reading pages 12–23 of the *Coding Projects in Scratch* guide. Understand that an algorithm is simply a list of steps to complete a task and a program is an algorithm that a computer can run.

You can introduce coding to your students by showing one or all of the following videos:

Coding for Kids 1: What Is Computer Coding?
http://bit.ly/2rmKFeV

Coding for Kids 2: How Computer Programs Work
http://bit.ly/2qociHm

Coding for Kids 3: Think Like a Computer
http://bit.ly/2qnZnoL

In **Coding for Kids 3: Think Like a Computer**, the robot has to be given very detailed and clear instructions to successfully serve the food. You can model this with your students by pretending to be a robot. Ask students to "program" you by making a list of instructions for you to follow to complete a task (e.g. walk to the door, go to your desk and pick up a pencil, etc.). Also read through the Think Like a Computer example on pages 16–17 of *Help Your Kids with Computer Coding.*

INTRODUCING STUDENTS TO SCRATCH

Next, you will introduce your students to Scratch. You can show the **Computer Coding Games for Kids: Introducing Scratch** video (*http://bit.ly/2pQNKnr*) to give an overview of the environment. Then have students open to pages 8–9 of the Coding with Scratch guide and have them point to each area as you say them (Stage area, Sprite list, Stage info . . .). The goal at this point is not to have the students immediately memorize what each button and block in Scratch does. Rather, they should become familiar with the environment and start to develop a common vocabulary when describing what they are doing in Scratch.

For many teachers, the next natural step would be to teach a concept and give an assignment. However, experienced computer science teachers will tell you that after the introduction, students will be so excited to get started that they most likely won't listen to anything else you have to say. That's why you should jump right in by having students login or open Scratch. Set a timer and tell your students they have *X* minutes to "discover something new." Encourage them to share discoveries with their neighbors and to use the correct names when referring to areas in Scratch. While circulating, ask students to point out something interesting they have discovered. Make sure to model using correct terminology. When the time is up, select students to share projects they made or interesting facts they learned. You may be surprised at what students can make and learn without any instruction from you. Encourage divergent thinking and self-reflection.

At the end of your lesson, congratulate your students for becoming computer programmers! Explain that they have some exciting projects ahead of them and will be learning about how to make their computers do amazing things.

LOOPS

The concept of what a loop is in programming can be very easy to understand. However, knowing when and how to use loops takes middle school students some time to master. It requires thinking about patterns, decomposing problems, and improving programs iteratively. Show your students the three types of loop blocks found in Scratch and discuss how they are different.

Help your students by guiding them through the explanation of simple loops found on pages 46–47 of ***Help Your Kids with Computer Coding***. If your students are ready, also check out the explanation of complex loops on pages 68–69 of ***Help Your Kids with Computer Coding***.

Next, have students work on the simple introduction project: ***Help Your Kids with Computer Coding***—Roll the Dice (found on pages 60–73).

As with all main projects, the end product shown in the book should not be where your students finish working. The goal for your class should be that all of your students' projects are unique. Many of the projects in the books have a "hacks and tweaks" section. These suggestions are perfect for helping your students generate ideas on how they can make their programs unique. Requiring uniqueness is also a great way to differentiate learning. For some students, their modifications may be simple, which may be perfect for them. Other students will look forward to pushing themselves as they create something more complex. In the end, every student should feel like they were challenged and like they accomplished something they are proud of.

The project uses a simple repeat loop to animate a dice to look like it is rolling. Have students experiment by placing different numbers in the repeat block or by adding other blocks to the inside or outside of the loop. When they are ready to move on, have your students complete this main project:

Main project:

Help Your Kids with Computer Coding—Escape the Dragon (found on pages 12–23)

For the main project, your students will create a simple chase game. The program uses three loops to keep the game running until the player loses. To make their project unique, encourage students to try and incorporate one of the other two loops into their code.

After the students finish their projects, have them explain what loops do. Spend time reflecting on their work. Ask the students to think of a few more examples of when loops may be useful (even outside of coding).

EVENTS AND PARALLELISM

Events and parallelism will bring your students' programs to life. Event blocks trigger scripts to run. Your students have already used events blocks like "when green flag is clicked" and "when space bar is clicked." Parallelism simply means that computers can run multiple scripts at the same time, either independently or in conjunction with each other. Guide your students through the Events section on pages 44–45 and the Sending Messages section on pages 70–71 of *Help Your Kids with Computer Coding* to help them understand the the many types of events used in Scratch.

To illustrate that events trigger certain chunks of code, have your students work on the following starter project:

Coding Projects in Scratch—Sprites and Sounds (found on pages 182–188)

The Sprites and Sounds project only uses one type of event: the "when this sprite is clicked" block. In the program, the students create a lot of code, but only small chunks (or scripts) are run at a time, depending on which sprites are clicked.

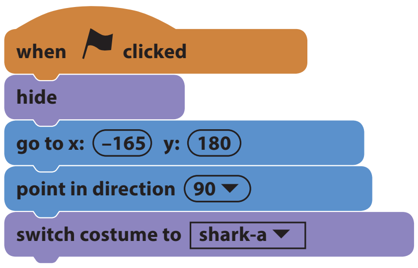
Now introduce more events and parallel programming with the main project:

Main project:

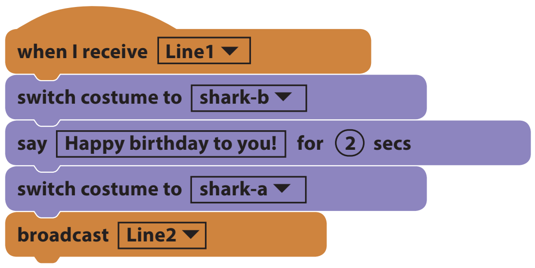
Coding Projects in Scratch—Birthday Card (found on pages 82–91)

For the main project, students will use events and parallelism to make an interactive birthday card. Events will allow the user to control when chunks of code should run, and parallelism will enable multiple sprites to run scripts at the same time. The Birthday Card project has some fun hacks and tweaks for students to make their project unique.

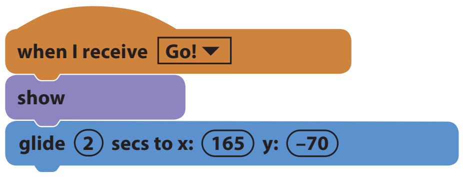
Here are some examples of events in the Birthday Card project:



```
when ⚑ clicked
hide
go to x: (-165) y: (180)
point in direction (90 ▾)
switch costume to [shark-a ▾]
```

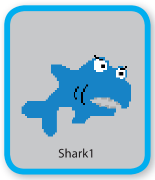


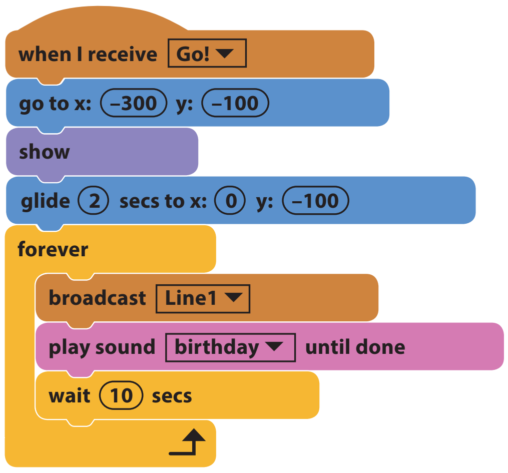
```
when I receive [Line1 ▾]
switch costume to [shark-b ▾]
say [Happy birthday to you!] for (2) secs
switch costume to [shark-a ▾]
broadcast [Line2 ▾]
```



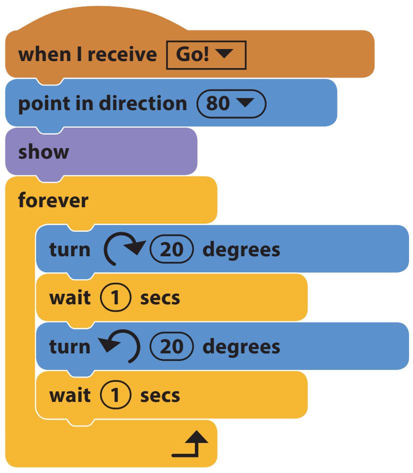
```
when I receive [Go! ▾]
show
glide (2) secs to x: (165) y: (-70)
```

Here are some examples of scripts running in parallel:
When the message "Go!" is broadcast, the Cake, Banner, and Shark1 sprite all run different scripts at the same time.

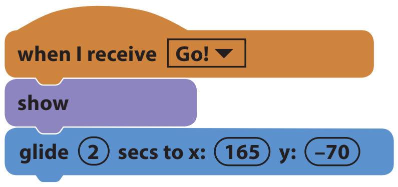




```
when I receive [Go! ▾]
go to x: (-300) y: (-100)
show
glide (2) secs to x: (0) y: (-100)
forever
    broadcast [Line1 ▾]
    play sound [birthday ▾] until done
    wait (10) secs
↩
```



```
when I receive [Go! ▾]
point in direction (80 ▾)
show
forever
    turn ↻ (20) degrees
    wait (1) secs
    turn ↺ (20) degrees
    wait (1) secs
↩
```



```
when I receive [Go! ▾]
show
glide (2) secs to x: (165) y: (-70)
```

CONDITIONAL STATEMENTS

Conditional statements will make your students' programs smarter. In Scratch programming, conditional statements use the if-then and if-then-else blocks. It's a good idea to introduce conditional statements by giving real-world examples that students can relate to. For example, "If you don't clean your room, then you are grounded." Or, "If you make a goal, your team gets a point added to their score." Try writing these examples in plain English, then illustrate what they would look like if they were Scratch blocks. Have students come up with their own conditional statements and have them illustrate them as Scratch blocks. If they are ready, have students also create if-then-else statements like "if you clean your room, then you can watch TV, else you are grounded." Guide your students through the Decisions and Branches section on pages 64–65 of ***Help Your Kids with Computer Coding***. For their project, have students choose to make one of the following:

Coding Projects in Scratch—Dino Dance Party
(found on pages 34–45)

For the Dino Dance Party, students will use loops, events, parallelism, and conditional statements to make animated characters dance on the screen. A small script in the program will ask the computer if the left or right arrow key is pressed on the keyboard. If either key is pressed, Dinosaur 3 will move in the corresponding direction. These conditional statements make the program interactive and more fun!

Coding in Scratch: Projects Workbook—Pet Party
(found on pages 24–29)

For the Pet Party, students will use loops, events, parallelism, and a conditional statement to make their pet do crazy things. A conditional statement will check if a food sprite is touching the pet and will run a mouth animation when the condition is true. The easiest modification is to change the types of sprites used in this program. However, push your students to make their project unique through original code they add.

Don't forget to have your students check out the "hacks and tweaks" section to come up with ideas on how they can make their version of the Dino Dance Party program unique!

VARIABLES AND OPERATORS

While math operators like addition, subtraction, and division work the same in programming as they do in math, variables are slightly different. Have your students work through pages 50–51 of ***Help Your Kids with Computer Coding*** to understand variables and pages 52–53 to understand how to use math tools in Scratch. A common use for variables in Scratch is to keep track of a player's score in a game. Before jumping to the main project, have your students work on the starter projects below. The first project uses math without variables and the next project uses two variables.

Coding in Scratch: Projects Workbook—Weird Music
(found on pages 8–12)

Coding in Scratch: Projects Workbook—Skywriting
(found on pages 14–17)

Before moving on to the main project, give students an opportunity to modify their starter projects to show that they understand how to use variables and math in Scratch. When ready, allow students to choose one of the following for their main project:

Coding in Scratch: Games Workbook—Fishball
(found on pages 8–16)

Coding in Scratch: Games Workbook—Ghost Hunt
(found on pages 18–22)

Coding in Scratch: Games Workbook—Rapid Reaction
(found on pages 24–28)

Coding Games in Scratch—Cheese Chase
(found on pages 50–73)

Coding Projects in Scratch—Ask Gobo
(found on pages 60–65)

Coding Projects in Scratch—Tunnel of Doom
(found on pages 122–132)

Coding Projects in Scratch—The Magic Spot
(found on pages 200–206)

If students finish early, encourage them to either continue improving their project or choose another project from the list to work on.

FUNCTIONS

Functions will take your students' programs to the next level. Functions can also be difficult for students to use effectively because they require students to think about how they can reuse portions of code in multiple ways or with varying input. In Scratch, functions are made by creating a new block. Carefully work with your students through the explanation of how to create blocks found on pages 72–73 of ***Help Your Kids with Computer Coding***. You may want to break up the starter project found below and work on it together as a class. Provide plenty of scaffolding for students as they learn about using functions.

Coding Projects in Scratch—Drumtastic (found on pages 190–197)

When ready, have your students choose a main project from the following options:

Coding Projects in Scratch—Ask Spiral-o-tron (found on pages 208–214)

Coding Games in Scratch—Glacier Race (found on pages 166–189)

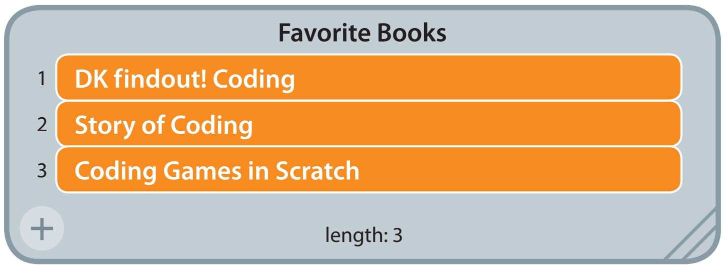
Coding Games in Scratch—Dog's Dinner (found on pages 130–165)

As always, if students finish early, encourage them to either continue improving their project or choose another project from the list to work on.

For an extra challenge, the following project uses parameters, or input, in the created functions to draw flowers of various sizes and shapes: ***Coding Projects in Scratch***—Fantastic Flowers (found on pages 106–117)

LISTS

If your students have ever made a list of their top-ten favorite candies, friends, or movies, then they clearly understand what lists are. In Scratch, lists act like variables but hold many items. Each item is assigned a row number. Things get tricky when students have to insert, delete, and replace items in their lists.



Favorite Books

1 DK findout! Coding
2 Story of Coding
3 Coding Games in Scratch

+ length: 3

Have your students work through page 55 of Help ***Your Kids with Computer Coding*** to learn about making and using lists. When ready, ask your students to create, then modify this musical game:

Scratch Challenge Workbook—Memory Master (found on pages 28–34)

If you have students who are looking for something more challenging, differentiate their experience by assigning them this more intense project:

Coding Games in Scratch—Tropical Tunes (found on pages 190–205)

Push your students to think of creative ways to use lists and store other bits of information in their projects.

CLONING

There are probably days in the classroom when you wish you could clone yourself. While that is sadly impossible, it is possible to clone sprites in Scratch. Instead of re-creating the same sprite manually, there are blocks in Scratch to create, control, and delete clones while the program is running. If your students ever make an exact duplicate of a sprite, you should ask them if they would be better off using clones. Help your students understand how to use clones by having them complete the following starter project:

Coding Projects in Scratch—Funny Faces (found on pages 70–77)

In the starter project, students created clones of a sprite without defining code for each clone to run when it gets created. In the main project, students will not only create clones, but they will also tell those clones how they should act. Have your students choose from one of the following main projects:

Scratch Challenge Workbook—Sound Party (found on pages 8–14)

Coding Projects in Scratch—Spiralizer (found on pages 94–102)

Coding Games in Scratch—Doom on the Broom (found on pages 108–129)

If students need an extra challenge, ask them to use lists or functions in their modified version of the main project.

CAPSTONE PROJECT

At this point your students haven't mastered all there is to know about coding, but they have learned enough to make some really neat programs. Give students an opportunity to show off what they know by creating and presenting a capstone project. Have students choose a base project from the list below.

Coding Projects in Scratch—Fractal Trees
(found on pages 162–167)

Coding Projects in Scratch—Snowflake Simulator
(found on pages 172–178)

Coding Projects in Scratch—Fireworks Display
(found on pages 154–159)

Coding Projects in Scratch—Window Cleaner
(found on pages 134–141)

Coding Projects in Scratch—Virtual Snow
(found on pages 144–152)

Coding Games in Scratch—Circle Wars
(found on pages 74–89)

Scratch Challenge Workbook—Monkey Rescue
(found on pages 22–26)

Consider having students work in groups to create their capstone projects. Clearly outline roles and what the expectations are for completing the project and working in a group. Actively help students stay on track so that they finish their projects by the due date.

Make sure students modify their end product so that the program is unique. When your students finish, it is a good idea to have them present their projects to their class, parents, or students in other grade levels.

Congratulations! You made it! You have taught your students the basics of coding in Scratch. But don't stop now—keep challenging them! Get your hands on other DK coding books or come up with your own subject-aligned projects. Consider transferring students to a written programming language like Python. Check out *Coding Projects in Python* by DK for more ideas. Continue to prepare your pupils for thriving in the twenty-first century.

MORE RESOURCES

Scratch ED
http://scratched.gse.harvard.edu/
An online community for Scratch educators to collaborate and exchange resources.

Downloadable DK Coding Kit
https://www.dk.com/us/explore/education/celebrate-global-scratch-day-with-this-downloadable-computer-coding-kit/

DK 9 Easy Steps Coding Guide
https://www.dk.com/us/explore/education/9-easy-steps-for-teaching-coding-in-the-classroom/

ARTICLES TO READ

Mitch Resnick: Let's teach kids to code
https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code

Learning to Code Isn't Enough
https://www.edsurge.com/news/2013-05-28-opinion-learning-to-code-isn-t-enough

About the Author

Grant Smith is the founder of **Launch CS** (*www.launchcs.com*), the premier provider of K–8 computer science teacher professional development. Grant is also a former K–8 computer science teacher and district administrator. He has led #CSforAll initiatives at multiple school districts across the nation and has developed computer science curricula and standards. He has served on national computer science education teams including the CSTA Standards Review Committee.