Timothy L. Warner

In Full Color

Sams Teach Yourself

Scratch™ 2.0

in 24 Hours

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Timothy L. Warner

Sams **Teach Yourself**

Scratch™ 2.0

in 24 **Hours**

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself Scratch™ 2.0 in 24 Hours

ISBN-13: 978-0-672-33709-3
ISBN-10: 0-672-33709-6

Library of Congress Control Number: 2014933847

Printed in the United States of America

First Printing: July 2014

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Scratch is a registered trademark of MIT.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author(s) and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Editor-in-Chief
Greg Wiegand

Executive Editor
Rick Kughen

Development Editor
Mark Renfrow

Managing Editor
Kristy Hart

Senior Project Editor
Betsy Gratner

Copy Editor
Karen Annett

Senior Indexer
Cheryl Lenser

Proofreader
Katie Matejka

Technical Editor
Patrick Mangan

Publishing Coordinator
Kristen Watterson

Cover Designer
Mark Shirar

Compositor
Nonie Ratcliff

Contents at a Glance

Table of Contents

About the Author

Timothy Warner is an IT professional and technical trainer based in Nashville, Tennessee. Tim began his programming career in 1982 when his dad bought the family a Timex Sinclair 1000 home computer and he began teaching himself BASIC programming. Today Tim works as a technical trainer for Skillsoft, a premier provider of live instructor-led training. You can reach Tim directly via his LinkedIn profile at https://www.linkedin.com/in/timothywarner.

Dedication

To my beautiful, amazing daughter, Zoey Elizabeth, who loves technology and Scratch programming as much as her daddy does.

Acknowledgments

For a variety of reasons, this book was challenging to write. I extend my biggest debt of gratitude to my valiant and open-minded editors, Rick Kughen and Mark Renfrow. Thanks also to my publishers, Greg Wiegand and Paul Boger—you guys are great. Thank you to the always helpful and efficient Pearson production and administrative staff, including Kristen Watterson, Betsy Gratner, and Kristy Hart.

Technical books like this put a special burden on its content editors. Thanks so much to Patrick Mangan, my technical editor, and to Karen Annett, my copyeditor, for your thoroughness in making the manuscript as best as it can be.

Thanks to my family—Susan, Zoey, and our menagerie of pets—for putting up with my occasional grumpiness as I burned through the rough spots of this project.

Finally, thanks to you, my readers: Without you, I have no teacher-student circuit to complete, and I would write into the void. That's no fun, so I want you to know how grateful I am that you're reading this book and participating in this learning journey with me.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: consumer@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Introduction

"My task, which I am trying to achieve is, by the power of the written word, to make you hear, to make you feel—it is, before all, to make you see."

—Joseph Conrad, *Lord Jim*

So you want to learn programming? If you were here, physically right in front of me, I would ask you the following questions:

- What was it that got you interested in learning to do computer programming? An iOS or Android app? One of your teachers at school? A family member or friend?
- Where do you envision taking your programming skills? Are you considering programming as a career, a money-making venture, or simply a fun, satisfying hobby?

Regardless of your motivations, I'm happy to welcome you to the always challenging, sometimes fun, sometimes tedious world of learning to write computer programs. In choosing Scratch 2.0, you've made an excellent choice for your first programming language, if you are a beginner.

Why? Because Scratch is a visual, drag-and-drop programming environment that allows you to be creative without having to get bogged down in learning strange syntax rules like you do in more formal languages such as JavaScript or Python.

By the time you've finished this book and completed all of the Try It Yourself exercises, you'll not only be an expert with Scratch programming, but you'll also have a number of real-world programming best practices under your belt.

Please note that I don't dismiss Scratch as a "toy" programming language. In this book, you'll make various projects that other people can actually play and enjoy. We're talking about programs like games, educational interactions, or multimedia storybooks—the proverbial sky is the limit.

Who Should Read This Book

Any author worth his or her salt always writes with the audience in mind. As far as I'm personally concerned, I envision my readers as coming from one (or more) of the following experience contexts:

- **Brand new to programming**: Welcome! You don't need any prior experience with programming to gain value from this book. The only related experience I hope you have is a lot of time spent playing video games and using other multimedia apps so you have some ideas to pursue in Scratch.
- **Considering a career change**: Perhaps you are a K-12, junior college, or university student who has perhaps a bit of past programming experience, and you are pondering a full-time career as a software developer. Learning Scratch serves as an excellent diagnostic to gauge your aptitude and interest in the subject matter.
- **Just tinkering**: Maybe you are a technology buff who always wondered what work went into developing a software project. You have no real career aspirations in programming—you just enjoy tinkering and having fun. Well, welcome! You are bound to have a blast learning Scratch!

If you find that you don't belong in any of the previous three classifications, then don't worry about it. Set your sights on learning as much as you can and, above all else, having fun, and you'll be fine!

How This Book Is Organized

Can you learn how to program with Scratch 2.0 in 24 one-hour sessions? Absolutely! The following chapter-by-chapter breakdown details how the material is structured:

- Hour 1, "What Is Scratch?" formally defines what Scratch is, how it came to be, and how you can use the platform to learn real, honest-to-goodness computer programming skills.
- In Hour 2, "Creating Your First Project," you create your first Scratch project. If you are to become an expert Scratch programmer, then you need to get right into the mix.
- In Hour 3, "Working with Costumes and the Stage," you turn your attention to sprites and their potentially many costumes. You also formally meet the Stage and its accompanying backdrops.
- In Hour 4, "Using Motion Blocks," you begin a detailed consideration of every script block in Scratch 2.0. Here, you use Motion blocks to make stuff happen on the Stage.

- In Hour 5, "Using Looks Blocks," you learn how Looks blocks enable you to make sprites interact with your players.
- In Hour 6, "Using Sound Blocks," you add audio to your Scratch projects.
- In Hour 7, "Working with Pen Blocks," you learn to draw on the Stage by using the fun Pen blocks.
- In Hour 8, "Using Events Blocks," you get comfortable with event-driven programming and using Events blocks to orchestrate action in your Scratch projects.
- In Hour 9, "Using Control Blocks," you use the powerful Control blocks to program potentially complicated branching and looping logic.
- In Hour 10, "Using Operators Blocks," you apply (ugh!) mathematics and logical thinking by means of Scratch's Operators blocks.
- In Hour 11, "Using Sensing Blocks," you start to operate with both analog and digital processes and interact with the player more intimately through the Sensing blocks.
- In Hour 12, "Using Data Blocks," you learn how to implement dynamic data (that is to say, variables and lists) into your Scratch 2.0 projects.
- In Hour 13, "Using Cloud Data," you take what you learned about local variables in Hour 12 and scale them out to the cloud by using cloud data—exciting, cutting-edge stuff here, people!
- In Hour 14, "Adding Multimedia to Your Project," you add multimedia (recorded audio, video) to your Scratch projects.
- In Hour 15, "Creating Your Own Blocks," you learn how to build your own custom blocks using the built-in tools in the Scratch 2.0 editor.
- In Hour 16, "Documenting Your Project," you pick up some real-world programming best practices as they relate to source code documentation and unit testing.
- In Hour 17, "Publishing Your Project," you take your work and publish your project on the Scratch website to enable other people from all over the world to play your game.
- In Hour 18, "Using the Scratch Offline Editor," you discover how you can work on your Scratch projects even if your computer is not connected to the Internet.
- In Hour 19, "Troubleshooting Your Project," you learn valuable tips and tricks for debugging your Scratch 2.0 projects, ensuring that they are free from errors and give players the best possible experience.
- In Hour 20, "Remixing a Project," you "stand on the shoulders of giants" by building new Scratch 2.0 projects based on the work of other Scratchers.

- In Hour 21, "Creating Your Own Sprites and Backdrops," you put on your artist's beret and learn how to draw your own sprite costumes and Stage backdrops by using both the built-in Paint Editor as well as a third-party image-editing program.
- In Hour 22, "Implementing Buttons and Multiple Screens," you put your Scratch project on another level of quality by including multiple game screens and button controls.
- In Hour 23, "Connecting Scratch to the Physical World," you use third-party add-on products to link your Scratch programs with stuff happening in the real world. (Think temperature, touch, volume...this is some fun stuff, trust me!)
- In Hour 24, "Capstone Project: Arcade Game," you consolidate what you learned through the previous 23 hours by writing and publishing a fully functional game.

Downloading the Sample Files

As you'll learn soon enough, the Scratch programming community is all about resource sharing. To that end, the solution files for every Try It Yourself exercise in the book are provided for you in the solution archive. To access the archive, go to www.informit.com/title/9780672337093 and click the Downloads tab.

Conventions Used in This Book

In my experience as an author and a teacher, I've found that many readers and students skip over this part of the book. Congratulations for reading it! Doing so will pay off in big dividends because you'll understand how and why we formatted this book the way that we did.

Try It Yourself

Throughout the book, you'll find Try It Yourself exercises, which are opportunities for you to apply what you're learning right then and there in the book. I do believe in knowledge stacking, so you can expect that later Try It Yourself exercises assume that you know how to do stuff that you did in previous Try It Yourself exercises.

Therefore, your best bet is to read each chapter in sequence and work through every Try It Yourself exercise.

About the is.gd Hyperlinks

Whenever I want to point you to an Internet resource to broaden and deepen the content you're learning, I provide a uniform resource locator (URL, also called an Internet address) in the form: http://is.gd/uaKpYD

You might wonder what the heck this is. The way I look at the situation, if I were reading this title as a print book and needed to type out a URL given to me by the author, I would rather type in a "shortie" URL than some long, crazy URL with all sorts of special characters, you know what I mean?

The most important thing I have to tell you concerning the is.gd short URLs is that the ending part is case sensitive. Therefore, typing the previous URL as http://is.gd/UaKpyD isn't going to get you to the same page as what I intended.

NOTE

URL Shortening Services

Is.gd is just one of many URL shortening services; others include bit.ly, goo.gl, and TinyURL.com. I like is.gd because the service is free and the owner operates with high integrity. For more information on is.gd, visit their FAQ page: http://is.gd/faq.php#owner.

TIP

Notes, Tips, and Cautions

This book uses the Note formatting (see the previous URL Shortening Services note) to frame supplemental content that adds to the current topic of discussion. Or, perhaps the Note represents a clarification or an expansion of the information. This extra information could also be formatted as a Tip, which identifies tips, tricks, or other pieces of expert advice, or a Caution, which warns you of potential hazards. Call it potpourri!

About the Code Images

For most Try It Yourself exercises, you'll see one or more source code images that are annotated with alphabetical letters. The Try It Yourself steps are then cross-referenced with parts of each code image. Hopefully, you find this format convenient to your learning. Remember not to fall into the trap of blindly copying the provided code; instead, remember that learning to program requires (yes, *requires*) lots and lots of trial and error.

System Requirements

You don't need a heck of a lot, computer-wise, to perform all of the Try It Yourself exercises in this book. However, if you do not meet the necessary system requirements, then you are stuck. To that end, make sure that you have the following met prior to beginning your work:

- **A standard computer**: It doesn't matter whether your computer runs Windows, OS X, or Linux. Likewise, you can use either a desktop or laptop computer. However, Scratch 2.0

won't run on any device that does not fully support Adobe Flash. That rules out, at the least, iOS devices such as iPhones, iPads, and iPod touches.

- **An Internet connection**: Scratch 2.0 is a web application, so you need to be connected to the Internet to complete the exercises. Yes, in Hour 18, you'll learn about the Scratch 2.0 Offline Editor. However, the Offline Editor does not support all Scratch 2.0 features, and in this book you learn to use *all* of the features in the product.
- **An Adobe Flash-enabled web browser**: Again, it doesn't matter whether your web browser of choice is made by Microsoft, Apple, Google, or another vendor—what does count is whether the browser has the Adobe Flash plug-in installed. Point your browser to the Adobe website (http://is.gd/pCNCvd) to perform a check; if you don't have the plug-in, you can install it at Adobe.com for free.

Okay—that's enough of the preliminaries. It's time to learn to program with Scratch 2.0!

This page intentionally left blank

HOUR 5
Using Looks Blocks

What You'll Learn in This Hour:

- Getting to know the Looks blocks
- Interacting with the player
- Getting sprites to "talk" to each other

The main theme for this hour is that of involving your user in your Scratch project. I've always enjoyed video games of the first-person shooter (FPS) variety; however, I have remarkably low patience for long, drawn-out cut scenes that have no player interaction. I just want to get to the good stuff and to start playing the darned game!

Likewise, you should always keep your player in mind as you develop your Scratch projects. Believe me, they don't want to sit there twiddling their thumbs while you present all of your nifty animations. Instead, they want to control the behavior and perhaps the outcome of the project—and you can make that happen.

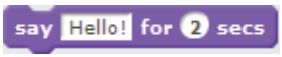
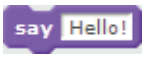
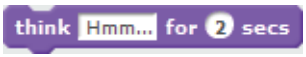
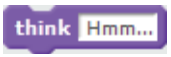
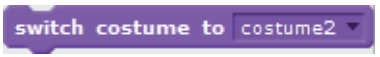
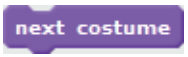
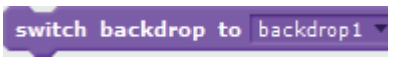
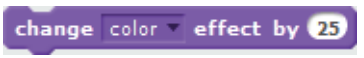
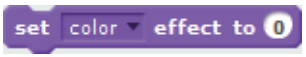
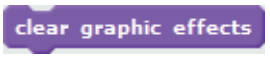
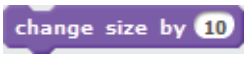
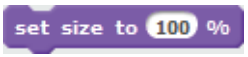
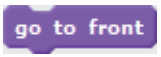
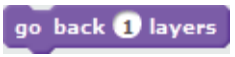
The Looks block palette in Scratch 2.0 includes lots of goodies that put the power and control into your players' hands. In this hour, you learn how to communicate with your player both by using speech and thought bubbles. You also learn how to ask the player questions, get his or her feedback, and act accordingly on that feedback.

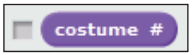
By the end of this hour, you'll have the ability to grow, shrink, fade, or otherwise modify your sprites in novel and effective ways. Let's get to work.

Getting to Know the Looks Blocks

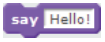
You know the drill by now: Fire up a new, blank project, select the Cat sprite, and click the Looks tab from the Scripts palette. Next, study Table 5.1 to familiarize yourself with the purple Looks blocks.

TABLE 5.1 Scratch Looks Blocks

Block Image	Block Type	Function
	Stack	Makes a time-limited speech balloon appear above the selected sprite
	Stack	Makes a persistent speech balloon appear above the selected sprite
	Stack	Makes a time-limited thought balloon appear above the selected sprite
	Stack	Makes a persistent thought balloon appear above the selected sprite
	Stack	Displays the sprite immediately
	Stack	Hides the sprite immediately
	Stack	Changes the sprite's costume to another one
	Stack	Transitions the sprite from the current costume to the next one in the sprite's costume list
	Stack	Changes the Stage backdrop to a particular one
	Stack	Changes the current value of a particular sprite's graphical effect by a percentage; the effects are color, fisheye, whirl, pixelate, mosaic, brightness, and ghost
	Stack	Sets the sprite's effect to a percentage
	Stack	Restores all of the sprite's graphical effects to their default zero value
	Stack	Changes the sprite's default size (100) to a larger or smaller value
	Stack	Sets a sprite's size to a given magnification level
	Stack	Brings the sprite to the top layer of the Stage "stack"
	Stack	Moves the sprite back one layer on the Stage

Block Image	Block Type	Function
	Reporter	References the sprite's current costume ID number
	Reporter	References the current backdrop's Name property
	Reporter	References the sprite's current size

Make sure that you have the Scratch Cat sprite selected on Stage and spend some time double-clicking each of the Looks blocks. You'll observe that you can try out that block's functionality without having to compose an actual script.

For instance, double-click the  block. Now change the "Hello!" text to something else and test it out again.

This "click and try it" procedure is especially fun with the graphical effects stack blocks. Have fun, friend—that is largely what computer programming is all about!

By way of review, recall that stack blocks have a notch on top and a bump on bottom, just like a jigsaw puzzle piece. This is meant to indicate that you can easily stack these blocks together, one after the other, to chain actions and trigger events.

Reporter blocks are shaped like flattened-out ovals and hold values. You use reporter blocks by dropping them into a space within another block.

Boolean blocks (shaped like flattened-out hexagons) are a special type of reporter block. Whereas reporter blocks can hold alphanumeric data, Boolean blocks can contain only True or False values. You can insert Boolean blocks into the appropriately shaped holes in other Boolean blocks.

Hopefully, you are beginning to see the beauty and logic in how Scratch uses color-coded blocks to help you think like a programmer without getting bungled up in arcane syntax.

NOTE

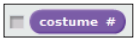
Imitation Is the Sincerest Form of Flattery

You should know that Scratch is no longer the only game in town (pun most certainly intended) with regard to block-based, learning programming environments. Google released Blockly (http://cbt.gg/1969Kus) as a web-based, graphical programming editor whose color-coded blocks look and behave suspiciously like those in Scratch.

As you can imagine, some members of the Scratch community have mixed feelings about Google's "appropriation" of Scratch's main design and usage paradigms. On the other hand, competition

between software vendors can often lead the way to increased innovation. For a nice comparison among most of today's graphical and/or block-based programming learning environments, check out Alfred Thompson's Computer Science Teacher blog at http://cbt.gg/1969T10.

For the first project example in this hour, let's play around with a sprite's appearance. To set the foundation for this exercise, do the following in the Scratch Editor:

- Rename the Scratch Cat to Cat.
- Import the Stage backdrop named route66 and make it the active backdrop.
- Turn on the 

and 

reporter blocks to make them appear as Stage monitors. To do this, simply enable the check box to the immediate left of each reporter block in the Looks block palette. You can see what this looks like in Figure 5.1.

FIGURE 5.1
Stage monitors make Scratch reporter data visible on the Stage. This is extraordinarily useful when you debug your project code or when you want to display information (such as a game score) to the user.

TRY IT YOURSELF

Lookin' Good!

In this Try It Yourself exercise, you become familiar with how you can dynamically alter the appearance of a sprite by using Looks blocks. You start by having the Cat "say" something to the player, and then you make the sprite do a bunch of contortions, including growing, recoloring, warping, spinning, and then fading out.

The completed solution file is named `Hour05a.sb2`.

Complete the following steps using Figure 5.2 as a guide, which shows you the code in context:

1. Bring out a Green Flag block and ignore the code section labeled A in Figure 5.2 for now. Note the say and think blocks; the difference between these is the appearance of the bubble that appears by your sprite. For the purposes of this exercise, you want the Cat to say "Watch this!" for 2 seconds (see the code section labeled B in the figure).

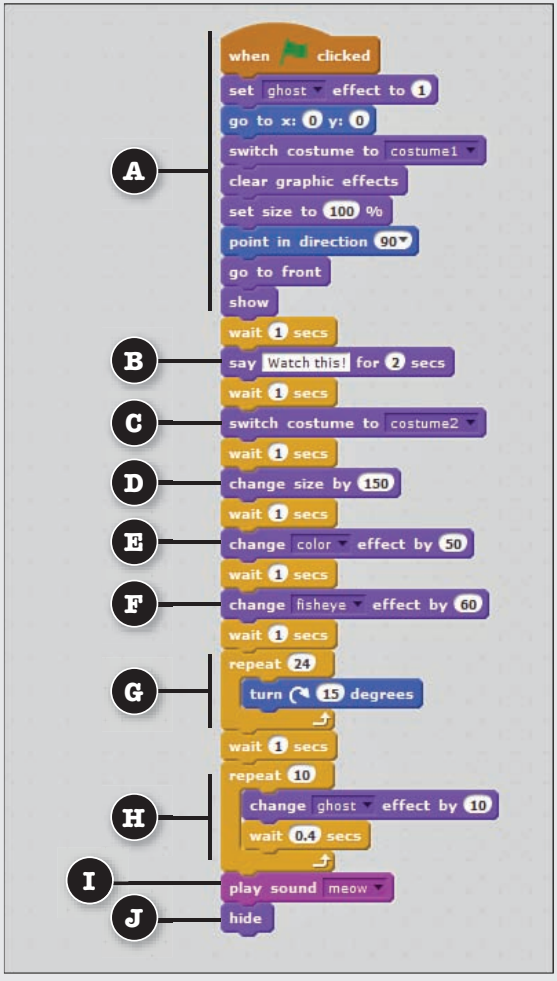
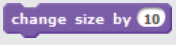
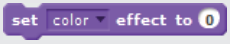


FIGURE 5.2
Source code for the Lookin' Good! Try It Yourself exercise.

2. After a 1-second pause, switch the Cat's costume (see the code section labeled C in the figure). Remember that the wait blocks are found in the Control palette.
3. Sprites start out at a size value of 100. Thus, if you bring out a 

block and use a value of 150 (see the code section labeled D in the figure), the sprite will grow by 150 percent. In other words, the sprite's size will go from 100 to 250.

4. Change the sprite's color effect by a factor of 50 (see the code section labeled E in the figure). A sprite's default color effect is 0, so any value you add or subtract alters the sprite's shade. A single sprite costume can take on 200 different color schemes by using the 

block. Thus, if you set the change color effect block to 200, you'll see no difference in the sprite's color.

5. Take a moment to just play around with other effects (see the code section labeled F in the figure). The fisheye effect is pretty cool; the higher you set the value about the default value of 0, the more warping you see in the sprite.

6. Spin the sprite in a 360-degree (full) rotation (see the code section labeled G in the figure). There are probably several ways in which you could accomplish this goal. For my money, repeating a 15-degree turn 24 times (perform the arithmetic; you'll find that 15 multiplied by 24 equals 360) gets the job done efficiently enough.

7. Instead of having the Cat simply disappear with a 

block, add some pizzazz to the project (see the code section labeled H in the figure). The ghost effect is excellent if you want to fade in or fade out a sprite. Here, increase the sprite's ghost (transparency) effect by 10, 10 times. In Scratch 2.0, a costume can have 100 different transparency levels. If you run a +10 ghost level 10 times, then by the end of the loop the sprite is fully transparent.

8. You'll get into adding and managing audio in your Scratch projects in the next hour. For now, simply play the default meow sound to signify the conclusion of the project (see the code section labeled I in the figure).

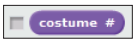
9. Add a hide block just for grins (see the code section labeled J in the figure). In programming, being explicit with your code is generally superior to being implicit.

10. Now return to the code section labeled A in Figure 5.2. This is discussed last so the blocks used make sense to you. If you try to rerun the project without this "cleanup" code, the project will look a mess and be pretty much unusabvle.

    It can't be stressed enough how important it is that you put code at the very front of your Scratch project that resets the environment. Here, you are resetting the sprite's ghost effect, Stage position, costume, size, directionality, and visibility. Strictly speaking, the set ghost effect block isn't needed in addition to the clear graphic effects block, but it's added here for completeness.

What do you think of the Stage monitors? Pretty cool, aren't they? Here's something else for you to try: Double-click each Stage monitor and notice what happens.

Sometimes the value you are reporting in a Stage monitor doesn't need a label, or perhaps you added the label to the Stage itself. For instance, I like the larger, no-label view in some of my games.

You won't need monitors for the rest of this hour, so feel free to return to the Looks palette and uncheck the 

and 

reporter blocks.

Interacting with the Player

I don't know about you, but when I'm playing a game or interactive presentation, I want to do something. The last thing you want to do as a Scratch developer is to bore your players.

Thus, the more options you give to your players, the more you ask (or require) them to take action on their part, the more involved they'll be in your project.

One great way to interact with the player is to have a sprite "ask" the player for input.

By prompting the user for input and then adding programming logic to react to that input, you accomplish many goals, including the following:

- The project becomes more dynamic instead of the same exact thing every time that it is run.
- The user feels that the project is personalized for him or her.
- The project has a longer "shelf life" because it has more than one outcome, and the outcome is at least partially dependent upon user input.

Here's how it works: First, navigate to the Sensing palette and bring out an 

block. You can add any text you want to the admittedly small text area.

During your program execution, a prompt box will appear at the bottom of the Stage, allowing the user to type some data and press Enter.

That answer from the user is captured and stored in the 

reporter block.

NOTE

Player, User, or Something Else?

This text has stressed to you many times the importance of coding your Scratch project with the user in mind. Just to be clear: When the text refers to "the user" or "the player," it's referring generically to the individuals who will access your project on the Scratch website.

TRY IT YOURSELF

Ask the User

In this Try It Yourself exercise, you have the Scratch Cat ask the player if he or she wants the Cat to grow or shrink in size. Depending upon the player's answer, the Cat then obeys the player's command.

You should create a new project that is set up the same way as what you had in the previous exercise. One important change: Add a second backdrop to the Stage, and name it end. You can use the Fill tool, which you'll learn about in Hour 21, "Creating Your Own Sprites and Backdrops," to create a black screen. You can then use the Text tool to add a simple "The End" banner, which you can see in the solution file.

The completed solution file is called `Hour05b.sb2`.

Complete the following steps, using Figure 5.3 as a guide, which shows you the code in context:

1. Make sure that the Cat sprite is selected, and head over to the Scripts area. In the code section labeled A in Figure 5.3, you have the "cleanup" code that ensures that the sprite shows up in the same spot and is the same size every time that the Green Flag is clicked.

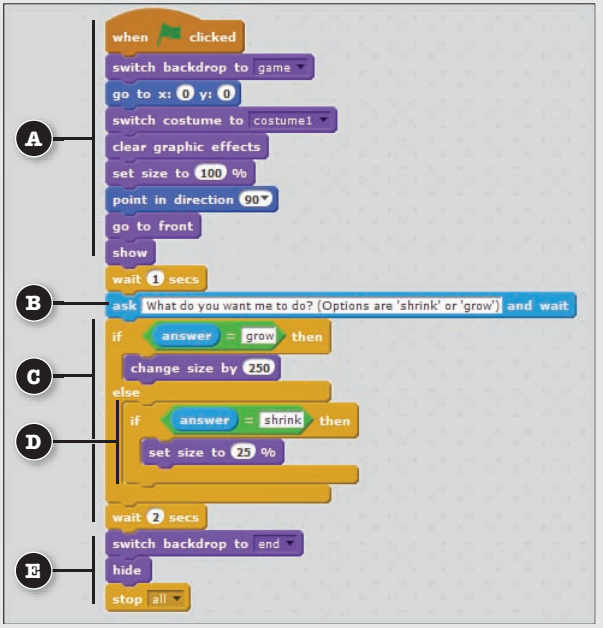


FIGURE 5.3
Source code for the Ask the User Try It Yourself exercise.

2. You want to be as descriptive as possible when you bring out the ask block (see the code section labeled B in the figure). For instance, adding What do you want me to do? (Options are 'shrink' or 'grow') tells the player exactly what is expected of him or her.

3. The main "engine" of this project occurs in the code section labeled C in the figure. You want to test for two conditions. The first condition will say "If the player types 'grow,' then the sprite should grow 250 percent. If the player types 'shrink,' then the sprite should shrink to 25 percent of its original, default size."

   You can turn that pseudocode into real code by using the if else Control block.

4. You can embed if or if else C blocks to test for more than one condition. In the code section labeled D in the figure, if the test for *grow* fails, then you proceed to the second, embedded if statement. Here, you catch the event of the player typing *shrink*.

5. Because the code in the code section labeled E in the figure needs to run regardless of whether the player grew or shrunk the sprite, you place these blocks outside of the C block structure. In this case, you switch to your second backdrop, hide the Cat sprite, and stop program execution.

As you worked through the Ask the User Try It Yourself exercise, you probably had the thought, "What if the user were to type something other than *grow* or *shrink*?"

If you did have that question, then good for you! That's what it takes to think like a computer programmer. This case of the user typing something you don't anticipate is called an *exception*. Unhandled exceptions are one of the biggest reasons why programs crash.

Asking a user for input, as you can see in Figure 5.5, can be risky because that user can possibly submit invalid or unhandled input to your project.

Exception handling is covered in greater detail in Hour 19, "Troubleshooting Your Project"; for now, take a look at Figure 5.4 to see one way to handle this particular exception.

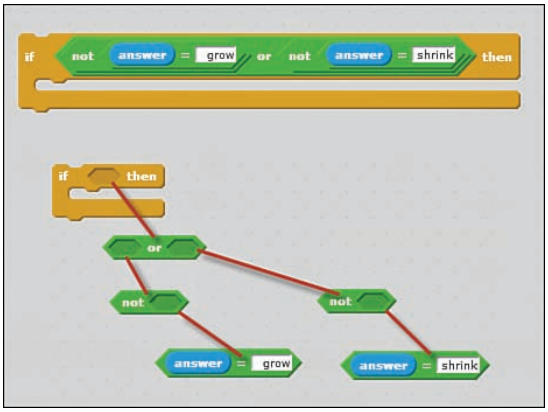


FIGURE 5.4
Trapping exceptions with Scratch blocks might not always be pretty, but it is possible. This code picks up the case where the player enters anything other than the required keywords.

FIGURE 5.5
Asking the user for input is a great way to build buy-in and interest for your Scratch project.

You need to become comfortable with the notion of nesting reporter blocks into Boolean blocks, and Boolean blocks into other Booleans. Remember that reporter blocks store variable data; you can pop them into any block that has a white, rectangular cutout.

By contrast, Boolean blocks look like flattened hexagons and have hexagonal cutouts as well as reporter cutouts.

Getting Sprites to "Talk" to Each Other

A common question among Scratchers who have made some progress in their work is, "How can I get sprites to affect each other?" For instance, what if you want Sprite B to move when Sprite A touches it?

In Scratch, you use *broadcasts* to pass messages among sprites. You can even communicate between sprites and the Stage by sending and receiving broadcasts.

You will find the broadcast message1 and broadcast message1 and wait blocks in the Events palette.

You'll also see a when I receive message1 Hat block that you use to catch outgoing broadcasts.

Trust me—before too long, you'll appreciate how powerful broadcasts are; they will be an indispensable addition to your Scratch programming toolkit.

Here are a few key points to keep in mind regarding Scratch broadcast messages:

- A sprite can both send and receive the same broadcast message.
- Broadcast messages can be received by all sprites (and the Stage).
- Broadcasts are most commonly used to (1) connect different events, (2) run two scripts in the same frame, and/or (3) prepare a scene with multiple sprites.

NOTE

Can You Broadcast to Specific Sprites?

Unfortunately, Scratch 2.0 has no built-in method for using broadcasts to target specific sprites. As you've seen, the default behavior is to make broadcast messages available to all assets in a project.

However, the adventurous can hop on over to the Scratch Wiki (http://is.gd/wwryoS) to learn a workaround (less charitably called a *hack*) to this behavior.

Essentially, you can tag each sprite with a unique ID by using private variables and then define a global variable that determines who should receive a particular broadcast.

If this procedure sounds frighteningly complex, don't worry about it for now. After all, you have yet to work with variables—you will, though, trust me!

To get set up for the Move from Room to Room Try It Yourself exercise, fire up a new, blank project that contains the following assets:

- **Sprite**: Default Scratch Cat; rename sprite to Cat.
- **Sprite**: Magic Carpet (look in the Transportation section of the Sprite Library); rename to Carpet.
- **Stage backdrops**: Add the room1 and room2 backdrops from the Indoors category of the Backdrop Library. You can delete the default backdrop for this exercise.

TRY IT YOURSELF ▼

Move from Room to Room

For this Try It Yourself exercise, you try your hand at a very brief and very simple interactive story. You first have the Scratch Cat move from one room to another (a very cool trick that you'll enjoy). Next, you have the Cat ask the user to press a key on the keyboard to move a second sprite (the Carpet) out of the way.

The Cat and Carpet scripts are illustrated in Figures 5.6 and 5.7, respectively.

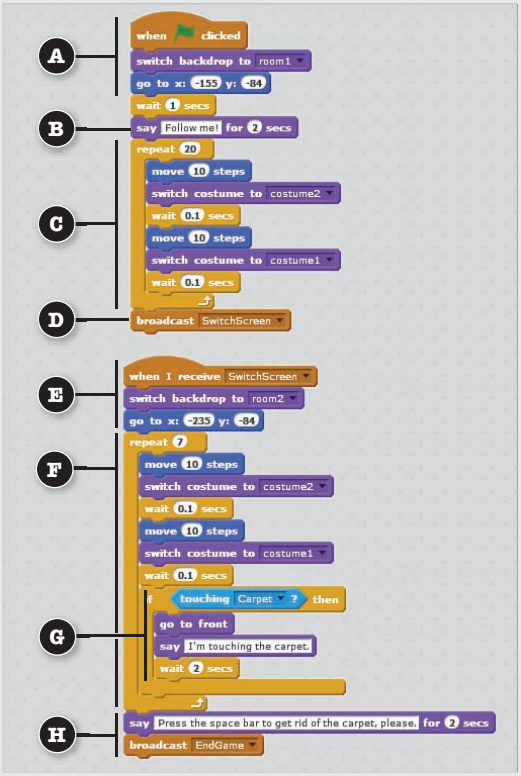


FIGURE 5.6
Cat sprite source code for the Move from Room to Room Try It Yourself exercise.

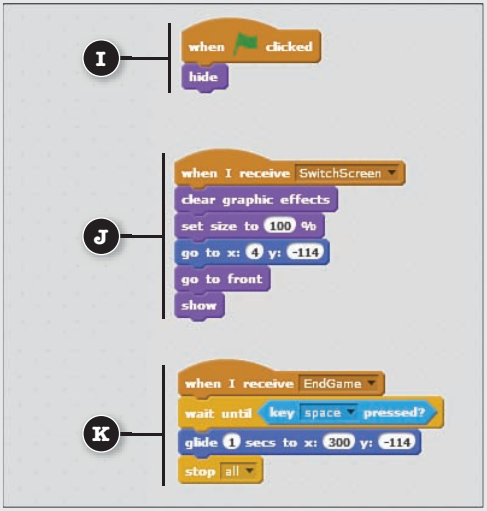
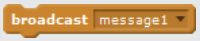


FIGURE 5.7
Carpet sprite source code for the Move from Room to Room Try It Yourself exercise.

The completed solution file is named `Hour05c.sb2`. Work through the following steps in order to complete this exercise. Let's get to work!

1. Begin by coding the Cat's scripts. Reset the story environment by ensuring that the room1 backdrop is active and the Cat is placed appropriately on the Stage, as shown in the code section labeled A in Figure 5.6.
2. Have the Cat say something to the player to help him or her feel more engaged with the story (see the code section labeled B in Figure 5.6).
3. The repeat block shown in the code section labeled C in Figure 5.6 defines the Cat's animation across the first backdrop. How to do this was covered earlier in the book; simply move the sprite 10 steps, switch its costume, and insert a short pause.

4. Now you get to the heart of the matter. Bring out a 

block, open up the drop-down, and select New Message from the menu. You can name a broadcast anything; for this purpose, make it SwitchScreen. (See the code section labeled D in Figure 5.6.)

5. In the code section labeled E in Figure 5.6, you see something that might strike you as surprising; namely, that sprites can actually "listen for" and receive their own messages! In just a moment, you have the Carpet sprite listen for this broadcast as well.

   For now, switch the Stage backdrop to the second room and place the sprite appropriately on the Stage. This action, combined with the previously defined sprite movement, gives the player the impression that the sprite traveled from one room to another.

6. Add in some animation blocks to get the Cat moved to the Carpet (see the code section labeled F in Figure 5.6). (We haven't addressed the Carpet yet, I realize that.)

7. Insert an 

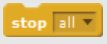
C block to test for the Sensing condition where the Cat sprite touches the Carpet sprite. If that statement evaluates to True, then bring the Cat sprite to the very top layer on the Stage and have it say "*I'm touching the carpet.*" (See the code section labeled G in Figure 5.6.)

8. After a 2-second pause, have the Cat sprite say "*Press the space bar to get rid of the carpet, please.*" You then need to create a second broadcast named EndGame that will be picked up by the Carpet. (See the code section labeled H in Figure 5.6.)

9. Now switch your focus to the Carpet sprite and cross-reference these steps with Figure 5.7. Place the Carpet sprite at (4, -114) on the Stage. Reminder: Coordinates are given as (x,y). Simply hide the Carpet until the Cat reaches the second room (see the code section labeled I in Figure 5.7).

10. Next, listen for the SwitchScreen broadcast. At that time, reset the style and position of the Carpet sprite, and explicitly show it on the Stage. (See the code section labeled J in Figure 5.7.)

11. Finally, listen for the EndGame broadcast, which is, you'll remember, the broadcast that is triggered if the Cat sprite touches the Carpet sprite. (See the code section labeled K in Figure 5.7.)

    Use the wait until block to test for the key space pressed Boolean condition (that block is found in the Sensing palette). If True, then code the carpet to glide horizontally across the screen to an X value that lies beyond the 480 pixels of the Stage. You then invoke the 

Control block to end the interaction.

Summary

By now, hopefully you are comfortable with the relationships between sprites, their costumes, the Stage, its backdrops, and how you can use broadcasts to connect scripts among sprites and the Stage.

Don't feel discouraged if you feel that you've learned nothing more than "toy code" thus far. You have all the time in the world to build full projects from beginning to end in the final two hours of this book.

Also, there is no way around the sheer time, repetition, and practice that is required for you to know your way around the Blocks palette.

You know that you are well on your way to becoming an honest-to-goodness computer programmer when your mind starts thinking in terms of "Ohh, I just thought of how to have the program do such-and-such" as opposed to, "Now where is the wait for block again?"

The next hour continues the fun. There, you'll learn how to integrate sound effects into your Scratch projects.

Workshop

Quiz

1. When you define a broadcast in Scratch, which assets can receive the broadcast message and therefore take action upon it?

   A. Only the sending sprite

   B. Only the receiving sprite

   C. Only the Stage and its backdrops

   D. The Stage and all sprites

2. Which of the following is a valid way to test for two or more conditions in a Scratch script?

   A. A Boolean block

   B. A nested if C block

   C. A repeat C block

   D. A broadcast block

3. Which of the following Scratch graphical effects can be used to fade in or fade out a sprite on the Stage?

   A. Color

   B. Pixelate

   C. Mosaic

   D. Ghost

Answers

1. The correct answer is choice D. In Scratch 2.0, any broadcasts that are sent out from a sprite or the Stage are receivable by both the Stage as well as all sprites in the project. Recall that the sending sprite can also receive its own message. However, a sprite or the Stage is not obligated to receive a message.

   The Scratch Wiki (http://cbt.gg/1bop3og) includes a hack or workaround that does enable you to effectively target specific sprites with broadcast messages.

2. The correct answer is choice B. By default, an if C block tests for the truth or falsity of a single condition (although you can certainly embed multiple operator blocks—you'll see how to do that in Hour 10, "Using Operators Blocks." An easy approach to solving this problem that you discovered in this chapter is nesting one or more if C blocks inside of the outer, original C block.

3. The correct answer is choice D. The ghost graphic effect works well for fading in or fading out a sprite on the Stage. Changing the color effect can make a sprite look like it's flashing. The pixelate effect makes a sprite look retro or old-fashioned. The mosaic graphic effect is useful for transitioning a sprite between costumes.

Challenge

Okay, here is a fun project for you to try out. This challenge gives you some more experience using graphical effects, implementing broadcasts, and responding to the player's keyboard input.

Here are the design goals for this project:

- Have the Scratch Cat instruct the player to press the spacebar (or another key of your choosing) to transform the sprite into a butterfly.
- Use graphical effects to pixelate the Cat and have it "become" a butterfly.
- Have the butterfly "tell" the gamer that he or she can use the arrow keys to move the butterfly around the Stage.
- Code the butterfly for player control.

Figure 5.8 shows a screenshot of my version of this game, and you can also examine the solution file `Hour05d.sb2`.

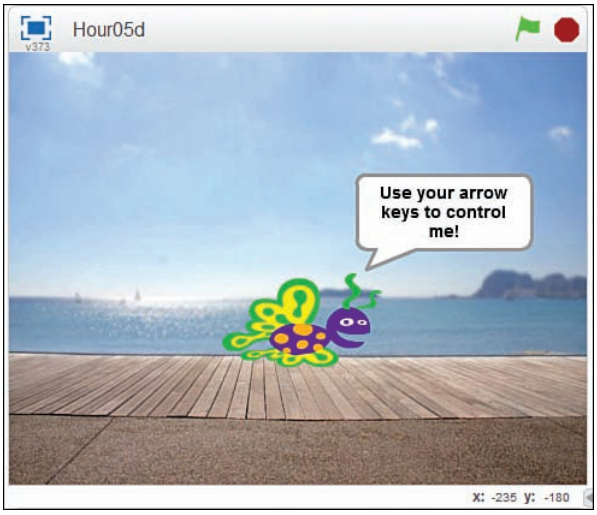


FIGURE 5.8
A screen capture from the Hour 5 Challenge project (`Hour05d.sb2`).

Remember that you need to download and install the free Scratch Offline Editor (http://is.gd/sB2h1k) to view the solution files. You'll learn everything there is to know about the Offline Editor in Hour 18, "Using the Scratch Offline Editor."

Before finishing this hour, though, it's only fair to discuss the pixelate graphic effect just a little because you didn't use it at all earlier in this hour.

As you can see in Figure 5.9, all you have to do is to wrap a change pixelate effect by block inside of a repeat C block. Test out this sample code block on one of your sprites and experiment with different pixelate intensities.

Also, please understand that there is no shame at all in turning to the Scratch project gallery (http://is.gd/tsr9gM) for help when you get stuck. For example, you can search for projects that include the word pixelate in their title or description, or you can do a tag search for the same term.

Most Scratchers are actually complimented when other Scratchers remix their projects. That's what the community effort of Scratch is all about, after all.

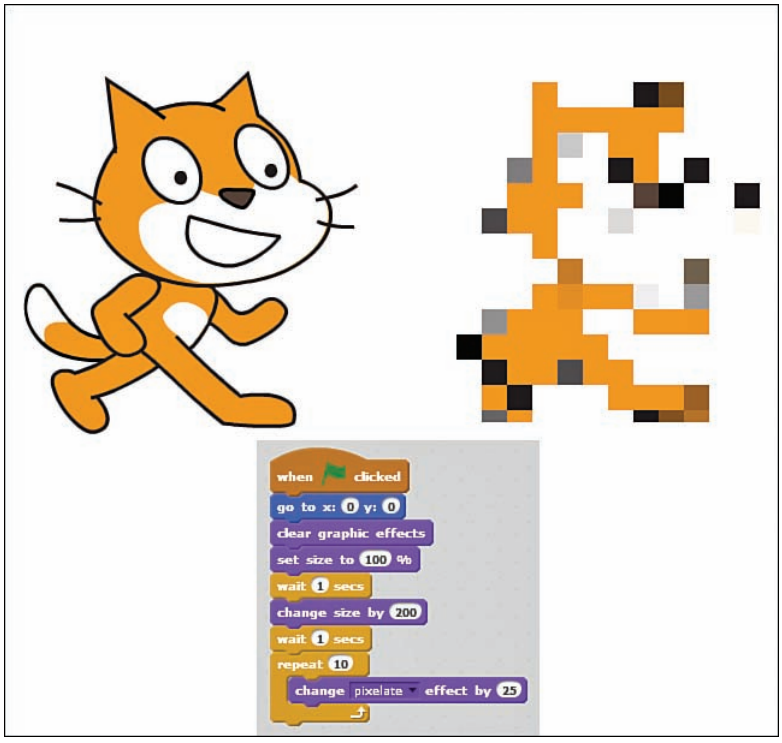


FIGURE 5.9
Pixelating a sprite has a cool retro effect that is useful for transitions.

This page intentionally left blank

Index

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q-R

S

T

U

V

W

X-Z