11 Awesome Projects
written especially for young people!

Adventures in Coding

Eva Holland
Chris Minnick

WILEY

Introduction

ARE YOU A fearless adventurer? Do you like to set sail on new ventures and learn new skills? Do you want to learn how to use technology to turn your ideas into reality? Are you curious about computer programming, but aren't sure where to start? If your answer to these questions is a confident "yes!" then this book is for you!

Like people, computers can be talked to in a variety of languages. Computer programming, or *coding*, is a way for people to talk to computers. Many computer languages are similar to one another, so once you know one programming language, it's much easier to learn another one.

Adventures in Coding introduces you to the world of programming, using Scratch.

What Is Scratch?

Scratch is for anyone new to coding. It's a great place to begin your lifelong adventures in the world of computer programming. Scratch introduces programming concepts in a fun and approachable way. Using simple drag-and-drop features, you can create real computer programs while learning the fundamentals of coding.

Who Should Read This Book?

Adventures in Coding is a great starting point for any young person who is interested in learning how to create games, apps, and art on a computer.

What You Will Learn

Adventures in Coding introduces and guides you through the world of coding using Scratch. You learn the ins and outs of the Scratch universe—from learning about the features of the Project Editor, to connecting with fellow "Scratchers" and sharing projects.

Adventures in Coding teaches you how to create fun games, animate characters, build interactive projects, and more!

How This Book Is Structured

Each chapter of this book is an adventure of its own. With each adventure, you learn a new aspect of Scratch while building on what you already learned. Each adventure finishes with a completed project.

What You Need to Use This Book

The Scratch programming interface lives on the web. All you need to get started and complete the adventures in this book is a computer with a web browser (such as Chrome, Safari, Firefox, or Internet Explorer) and an Internet connection. No experience is expected or necessary. There is nothing you need to purchase or install. Scratch is always free, to anyone!

Conventions

Throughout the book, there are some special boxes to guide and support you. They use the following key:

These boxes explain new concepts or terms.

These boxes provide tips to make your life easier.

These boxes include important things to watch out for.

These boxes further explain the inner workings of programs.

These boxes provide explanations or additional information about the topic at hand.

These boxes point you to videos on the companion website.

You will also find two sets of sidebars in the book. *Challenge* sidebars give you extra suggestions for expanding on the projects in the book. *Digging into the Code* sidebars further explain how some of the more complex programs work.

Companion Website

To download the videos mentioned in this book, visit the companion website at www.wiley.com/go/adventuresincoding.

Contact Us

Your authors, Chris and Eva, would love to hear about your progress in coding. You can reach out to us with questions,

or to show us a cool project you've created, by visiting www.watzthis.com or emailing us at info@watzthis.com.

PROGRAMMING COMPUTERS IS a lot of fun. It’s also a skill that many people see as mysterious and even magical. This chapter unmasks programming to show you just how easy it can be to start your own coding adventure.

Coding Is Everywhere

Computer programming, also known as **coding**, is how people tell computers what to do. What sorts of things can you do after you learn to program? For starters, you can write your own computer games, create modifications (or “mods”) for existing games, program robots to do your bidding, create beautiful computer art and animations, and instruct your computer to play songs! The best part is that the whole time you’re doing all these fun things, you’re learning a valuable skill that is in sky-high demand!

Coding is a common name for computer programming. When you code, you're using a computer language to tell computers what to do.

Can you think of other things that computers can do? Think of all the things that programmers can tell computers to do. There are hundreds, or thousands, of things. Think about all the things you see computers do every day—and not just the fun things. Computer programs are used to create new medicines, design buildings, do complex mathematics, control cars, and so much more.

This is the amazing world in which computer programmers live; we get to solve interesting problems every day and do things that other people see as magic.

Speaking the Language of Machines

All sorts of different people are **programmers**. Programmers come from different places and countries, with different experiences and different training. They speak different languages, have many different interests, and program for different reasons. What they have in common is that they've learned to speak at least one language that is understood by computers.

A **programmer** is a person who writes computer programs.

Computers don’t speak the same languages that people do. People speak languages such as English, French, Spanish, Portuguese, Japanese, and many others. Computers speak machine language. Machine language is a difficult-to-read (for us) language that uses numbers to provide instructions to computers.

If machine language were the only way people could talk to computers, coding would be difficult. Fortunately, people have invented languages, called **programming languages**, which make it easier for people to talk to computers. Here are some examples of programming languages:

- JavaScript
- BASIC
- Perl
- PHP
- Python
- Java
- Visual Basic
- C
- C++
- Scratch

These languages all have one thing in common: They take words and symbols that people understand and translate them into words and symbols that computers understand.

A **programming language** is a language used for giving instructions to computers.

The examples in this book use Scratch. Scratch is a language that was invented at the Massachusetts Institute of Technology (MIT). It was designed to be easy for beginners to learn while using (and teaching) all of the most important things that programmers need to know.

Knowing Your Coding Lingo

You already know some of the lingo of coding. You know that “coding” is just another name for “computer programming,” and you know that people who do computer programming—or coding—are called computer programmers (or coders).

Programming languages, like human languages, are made up of different parts. In English, we have nouns, verbs, adjectives, pronouns, and other parts of speech, not to mention punctuation, and they form sentences and paragraphs. In programming languages, you combine different statements (also known as **commands**) to make computer programs, which are also known as **applications** (or *apps*).

A **command** is an instruction, written in a programming language, that tells a computer to do a task.

An **application** is a set of programming commands that follow each other in a particular order to accomplish tasks. Application is another name for a computer program.

Scratch, and certain other programming languages, use the term **script**. Script is just another name for a program.

A **script** is another term for a computer program that is smaller and more limited than an application.

There are a lot of specialized words in coding that all have very specific meanings, and you'll find that sometimes there are many different words for the same thing. For this reason, we've included a glossary at the back of this book where you can look up or remind yourself of the meaning of terms you're not familiar with.

One of the greatest things about Scratch is that it's easy to dive right into! To get started, you don't need to learn a lot of new concepts or vocabulary. So enough talk! Let's begin!

Writing Your First Scratch Program

When we were growing up, kids didn't care about learning a specific style of dance. Instead of trying to learn complicated dance moves like the cha-cha or the hustle, we ran around like lunatics, jumping off of things, and we sometimes got hurt in the process. Your first Scratch program will be a simulator of an old-fashioned punk rock mosh pit.

Figure 1-1 shows what the finished product will look like. If you imagine the two characters in the figure bouncing off the walls and off each other while a drumbeat plays, you'll have a good idea of the program you'll be learning to make.

Figure 1-1 Your first Scratch program

Joining Scratch

In order to create, save, and share your programs on the Scratch website, you need to use your favorite web browser to visit http://scratch.mit.edu, where you can sign up for a free account. When you go to the website, you see a screen that looks similar to Figure 1-2.

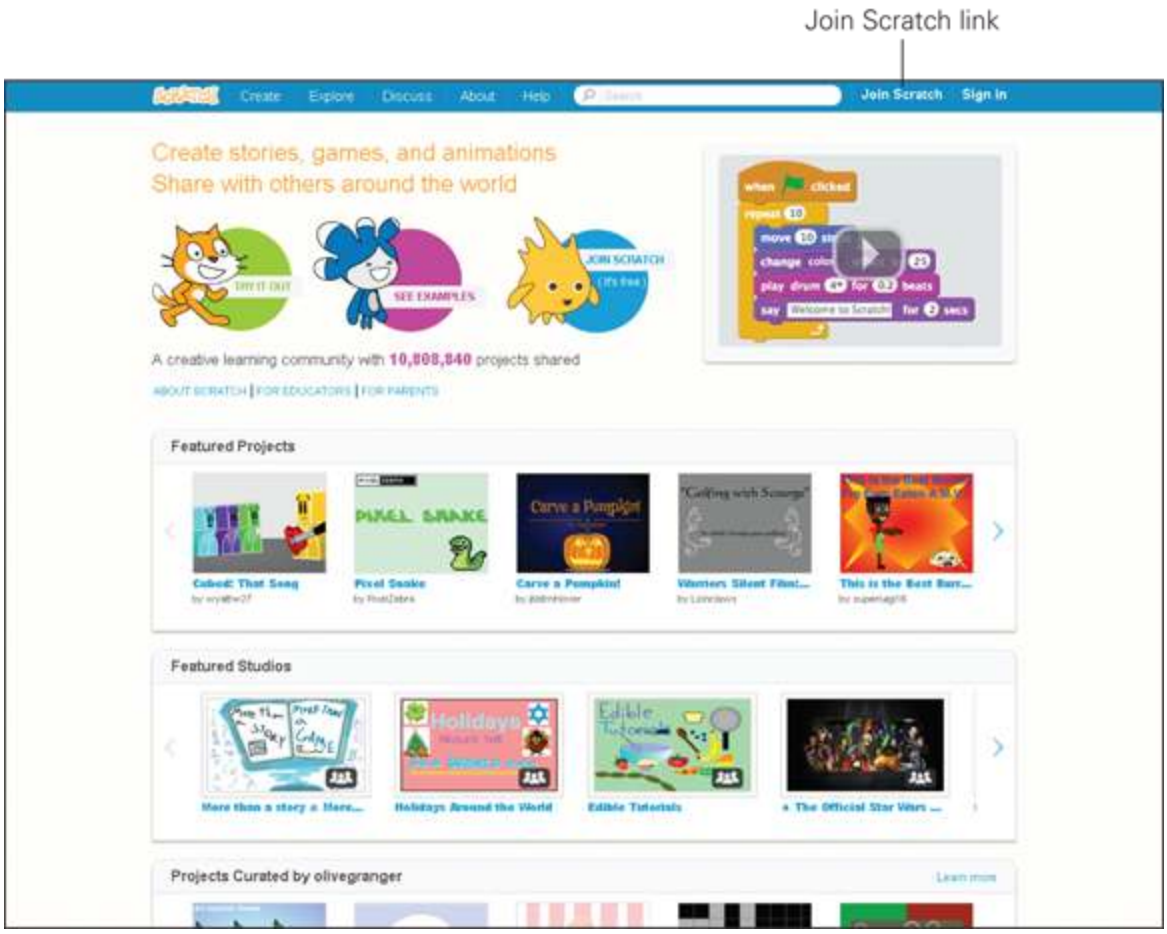

Figure 1-2 The Scratch website

Follow these steps to create your free account:

1. Click the Join Scratch link in the upper-right corner or center of the screen.

   The Join Scratch window opens.
2. Type a username into the field labeled Choose a Scratch Username.

Your username is how Scratch will know you and how other users will see you when you start to share programs. Be creative! Choosing a username can be fun! For everyone's safety, a good username shouldn't reveal any personal information, such as your full name, age, gender, or address. Try personalizing your username by including the name of your favorite sports team or musical group.

3. Choose a password and enter it into the Password and Confirm Password fields.

When creating your password, avoid using information that other people are likely to know, such as your address or birthday. Your password should be something you can remember, but it should also contain numbers or punctuation to make it more secure.

4. Click Next.

   You see the second screen of the signup form.
5. Enter your birthday, gender, and country and then click Next.
6. When you're asked for an email address, enter your email address in both the Email Address and Confirm Email Address fields and click Next.

   Scratch sends a confirmation email to the address you provided.
7. Click the OK Let's Go! button.
8. Check your email. When you get the email from Scratch, click the link in it to confirm your account.

Now you're ready to go. The next section tells you how to start coding!

Meeting Scratch the Cat

After you've joined Scratch and you're ready to start coding, click the Create tab in the top menu of the screen. When the new page loads, you see the scratch Project Editor, which looks like Figure 1-3.

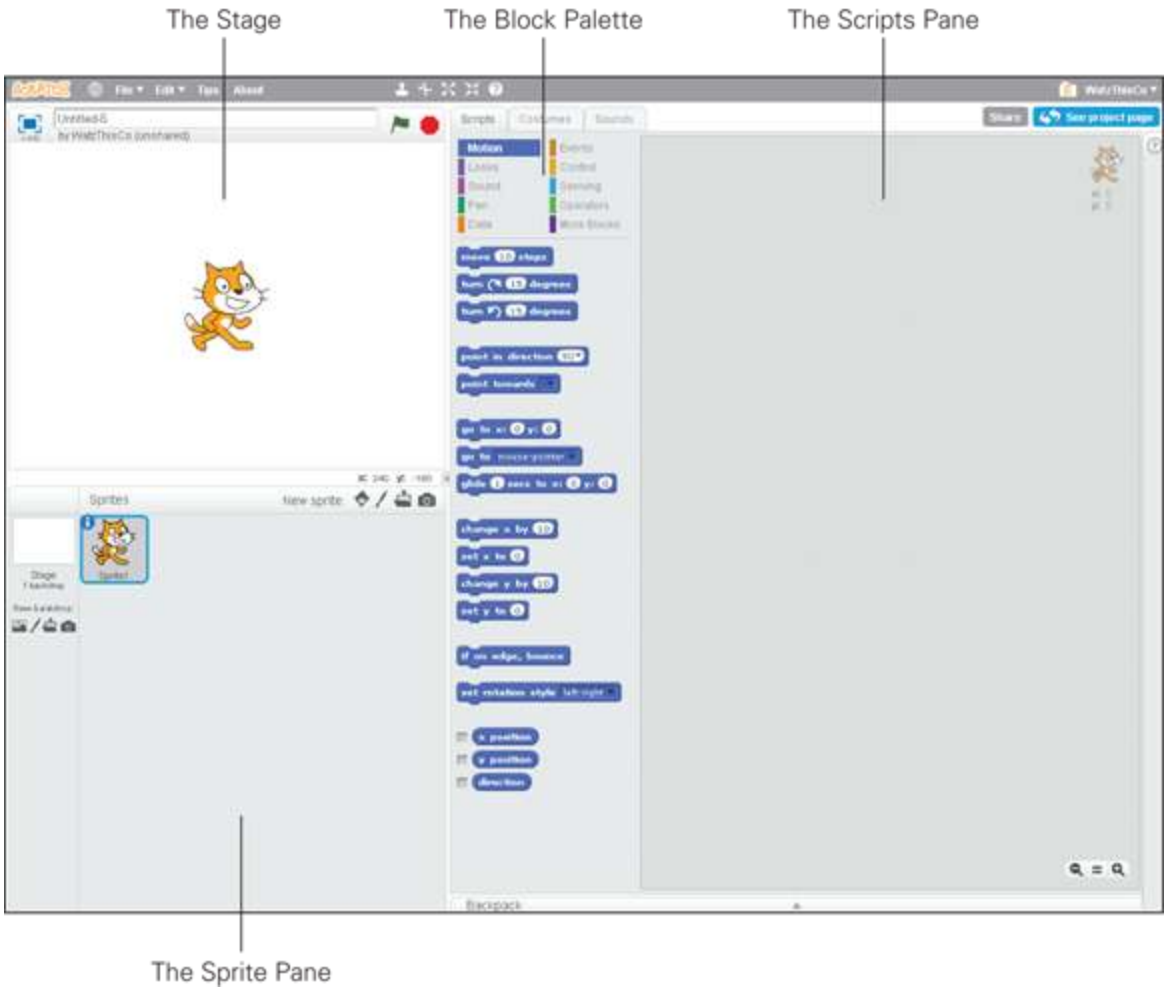

Figure 1-3 The Scratch Project Editor

Don't worry too much about what the things on this screen do. We'll be talking in detail about each part later. For now, let's build something!

See that cool cat in the middle of the screen? Her name is Scratch the Cat. Every new Scratch program starts with her sitting right there, waiting for instructions from you.

The area where she lives is called the Stage. This is where all the action of your program takes place.

Below the Stage is an area called the Sprite Pane. The Sprite Pane shows small images of each of the characters (also known as sprites) in your program.

To the right of the Stage is a rectangle containing differently shaped blocks. This is called the Block Palette. Think of it like a painter's palette, where a painter selects the colors that she wants to paint with before combining those colors to make a painting on a canvas.

To the right of Block Palette is the Scripts Area. This is your canvas, where you put together the blocks selected from the Block Palette to make your sprites do things.

Moving Scratch Around

To get a better idea of how all the pieces fit together, follow these steps to make Scratch the Cat do something:

1. Find the block that says `move 10 steps`. This is the `move ()` block. It looks like Figure 1-4.
2. Click the `move ()` block and drag it into the Scripts Area. Your Scripts Area should now look like Figure 1-5.
3. Double-click the `move ()` block and watch what Scratch the Cat does. Did you see it? She moved slightly to the right.
4. Click the number 10 inside the `move` block to highlight it and change the 10 to 20.
5. Double-click the block. Now Scratch moves twice as far as she did before.
6. Try changing the value in the `move ()` block to an even larger number and see what happens.

Figure 1-4 The `move` block

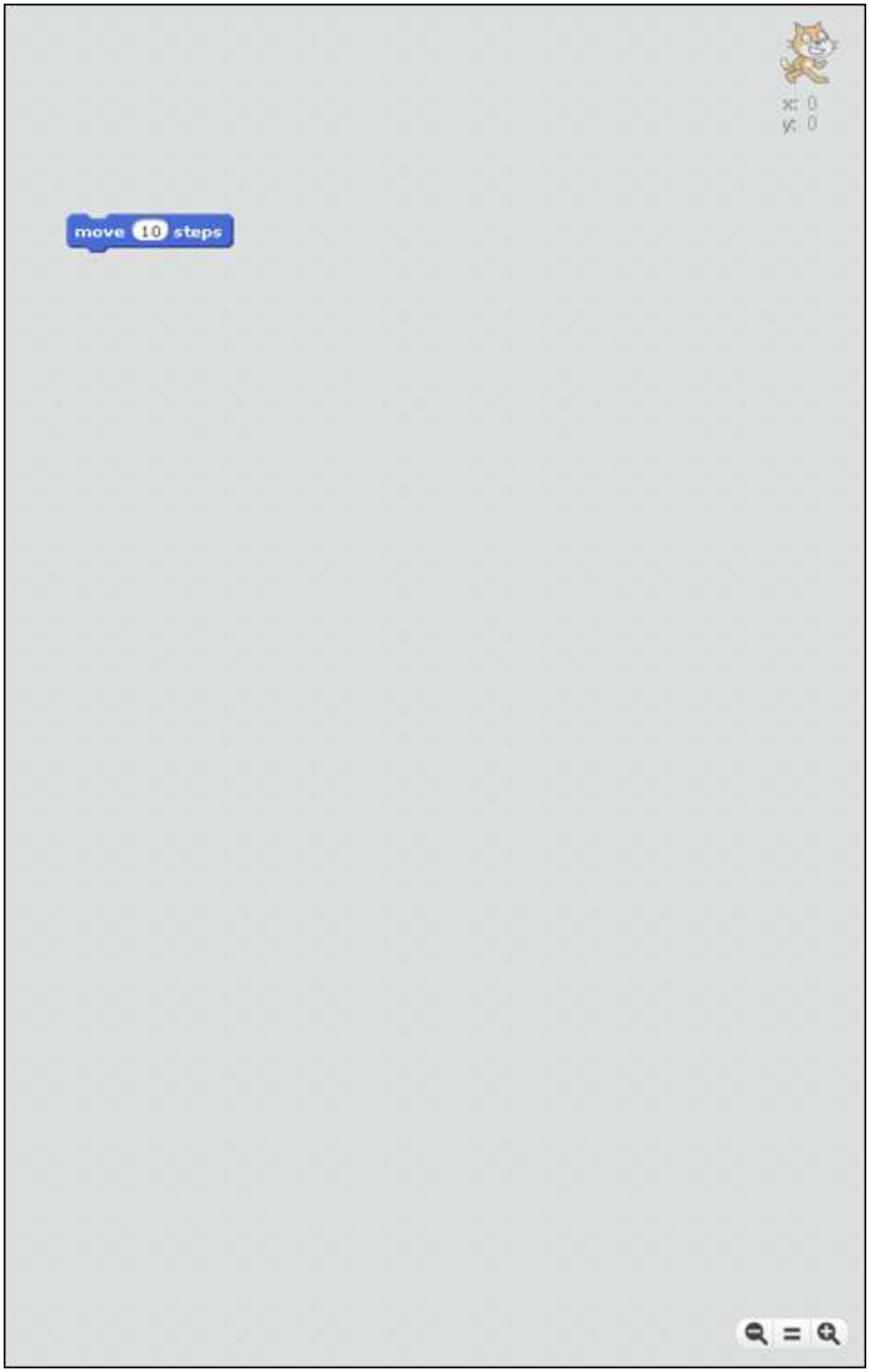

Figure 1-5 The Scripts Area, with one block

Connecting Blocks

Notice that the `move` block looks a bit like a puzzle piece. It works a bit like a puzzle piece too! Any time you see a block shaped like this, you know that you can attach it to another block.

Follow these steps to put some blocks together to make Scratch the Cat do something more complicated:

1. Find the `turn clockwise` block in the Block Palette. Figure 1-6 shows what the block looks like.

   Notice that it has the same shape as the `move` block.
2. Drag the `turn clockwise` block to the Stage and snap it to the bottom of the `move` block. When the blocks are connected, they look as shown in Figure 1-7.

Figure 1-6 The `turn clockwise` block

Figure 1-7 Your first connected blocks

Double-click the combination of blocks and watch what happens on the Stage. The instructions on the first block happen, and then the instructions on the second block happen!

By right-clicking the combination of blocks and selecting Duplicate, from the menu you can create an exact copy of the block combination. Try it out! Then snap the two combinations of blocks together so that they look like Figure 1-8.

Figure 1-8 Duplicating blocks

Now click the whole thing and watch what Scratch the Cat does.

Looping Movements

If you want to make Scratch the Cat do this turning and moving thing over and over, you could keep making as many duplicates as you want of these same blocks, or you could create what's called a **loop**.

A **loop** is a block that causes the commands contained within it to repeat one or more times.

Use the following steps to create a loop:

1. Separate the second two blocks from the first two by clicking the second `move` block and dragging downward.
2. Right-click the second set of blocks and then select Delete from the context menu to remove the block set from the Stage.
3. Click the word Control in the color-coded menu in the Block Palette. Then find the `forever` block (see Figure 1-9) inside the Control Block Palette.
4. Drag the `forever` block to the Stage and place it over the `move` and `turn` blocks so that it snaps around them, as shown in Figure 1-10.

When snapping blocks together and inside of each other, be sure they're completely fitted into place and are not overlapping; otherwise, your script will not run.

5. Double-click the combination of blocks and watch what happens!

   Scratch the Cat keeps moving and turning forever … or until you click the red Stop Sign shown in Figure 1-11.

6. Click the Stop Sign to end the loop.

There are other ways to stop a loop. You learn all those tricks in Adventure 3, "Using Control Blocks"!

Figure 1-9 The `forever` block

Figure 1-10 Wrapping the `forever` block around other blocks

Figure 1-11 The Stop Sign

Starting at the Green Flag

Next to the Stop Sign, you see a Green Flag. This is also known as the Run button. You can press it to start all the action in a program, rather than double-clicking on the blocks.

To enable the Green Flag for your program, follow these steps:

1. In the Block Palette, click Events, as shown in Figure 1-12.

   

   Remember that in Scratch, a **script** is a series of connected commands that cause a sprite to do some task.

2. Drag the `when green flag clicked` block to the Scripts Area and snap it onto the top of your existing script. Your script should now look like Figure 1-13.
3. Click the Green Flag above the Stage to see your program run.

   Scratch the Cat starts running in circles.
4. When you're done watching Scratch the Cat run in circles, click the Stop Sign.

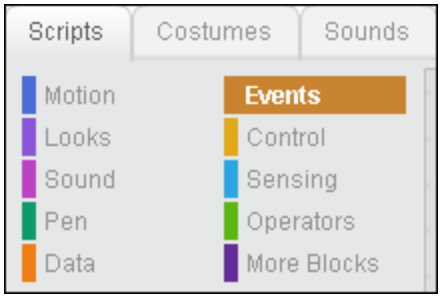

Figure 1-12 Selecting the Events Block Palette

Figure 1-13 Adding the Green Flag event to your program

Bouncing Off the Walls

Moving in circles is cool, but it's time to add some new instructions so Scratch can roam a bit more. Use the following steps to tell Scratch to do more things:

1. Click the `turn clockwise` block and drag it out of the Scripts Area and back into the Block Palette. It disappears from your script, which now looks like Figure 1-14.
2. Click the Green Flag to run the program.

   Scratch runs right off the screen and keeps on running forever!
3. Click the Stop Sign.
4. Click Motion in the Block Palette and drag the `if on edge, bounce` block to the Stage. Snap it underneath the `move` block, as shown in Figure 1-15.
5. Click the Green Flag to see what Scratch does now.

   She runs around, bouncing off the walls, until you click the Stop Sign.

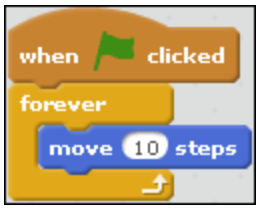

Figure 1-14 The program after you've removed the `turn clockwise` block

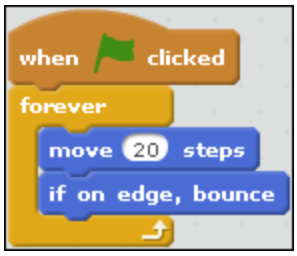

Figure 1-15 Adding an `if on edge, bounce` block

That's more like it! You're getting closer to having a mosh pit, but it's not really a proper mosh pit unless there are other dancers. In the next section, you add a second character to the dance floor.

Creating a Sprite

Look for the Sprite Pane at the bottom of the screen. At the top of it, you see New Sprite: and then some icons. Scratch has numerous built-in characters besides Scratch the Cat that you can use in your programs. You can also create your own characters by uploading a graphic or even by taking a picture!

1. Click the Choose Sprite from Library icon, which is the first icon to the right of New Sprite.
2. Browse through the Sprite Library and find a sprite you like.
3. When you find the sprite you want to use, click it, and then click the OK button at the bottom of the Sprite Library.

Your new sprite is added to the Stage and to the Sprite Pane.

After you add your second sprite, you see that the Scripts Area becomes blank. That's because each sprite has its own Scripts Area that is visible when that sprite is selected. Because you just added this sprite, it doesn't yet have any scripts tied to it.

Press the Green Flag. You see that your first sprite bounces around the screen while your new one just sits there.

To animate your new sprite, follow these steps:

1. In the Sprite Pane, click Scratch the Cat.

   You should now see her script appear in the Scripts Pane.
2. Click the first block in the Scripts Area (the `when green flag clicked` block) and drag it to the Sprite Pane, dropping it on top of your new sprite.
3. Click on your second sprite in the Sprite Pane.

   You see that the scripts from the first sprite have been copied over to the new sprite!
4. Click the Green Flag.

   Now, both characters are bouncing around like maniacs. Fun!

Handling Collisions on the Dance Floor

You may have noticed a strange thing about how your two sprites interact. In real dancing, when two people bump into each other, a collision happens and they bounce away from each other. It's time to make these sprites collide and bounce!

Select one of your sprites in the Sprite Pane. Because they have the same script now, it doesn't matter which sprite you choose. You're going to add the collision script to one of the sprites and then copy it to the other.

1. Click the Control Block Palette to open it.
2. Look for the `if … then` block. It looks like Figure 1-16.
3. Drag the `if … then` block to the Stage so that it's snapped to the bottom of the `if on edge, bounce` block, as shown in Figure 1-17.

Figure 1-16 The `if … then` block

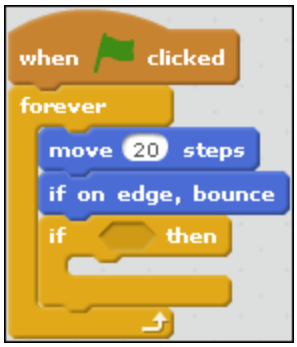

Figure 1-17 Adding an `if … then` block to the script

Next, you're going to use a new type of block—a Sensing block—to detect whether the sprites are touching.

1. Click the Sensing Block Palette.
2. Drag the `touching ()` block into the hexagon-shaped empty spot in the `if … then` block.

   The empty spot expands to fit the `touching` block, and the new block snaps into place, as shown in Figure 1-18.

Notice that the `touching ()` block has a drop-down menu inside of it.

3. Click the drop-down menu inside the `touching ()` block and select the name of the other character on the Stage.

   Now, whenever this sprite is touching the other sprite, it does the actions inside of the `if … then` block.
4. Click the Motion Block Palette to open it.
5. Drag the `turn clockwise` block from the Motion Block Palette and snap it inside of the `if … then` block.
6. Drag the `move` block to the Scripts Area and snap it to the bottom of the `turn clockwise` block.
7. Change the value in the `turn clockwise` block to 20 degrees.
8. Change the value in the `move` block to 30 steps.

   Your program now looks like Figure 1-19.
9. Copy the script to the other sprite by dragging it and dropping it onto the thumbnail image of the other sprite in the Sprite Pane.
10. Click the other sprite in the Sprite Pane.

    You see both the old script (without the `if … then` block) and the new one. If it looks like there's only one script, click and drag on the script to find the other script hiding behind it.
11. Right-click the old script and choose Delete from the context menu to remove the old script.
12. Finally, change the value in the drop-down menu inside the `touching` block to the name of the other sprite, so that each sprite turns and moves when they bump into the other sprite.
13. Click the Green Flag to start the crazy dancing!

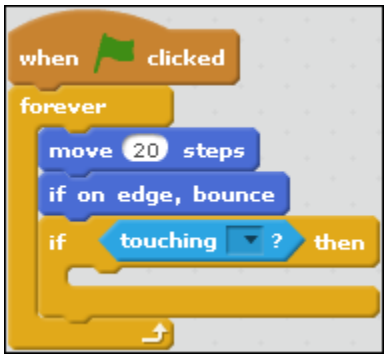

Figure 1-18 Adding a Sensing block to the `if … then` block

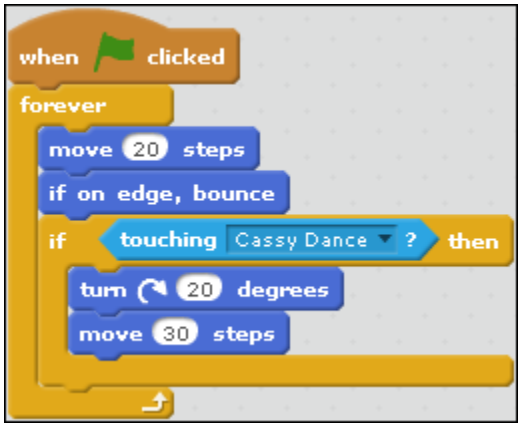

Figure 1-19 The dancing script, with collisions programmed

Slowing It Down

This dancing is way too fast. If this keeps up, the city council is for sure going to pass a law banning dancing! Follow these steps to slow down the dancing.

1. Go to the Control Block Palette and find the `wait` block. It looks like Figure 1-20.
2. Drag the `wait` block into the Scripts Area and snap it in above the first `move` block.

   The `wait` block should now be the first block inside the `forever` block, as shown in Figure 1-21.

   The `wait` block does just as it says: It causes the sprite to wait a certain number of seconds (or a fraction of a

second) before performing the next command. If you only want to slow down the dancing, not bring it to a stop, you can change the value in the `wait` block to a very small amount of time.

3. Click the oval inside the `wait` block to highlight the number and change it to 0.1.
4. Click the Green Flag to see that one of the sprites is now moving much more slowly than the other.
5. Apply the same wait change to the other sprite.

Figure 1-20 The `wait` block

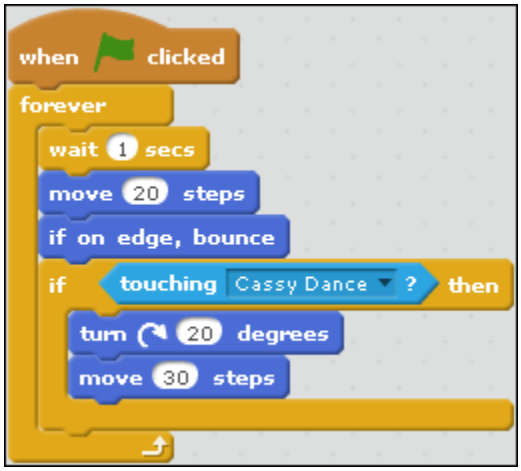

Figure 1-21 Adding the `wait` block to the script

Congratulations! You've built your first computer program! In the next part of this adventure, we take you on a tour of the Scratch Project Editor. You're already somewhat familiar with it from building the Mosh Pit Simulator, but read on, because the next few pages tell you the lingo and the ins and outs of Scratch that you need to know in order to build exciting projects.

Learning the Scratch Environment

Beginning with version 2.0, the Scratch Project Editor is available as an online application as well as an offline application. What this means is that you can download Scratch to your computer and install it, or you can program with Scratch through the website at http://scratch.mit.edu.

The benefit of using the offline version is that you don't need an Internet connection. Also, the online version sometimes has slowdowns and hiccups when a lot of people are using it at the same time. The offline version is immune to these problems.

In this book, we work with the online version. As you'll find out, the online version of Scratch has a ton of fun collaborative and sharing features that the offline version doesn't. Programming online makes it possible for you to share your work with your friends and to make new friends who can help you become a better coder.

If you want to install the offline version, check out the instructions in Appendix A (way in the back of the book). To complete most of the adventures in this book, both the offline or online versions of Scratch work great, and both are completely free to use.

Now that you understand the differences between the two versions of Scratch, let's start finding out how they work!

Exploring the Scratch Project Editor

Whether you go to http://scratch.mit.edu and click Create in the top menu bar or whether you're using the offline editor,

the place where you do all your coding in Scratch is called the Scratch Project Editor.

The Scratch Project Editor is split into different sections. When you first open it, there are two sections on the left, one long one in the middle, and one on the right. We call these panes. You can think of them as being like panes of glass in a window. Unlike window panes, however, you can easily change the size of some of these panes.

When you see a small arrow on the frame between two panes, you can click that arrow to change the size of a pane.

Notice that there’s an arrow between the long skinny pane (which is called the Block Palette) and the two panes on the left (the Stage and the Sprite Pane). Click that arrow now. The pane on the right, the Scripts Area, becomes larger while the panes on the left get smaller, as shown in Figure 1-22.