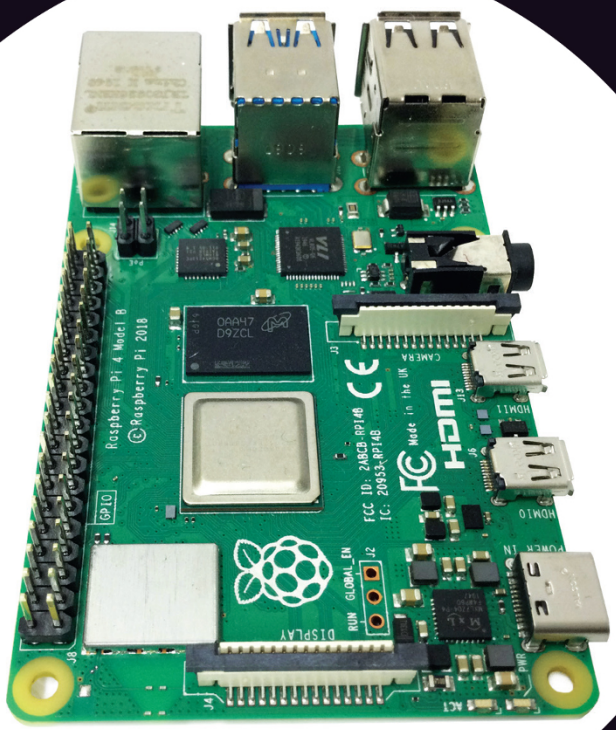
LEARNING MADE EASY

4th Edition

Raspberry Pi®

for dummies®
A Wiley Brand



Connect the Raspberry Pi and configure the OS

Create games, explore electronics, make music, and more

Start programming with Scratch™ and Python®

Sean McManus
Mike Cook

Raspberry Pi®

for dummies®

A Wiley Brand

Raspberry Pi®

4th Edition

by Sean McManus and Mike Cook

for dummies®
A Wiley Brand

Raspberry Pi® For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

Library of Congress Control Number: 2021941416

ISBN 978-1-119-79682-4 (pbk); ISBN 978-1-119-79686-2 (ebk); ISBN 978-1-119-79687-9 (ebk)

Contents at a Glance

Table of Contents

Introduction

Raspberry Pi computers are at the forefront of the maker movement, where people make their own inventions using a mixture of traditional craft skills and modern coding and electronics knowledge. They've also given more and more people access to a computer that provides a gateway into programming, electronics, and the world of Linux — the technically powerful (and free) rival to Windows and Mac OS. As a supercheap computer, the Raspberry Pi is also being pressed into service in media centers and as a family computer for games, music, photo editing, and word processing.

You might be a geek who relishes learning new technologies, or you might be someone who wants a new family computer to use with the children. In either case, *Raspberry Pi For Dummies*, 4th Edition, helps you get started with your Raspberry Pi and teaches you about some of the many fun and inspiring things you can do with it.

About This Book

Raspberry Pi For Dummies, 4th Edition, provides a concise and clear introduction to the terminology, technology, and techniques that you need to get the most from your Pi. With this book as your guide, you'll learn how to

- Set up your Raspberry Pi.
- Discover and install great free software you can use on your Raspberry Pi.
- Use the desktop environment to run programs, manage files, surf the web, and view photos.
- Use the Linux command line to manage your Raspberry Pi and its files.
- Use the Raspberry Pi as a productivity tool.
- Edit photos.
- Play music and video.
- Create animations and arcade games with the child-friendly Scratch programming language.

- Write your own games and other programs using the Python programming language.
- Compose music by programming with Sonic Pi.
- Get started with electronics, from an introduction to soldering to the design and creation of electronic projects controlled by the Raspberry Pi.

Incidentally, within this book, you may note that some web addresses break across two lines of text. If you're reading this book in print and want to visit one of these web pages, simply key in the web address exactly as it's noted in the text, pretending as though the line break doesn't exist. If you're reading this as an ebook, you've got it easy — just click or tap the web address to be taken directly to the web page.

Foolish Assumptions

Raspberry Pi For Dummies, 4th Edition, is written for beginners, by which we mean people who have never used a similar computer. However, we do have to make a few assumptions in writing this book, because we wouldn't have enough space for all its cool projects if we had to start by explaining what a mouse is! Here are our assumptions:

- **You are familiar with other computers, such as Windows or Apple computers.** In particular, we assume that you're familiar with using windows, icons, and the keyboard and mouse, and that you know the basics of using your computer for things like browsing the Internet, installing software, or copying files.
- **The Raspberry Pi is not your only computer.** At times, you'll need to have access to another computer — for example, to create your SD or microSD card for the Pi. (See Chapter 2.) When it comes to networking, we assume you already have a router set up with an Internet connection and a spare port that you can plug the Raspberry Pi into if you're using a wired connection.
- **The Raspberry Pi is your first Linux-based computer.** If you're a Linux ninja, this book still gives you a solid reference on the Raspberry Pi and the version of Linux it uses, but no prior Linux knowledge is required.
- **You share our excitement.** The Raspberry Pi can open up a world of possibilities to you!

Other than these assumptions, we hope this book is approachable for everyone. The Raspberry Pi is being adopted in classrooms and youth groups, and this book

is a useful resource for teachers and students. The Raspberry Pi is also finding its way into many homes, where people of all ages (from children to adults) are using it for education and entertainment.

Icons Used in This Book

If you've read other *For Dummies* books, you know that they use icons in the margin to call attention to particularly important or useful ideas in the text. In this book, we use four such icons:

The Tip icon highlights expert shortcuts or simple ideas that can make life easier for you.

Although we'd like to think that reading this book is an unforgettable experience, we've highlighted some points that you might want to particularly commit to memory. They're either important takeaways, or they're fundamental to the project you're working on.

As you would do on the road, slow down when you see a Warning icon. It highlights an area where things could go wrong.

Arguably, the whole book talks about technical stuff, but this icon highlights something that's *particularly* technical. We've tried to avoid unnecessary jargon and complexity, but some background information can give you a better understanding of what you're doing, and sometimes we do need to get quite techy, given the sophistication of the projects you're doing. Paragraphs highlighted with this icon might be worth rereading, to make sure you understand, or you might decide that you don't need to know that much detail. It's up to you!

Beyond the Book

In addition to what you're reading right now, this book comes with a free access-anywhere Cheat Sheet with tips on installing software and using Scratch. To get this Cheat Sheet, simply go to `www.dummies.com` and type **Raspberry Pi Dummies Cheat Sheet** in the Search box.

Also be sure to check out this book's companion website (www.dummies.com/go/raspberrypifd4e), where you can download the code listings that appear throughout this book.

Both of us maintain our own personal websites too, which contain some additional information on the Raspberry Pi. Mike's is at www.thebox.myzen.co.uk/Raspberry/Punnet.html, and Sean's is at www.sean.co.uk.

Where to Go from Here

It's up to you how you read this book. It's been organized to take you on a journey from acquiring and setting up your Raspberry Pi to learning the software that comes with it, and from writing your own programs to finally creating your own electronics projects. Some chapters build on knowledge gained in earlier chapters, especially the sections on Scratch and Python — and all of Part 5.

We understand, though, that some projects or topics might interest you more than others, and you might need help in some areas right now. When a chapter assumes knowledge from elsewhere, we include cross-references to help you quickly find what you might have missed. We also include some signposts to future chapters, so you can skip ahead to a later chapter if it provides the quickest answer for you.

If you haven't set up your Pi yet, start with Part 1. If you have your Pi up and running, Part 2 shows you how to use the software on it. Part 3 covers productivity, creativity, and entertainment software. To flex your programming muscles, perhaps for the first time, read Part 4. You can learn Scratch, Python, or Sonic Pi here, and feel free to start with any one of those languages. The Python chapters provide a good foundation for Part 5, where you can start building your own electronics projects.

1 Setting Up Your Raspberry Pi

IN THIS PART . . .

Get to know the Raspberry Pi and what other equipment you will need to be able to use it.

Download the Linux operating system and prepare a microSD card for use on your Raspberry Pi.

Connect your Raspberry Pi to the power, keyboard, mouse, and screen.

Install and test the Raspberry Pi Camera Module.

Change the settings on your Raspberry Pi.

IN THIS CHAPTER

- Getting up close and personal with the Raspberry Pi
- Taking stock of your Raspberry Pi
- Purchasing your very own Raspberry Pi
- Figuring out what else you need

Chapter 1

Introducing the Raspberry Pi

The Raspberry Pi is perhaps the most inspiring computer available today. Although most of the computing devices being used (including phones, tablets, and game consoles) are designed to stop people from tinkering with them, the Raspberry Pi is exactly the opposite. It invites you to prod it, play with it, and create with it. It comes with the tools you need to start creating your own software (or *programming*), and you can connect your own electronic inventions to it. Some models are cheap enough that breaking them won't break the bank, so you can experiment with confidence.

Lots of people are fired up about the Raspberry Pi's potential, and they're discovering exciting new ways to use it. Dave Akerman (`www.daveakerman.com`) and friends attached one to a weather balloon and sent it nearly 40 kilometers high to take pictures of the Earth from near space using a webcam. (You can read about Dave's ballooning project in Chapter 20.)

Professor Simon Cox and his team at the University of Southampton connected 64 Raspberry Pi boards to build an experimental supercomputer, held together by Lego bricks. In the supercomputer (see Figure 1-1), the Raspberry Pis work together to solve a single problem. The project has been able to cut the cost of a supercomputer from millions of dollars to thousands or even hundreds of dollars, making supercomputing much more accessible to schools and students. Others

have also experimented with combining the processing power of multiple Pis. There's even an off-the-shelf kit you can use to combine four Raspberry Pi Zeros with a full-size Raspberry Pi (the Cluster HAT from Pimoroni) so that you can experiment with running programs across multiple Pis at the same time.

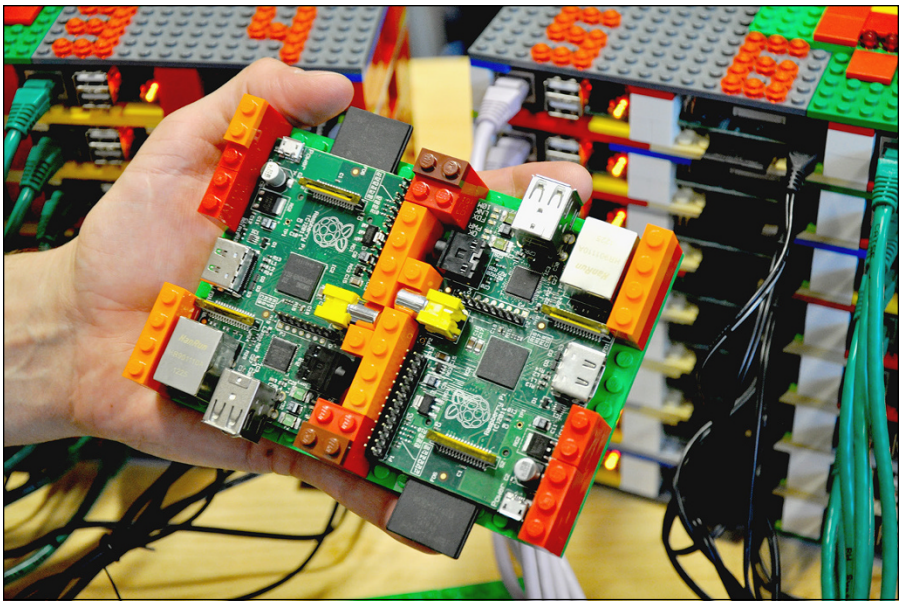


FIGURE 1-1: Two of the Raspberry Pi boards used in the University of Southampton's supercomputer, with the rest of the supercomputer in the background.

Courtesy of Simon Cox and Glenn Harris, University of Southampton.

The Pi is also being used to make fitness gadgets, gaming devices, electric skateboards, and much more, as you discover in Chapter 20.

Although those projects are grabbing headlines, another story is less visible but more important: the thousands of people of all ages who are taking their first steps in computer science, thanks to the Raspberry Pi.

Both of the authors of this book used computers in the 1980s, when the notion of a home computer first became a reality. Back then, computers were less friendly than they are today. When you switched them on, you were faced with a flashing cursor and had to type something in to get it to do anything. As a result, though, a whole generation grew up knowing at least a little bit about how to give the computer commands, and how to create programs for it. As computers started to use mice and windows, people didn't need those skills any more, and they lost touch with them.

Eben Upton, designer of the Raspberry Pi, noticed the slide in skill levels when he was working at Cambridge University's computer laboratory in 2006. Students

applying to study computer science started to have less experience with programming than students of the past did. Upton and his university colleagues hatched the idea of creating a computer that would come supplied with all the tools needed to program it — and would sell for a target price of $25 (about £20). It had to be able to do other interesting things, too, so that people were drawn to use it, and it had to be robust enough to survive being pushed in and out of school bags hundreds of times.

That idea started a six-year journey that led to the Raspberry Pi you probably have on your desk you as you read this book. It was released in February 2012, and sold half a million units by the end of the quarter. By July 2017, there were more than 14 million Raspberry Pis in homes, schools, and workplaces, 10 million of them made in the UK. More than 30 million Raspberry Pi computers have now been sold. It is, by a large margin, the best-selling British computer of all time.

Introducing the Raspberry Pi Range

Over the years, the Raspberry Pi has evolved, increasing its memory, improving its performance, and adding features. So which one should you get? Here's an overview designed to help you decide.

Raspberry Pi 4 Model B

This model is a circuit board with components and sockets stuck on it, as shown in Figure 1-2. In an age when most computing devices are sleek and shiny boxes, the spiky Pi, with tiny codes printed in white all over it, seems alien. That's a big part of its appeal, though: Many of the cases you can buy for the Raspberry Pi are transparent because people love the look of it.

The Raspberry Pi 4 is the latest Raspberry Pi board. It features the following:

- Up to 8GB of memory
- Four USB ports (two USB 2 ports and two higher-speed USB 3 ports)
- Built-in Wi-Fi and Bluetooth and a Gigabit Ethernet port for a wired Internet or network connection
- A headphones-style audio-out socket
- 40 general-purpose input/output (GPIO) pins, which you can use to connect your own electronics projects or specially designed add-ons (see Chapter 21)

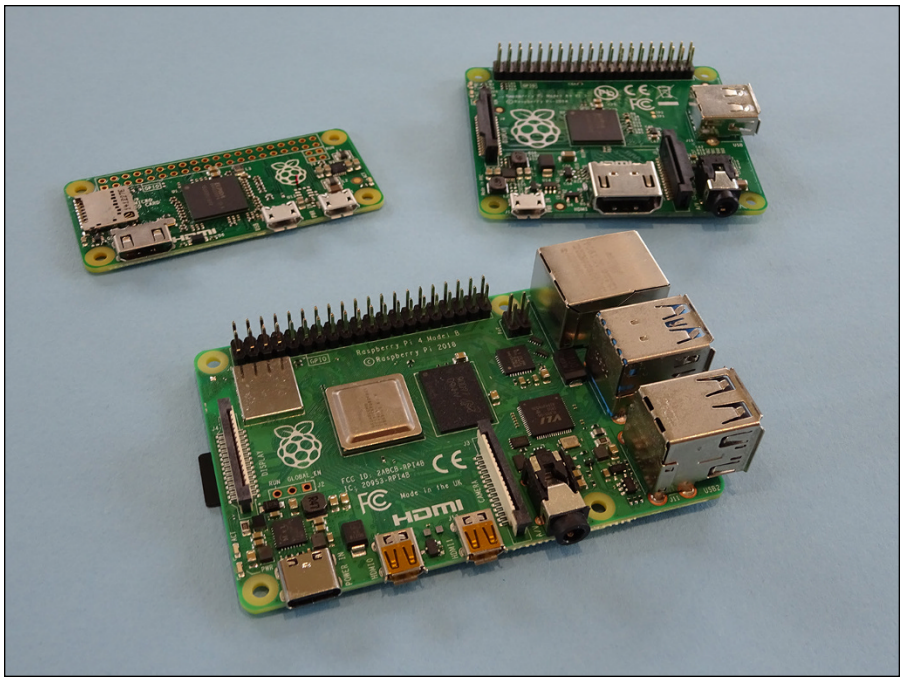


FIGURE 1-2: The Raspberry Pi 4 Model B (center), Model A+ (top right), and Pi Zero W (top left).

- Support for two monitors at resolutions of up to 4K
- Compatibility with the Raspberry Pi Camera Module
- Power over Ethernet (PoE) support when used with the Raspberry Pi PoE HAT, which enables you to use your Ethernet cable for both networking and powering your Pi

Like previous Pi models, the Raspberry Pi 4 is about the size of a deck of cards. As with any current Raspberry Pi, it uses a microSD card for storage. Its price is around $35 for 2GB of memory or $75 for 8GB of memory.

The Raspberry Pi Desktop Kit is also available, which includes the accessories you'll need, except for the monitor.

The Raspberry Pi 4 is our recommendation for the most powerful budget-friendly Raspberry Pi. You may be able to use it with your own keyboard and mouse to save money. The GPIO pins are great for electronics projects.

TECHNICAL STUFF

It's called the Model B, incidentally, as a tribute to the BBC Microcomputer that was popular in the UK in the 1980s. It's sobering to think that the BBC Micro cost about ten times the price of a Raspberry Pi, which, thanks to 40 years of progress in computer science, has more than 15,600 times more memory.

Raspberry Pi 400

The Raspberry Pi 400 (see Figure 1-3) takes even more inspiration from the classic computers of the '80s by building the Raspberry Pi 4 computer into a computer keyboard. It makes the whole setup much more compact, because you don't have the separate Pi unit on the table, with a cable going to the keyboard.

FIGURE 1-3: The Raspberry Pi 400 hides the computer inside the keyboard.

There are performance improvements, too. The Raspberry Pi 400 is faster than the Raspberry Pi 4, and it's designed with passive cooling built in.

The Raspberry Pi 400 is a white keyboard, with all the sockets on the back of it. It features the following:

- 4GB of memory.
- Three external USB ports (one USB 2 port and two higher-speed USB 3 ports). This is fewer than the four ports you get on a Raspberry Pi 4. The fourth port is used to connect the keyboard inside the case.
- Built-in Wi-Fi and Bluetooth and a Gigabit Ethernet port for a wired Internet or network connection.

- 40 GPIO pins, but these are on the back of the case, not on the top surface. You'll need to use an extension cable or board to use the pins easily and to use add-on boards (see Chapter 21). Although add-on boards can be connected directly, few will work well because their top surface will face away from you.
- Support for two monitors at resolutions of up to 4K.
- No compatibility with the Raspberry Pi Camera Module. You can use a USB camera, as you can on any Raspberry Pi computer.

There is no audio out socket, so you'll need to pass audio through your monitor.

The Raspberry Pi 400 costs $70. The Raspberry Pi 400 Personal Computer Kit adds the accessories you'll need, except for the monitor. The Raspberry Pi 400 is a fantastic value, but it's more expensive than the bare board. We recommend the Raspberry Pi 400 if your budget will bear it and you plan to use the Raspberry Pi as a desktop computer. For electronics projects, we find the bare board easier to use.


TIP

The official Raspberry Pi keyboard and the Raspberry Pi 400 look the same. If you have both on your desk, put a sticker on one of them; otherwise, you'll waste time trying to use the wrong one!

Raspberry Pi 3 Model A+

The Model A+ is a cut-down bare-board Raspberry Pi. It's useful for projects that need lower power consumption — typically battery-based projects. It is suitable for robots and projects in remote locations, where a wired electricity supply isn't viable and batteries must be used instead.

It features the following:

- 512MB of memory
- One USB 2 port
- Built-in Wi-Fi and Bluetooth
- A headphones-style audio-out socket
- 40 GPIO pins
- Compatibility with the Raspberry Pi Camera Module

This model has a price of $20. The Model A+ is slightly shorter on the long side than the Raspberry Pi 3, measuring 2½ inches by 2 inches.

Raspberry Pi Zero

The Raspberry Pi Foundation astounded everyone when it gave the Raspberry Pi Zero computer away with the print edition of its magazine *The MagPi.* We'd seen cover-mounted CDs and even tapes long ago, but never a computer before.

There are three models: Raspberry Pi Zero, Raspberry Pi Zero W (adding wireless networking), and Raspberry Pi Zero WH (adding wireless networking and GPIO pins).

The Raspberry Pi Zero family features the following:

- A lightweight, smaller board measuring just 2½ inches by 1 inch.
- A single-core 1 GHz processor. This is less powerful than the bigger boards. The Model B and A+ are quad-core, which means there are four processing units inside the chip that can all work at the same time. The quad-core processors run at a higher frequency, too. Here, you get a single core running at a lower frequency.
- 512MB of memory.
- One Micro USB port.
- Built-in Wi-Fi and Bluetooth, only on the Raspberry Pi Zero W and Zero WH.
- 40 GPIO pins, only on the Raspberry Pi Zero WH. On other models, you can solder your own pins.
- Compatibility with the Raspberry Pi Camera Module, only on the Raspberry Pi Zero W and Zero WH.

You'll also need a converter for the Mini HDMI socket, and for the Micro USB socket, so you should expect to spend a bit more than the price of the Pi (and have a bit more complexity in your setup). Billed as the $5 computer, the Raspberry Pi Zero has at times been difficult to get hold of, which is perhaps not surprising given the phenomenal demand for it.

The Raspberry Pi Zero is great for compact electronics projects that don't need the performance of a Model B or Model A+.

Older models

Of course, the older Raspberry Pis are still out there. Recent models usually remain in production while there is demand, and you can buy secondhand versions online from websites such as eBay. Generally speaking, the newer the model, the faster its performance. Memory upgrades have made a difference, as well as the use of more powerful processors, as the Pi has evolved. There are plenty of uses for the Pi that don't need especially fast performance, though, so you might find that an older Pi is perfect for your project. If you want to support the Raspberry Pi Foundation while buying cheaper, secondhand boards, you can donate to the foundation online.

The older models are described in this list:

- **Raspberry Pi 1 Model B with 256MB memory:** Although it's called Model B, this was the first Raspberry Pi to be released, in February 2012. The Raspberry Pi Model B features an Ethernet connection for the Internet and two USB ports. It uses an SD card for storage.
- **Raspberry Pi 1 Model B with 512MB memory:** Released in October 2012, the Raspberry Pi Model B had twice the memory capacity. This improved the speed of some software, especially applications that used images heavily.
- **Raspberry Pi 1 Model A:** The Model A, released in February 2013, is a stripped-down version of the Model B. It has just one USB port and doesn't have an Ethernet port for connecting to the Internet. It has 256MB of memory.
- **Raspberry Pi 1 Model B+:** The Model B+, released in July 2014, has been described by the Raspberry Pi Foundation as "the final evolution of the original Raspberry Pi." It runs all the same software as the previous versions of the Raspberry Pi, but it has four USB ports, more GPIO pins for connecting electronics projects to the Pi, and lower power consumption and better audio than the Model B. In common with the Model B, it has 512MB of memory. Although all previous versions use SD cards for data storage, the Model B+ introduced the smaller microSD cards, which are now standard on the Raspberry Pi.
- **Raspberry Pi 2 Model B:** Launched in February 2015, this model doubled the memory on the Model B+ to 1GB. It increased performance, compared to the Model B+, while retaining its physical features. Over the years the Pi's performance has been improved through new software releases as well as updates to the hardware. The Pi 2 represents an immediately noticeable speed-up, compared to the Model B+.

- **Raspberry Pi 3 Model B:** Launched in February 2016, this model has a new 64-bit processor, which means it can handle data in bigger chunks than the previous 32-bit processor. The Raspberry Pi 3 Model B is 50 percent to 60 percent faster than the Raspberry Pi 2 Model B when working in 32-bit mode.
- **Raspberry Pi 3 Model B+:** Launched in March 2018, this model has a faster processor and improved networking speeds. It introduced support for PoE, which enables the Raspberry Pi to be powered through the Ethernet cable. You'll need to add the Raspberry Pi PoE HAT accessory.

REMEMBER

If you're using anything earlier than the Model B+, you'll need full-size SD cards (not microSD) for storage, and you'll only have 26 GPIO pins to play with. Current add-ons are unlikely to be compatible with the early boards, so check their requirements before you buy.

Many of the projects in this book will work on older Raspberry Pi models (indeed, they first appeared in previous editions of this book when those models were the latest thing). But for best performance, we recommend using a current model, if possible.

WHAT'S THE RASPBERRY PI COMPUTE MODULE?

You'll also see the Raspberry Pi Compute Module in the online stores alongside the Raspberry Pi, but this is something quite different.

The Compute Module, or C for short, is designed for industrial use and intended to be built into a product you're manufacturing. The modules tend to follow the release of the main Raspberry Pi models. There are also light versions available that correspond to the Model A of the Raspberry Pi. They're built on a SODIMM board, which is what is sometimes used for PC memory modules. You're supposed to design your own board to plug the Compute Module into, but a development kit is available with a C module and an example motherboard containing all the normal plug-in connectors (see `www.raspberrypi.org/products/compute-module-development-kit-2`). Note, however, that this is an expensive way to buy what is otherwise a normal Raspberry Pi. Currently, the Raspberry Pi Compute Module 4 is the latest one, but the dev kit uses the Compute Module 3.

We only mention the Compute Module here in case you wonder what it is: It's not covered further in this book, and it's almost certainly not what you want to buy for your first Raspberry Pi.

RASPBERRY PI PICO: A MICROCONTROLLER, NOT A COMPUTER

The Raspberry Pi Pico is a radical new departure for the Raspberry Pi Foundation. Whereas previous devices were general-purpose computers, Raspberry Pi Pico is a microcontroller. A microcontroller is usually built into a device that does one job, such as a heating system or a microwave oven.

You can use the Raspberry Pi Pico for your electronics projects. You program it by connecting it to a computer. It's similar to the Arduino, which you might have heard of, but the Pico uses the Raspberry Pi Foundation's own custom chip.

The big advantage of a microcontroller is that there is no operating system to get in the way of things, so you can get precise control over the signals coming from its pins. This is important for things like audio generation and motor/servo control.

The Raspberry Pi Pico can be programmed using either MicroPython or C, which are both programming languages. (A programming language is a way of giving instructions to a computer or computing device – Part 4 introduces you to some programming languages). MicroPython is a version of Python optimized for running on microcontrollers. There are a few differences in some instructions, but MicroPython mostly looks the same as Python. You can program a Pico using Thonny, a Python programming tool available in Raspberry Pi OS. You get the option of saving your code into the Pico's memory or your computer. Any code saved into a file called `main.py` will run automatically when power is applied to the Pico, independently of whether you have a computer attached.

Programming a Pico in C, however, is not for the fainthearted. It requires a long process to prepare the C code for compiling or the use of a complex piece of software. We expect it to get easier, but at the moment we would recommend MicroPython instead.

The Raspberry Pi Pico is extremely cheap: It costs just $4, and it doesn't need an additional microSD card for storage.

You can find more information on the Raspberry Pi Pico in Chapter 17, but our focus in this book is on the Raspberry Pi computers and not the microcontroller. When we say "the Raspberry Pi," we're referring to the computers.

Figuring Out What You Can Do with a Raspberry Pi

The Raspberry Pi is a fully featured computer, and you can do almost anything with it that you can do with a desktop computer.

Instead of running Windows or macOS, the Raspberry Pi uses an operating system called Linux. It's a leading example of open source, a completely different philosophy to the commercial software industry. Rather than being created within the heavily guarded walls of a company, with its design treated as a trade secret, Linux is built by companies and expert volunteers working together. Anyone is free to inspect and modify the source code (a bit like the recipe) that makes it work. You don't have to pay to use Linux, and you're allowed to share it with other people, too.

You probably won't be able to run the software you have on your other computers on your Raspberry Pi. It won't run Windows or Mac software, and not all Linux software works on the Raspberry Pi. But a lot of Linux software that is compatible with the Raspberry Pi is available and is free of charge.

The Raspberry Pi has a graphical windows desktop to start and manage programs (see Chapter 4) as well as a shell for accepting text commands (see Chapter 5). You can use it for browsing the Internet (see Chapter 4), for word processing and spreadsheets (see Chapter 6), or for editing photos (see Chapter 7). You can use it for playing back music or video (see Chapter 8) or for playing games (see Chapter 19). You can use the built-in software to write your own music, too (see Chapter 14). It's the perfect tool for homework, but it's also a useful computer for writing letters, managing your accounts, and paying bills online.

The Raspberry Pi is at its best, however, when it's being used to learn how computers work, and how you can create your own programs or electronics projects using them. It comes with Scratch (see Chapter 9), a visual programming language that enables people of all ages to create their own animations and games while learning some of the core concepts of computer programming along the way.

It also comes with Python (see Chapter 11), a professional programming language used by YouTube, Google, and Industrial Light & Magic (the special effects gurus for the *Star Wars* films), among many others.

It has GPIO pins on it that you can use to connect up your own circuits to the Raspberry Pi, so you can use your Raspberry Pi to control other devices and to receive and interpret signals from them. In Part 5, we show you how to build some electronic projects controlled by the Raspberry Pi. In Chapter 21, we show you some add-ons you can connect to the GPIO pins.

Getting Your Hands on a Raspberry Pi

One of the great things about the Raspberry Pi is that it's established a community of businesses that have created products for it, or have shared in its success by selling it. You can now buy the Raspberry Pi from a wide range of electronics companies for hobbyists. Global retailers include Pimoroni (`www.pimoroni.com`), The Pi Hut (`https://thepihut.com`), and Adafruit (`www.adafruit.com`). It's also available from the Raspberry Pi's distributors, RS Components (`www.rs-components.com`) and Element14 (`www.element14.com`).

You might also be able to buy it from your local computer or electronics store, although you'll probably find it's only available as part of a kit there. Shops often bundle the Raspberry Pi with other items you need to use it. It can be convenient to get everything at once, but it might not represent the cheapest way to get started.

Determining What Else You Need

The creators of Raspberry Pi have stripped costs to the bone to enable you to own a fully featured computer for less than $35, so you'll need to scavenge or buy a few other bits and pieces in order to use your Pi. We say *scavenge* because the things you need are exactly the kind of things many people have lying around their house or garage already, or can easily pick up from friends or neighbors. In particular, if you're using a Raspberry Pi as your second computer, you probably have most of the peripherals you need.

WARNING

Not all devices are compatible. In particular, incompatible USB hubs, keyboards, and mice can cause problems that are hard to diagnose. USB hubs that feed power back into your Raspberry Pi through the Pi's USB port (known as *backpowering*) could potentially cause damage to the Raspberry Pi if they feed in too much power.

A list of compatible and incompatible devices is maintained at `https://elinux.org/RPi_VerifiedPeripherals`, and you can check online reviews to see whether others have experienced difficulties using a particular device with the Raspberry Pi.

TIP

If you're buying new devices, you can minimize the risk by buying recommended devices from Raspberry Pi retailers.

In any case, you should set a little bit of money aside to spend on accessories. The Raspberry Pi is inexpensive, but buying a keyboard, mouse, USB hub, and cables

can easily double or triple your costs, and you may have to resort to that if what you have on hand turns out not to be compatible.

The following sections offer a roundup of what else you may need.

Essentials

There are a few things that are essential to get your Raspberry Pi up and running:

- **Monitor:** The Raspberry Pi has a high-definition video feed and uses an HDMI (high-definition multimedia interface) or Micro HDMI connection for it. If your monitor has an HDMI socket, you can connect the Raspberry Pi directly to it. If your monitor does not support HDMI, it probably has a DVI socket, and you can get a simple and cheap converter that enables you to connect an HDMI cable to it. Older VGA (video graphics array) monitors require a device to convert the HDMI signal into a VGA one. If you're thinking of buying a converter, check online first to see whether it works with the Raspberry Pi. A lot of cheap cables are just cables, when what you need is a device that converts the signal from HDMI format to VGA, not one that just fits into the sockets on the screen and your Raspberry Pi. These converters can be quite expensive, so Gert van Loo has designed a device that uses the Raspberry Pi's GPIO pins to connect to a VGA monitor. He's published the design specs so that anyone can build one, and sell it if they want to, too. Take a look at eBay if you need one, and you might well find what you need. For more information, check out `https://github.com/fenlogic/vga666`. (If your monitor is connected using a blue plug and the connector has three rows of five pins in it, it's probably a VGA monitor.)
- **TV:** You can connect your Raspberry Pi to a high-definition TV using the HDMI socket and should experience a crisp picture. If you have an old television in the garage, you can also press it into service for your Raspberry Pi. The Pi can send a composite video signal, so it can use a TV as its display. When we tried this, it worked but the text lacked definition, which made it difficult to read. You'll need to get a cable with the right connector to fit your Pi: The original Model A and Model B have a dedicated RCA video socket, but current models use the headphone socket for RCA video output, too.
- **USB keyboard and mouse:** The Raspberry Pi only supports wired USB keyboards and mice. If you're still using ones with PS/2 connectors (round rather than flat), you may be able to use a PS/2 to USB adapter. Official Raspberry Pi keyboards and mice are available with an attractive white and red design. You can use Bluetooth devices, but you'll need to use a wired keyboard and mouse to set them up.

TIP

When the Raspberry Pi behaves unpredictably, it can be because the keyboard is drawing too much power, so avoid keyboards with too many flashing lights and features.

- **SD card or microSD card:** The Raspberry Pi doesn't have a hard drive built into it, so it uses a microSD card (current models) or SD card (older models, earlier than the Model B+) as its main storage. You probably have some SD cards that you use for your digital camera, although you might need to get a higher-capacity one. We recommend a 16GB card as a minimum for Raspberry Pi OS, but you can use a 4GB card if you use a media center operating system (OS) like LibreELEC (see Chapter 8 for a guide to LibreELEC). SD and microSD cards have different class numbers that indicate how fast you can copy information to and from them. You will be fine with a Class 6 or higher. If you buy an official Raspberry Pi kit, it includes a microSD card with Raspberry Pi OS already installed on it.

  Note: In this book, when we say microSD card, we also mean SD card if that's what you're using. If we're talking about something that's different for SD cards, we tell you.

- **SD or microSD card writer:** Many PCs today have a slot for SD or microSD cards, so you can easily copy photos from your camera to your computer. If yours doesn't, you might want to consider getting an SD or microSD card writer to connect to your computer. You can use it to copy software to an SD card for use with your Raspberry Pi, but you won't be able to use it to copy files from your Raspberry Pi to a Windows computer. You can also use the card writer to create a backup copy of your Raspberry Pi's files and software. (You can read about making back-ups in Chapter 4.)

- **Power supply:** To power your Raspberry Pi, you need to use a 5V power supply. The Raspberry Pi 4 and Raspberry Pi 400 use a USB-C connector, and earlier models use a USB-C Micro USB connector. Although you may have mobile phone and tablet chargers that fit, many of them can't deliver enough current (up to 2,500 milliamperes for a Raspberry Pi 3 Model A+, and up to 3,000 milliamperes for Raspberry Pi 4), which can make the Raspberry Pi perform unreliably. It's worth checking to see whether you have a 5V charger that may do the job (it should say on it how much current it provides), but for best results, we recommend buying a compatible charger from the same company that you buy your Raspberry Pi from. There is an official Raspberry Pi 4 power supply available, which has plug styles for the United States, Canada, United Kingdom, Australia, New Zealand, Europe, India, and China.

  Don't try to power the Pi by connecting its power port to the USB port on your PC with a cable, because your computer probably can't provide enough power for your Pi. You can also power the Pi through the GPIO pins, but you could damage the Raspberry Pi if there is a spike in current or the wrong voltage is applied. If you want to provide power through the GPIO pins, a

safer approach is to use a hardware-attached-on-top (HAT) device designed to sit on the GPIO pins and provide the consistent power you need while protecting the Pi underneath. For portable applications, you can power the Raspberry Pi using a battery pack designed for mobile phone charging. The Raspberry Pi Foundation advises that you should only use batteries to power your Raspberry Pi if you know what you're doing, because there's a risk of damaging your Raspberry Pi. There is an official Raspberry Pi PoE HAT if you want to power your Pi through an Ethernet cable.

For more details on the power requirements of various Raspberry Pi models, consult the FAQ at `www.raspberrypi.org/documentation/faqs`.

- **Cables:** You'll need cables to connect it all up, too. In particular, you need an HDMI cable (if you're using an HDMI or DVI monitor), an HDMI-to-DVI adapter (if you're using a DVI monitor), an RCA cable (if you're connecting to an older TV), an audio cable (if you're connecting the audio jack to your stereo), and an Ethernet cable (for networking on models with an Ethernet port). The Raspberry Pi 4 and 400 use Micro HDMI connections, so you'll need a cable that connects Micro HDMI to (normal) HDMI for your monitor, or an adapter. Note that the Raspberry Pi 2 and later (including Raspberry Pi 4) send the RCA video signal through a 3.5mm jack (headphone socket). Earlier models had a dedicated RCA socket. You need a different cable, depending on which version of the Pi's design you have, if you plan to use RCA. If you have a Raspberry Pi Zero, you'll need a converter for the Mini HDMI socket and for the Micro USB socket (see Figure 1-4). You can get these cables from an electrical components retailer, and you may be able to buy them at the same time as you buy your Raspberry Pi. Any other cables you need (for example, to connect to PC speakers or a USB hub) should come with those devices.

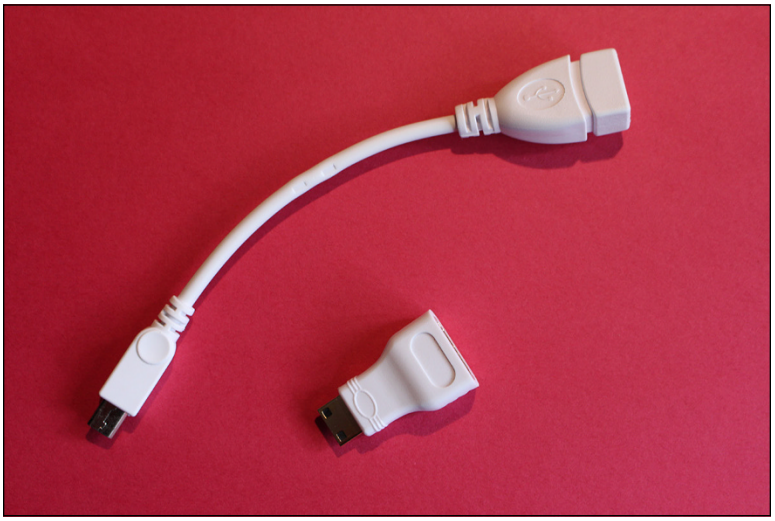


FIGURE 1-4: The Micro USB–to–USB converter cable and the Mini HDMI–to–HDMI converter for the Raspberry Pi Zero.

Optional extras

There are a few additional items you may want to get for your Raspberry Pi. They can make your Raspberry Pi easier to use and enable new applications.

- **USB hub:** The Raspberry Pi has one, two, or four USB sockets (depending on the model you get). Consider using a powered USB hub, for two reasons. Firstly (and especially if you have a Model A, A+, B, or Zero), you're going to want to connect other devices to your Pi at the same time as your keyboard and mouse, which need two sockets. And secondly, a USB hub provides external power to your devices and minimizes the likelihood of experiencing problems using your Raspberry Pi, especially if connecting relatively power-intensive devices such as hard drives. Make sure your USB hub has its own power source, independent of the Raspberry Pi.
- **External hard drive:** If you want lots of storage, perhaps so that you can use your music or video collection with the Raspberry Pi, you can connect an external hard drive to it over USB. You'll need to connect your hard drive through a powered USB hub, or use a hard drive that has its own external power source.
- **Raspberry Pi Camera:** The Raspberry Pi has stimulated entrepreneurs to create all kinds of add-ons for it, but the Camera Module is a product that originated at the Raspberry Pi Foundation. This fixed-focus camera can be used to shoot HD video and take still photos. The standard camera has 8-megapixel resolution, and the Raspberry Pi High Quality Camera offers 12-megapixel resolution. There is also a version of the standard camera without an infrared filter (the PiNoIR Camera), which can be used for wildlife photography at night or weird special effects by day.
- **Speakers:** Raspberry Pis (excluding the Pi 400) have a standard audio out socket, compatible with headphones and PC speakers that use a 3.5mm audio jack. You can plug headphones directly into it, or use the audio jack to connect to speakers, a stereo, or a TV. If you're using a TV or stereo for sound, you can get a cable that connects the 3.5mm audio jack and the audio input(s) on your television or stereo. You won't always need speakers: If you're using an HDMI connection, the audio is sent to the screen with the video signal, so you won't need separate speakers. If you're using a DVI monitor, you can get an HDMI-to-DVI adapter that includes audio extraction, so you can connect the audio separately. Some adapters can also convert from HDMI to VGA, with sound extracted separately.
- **Case:** It's safe to operate your Raspberry Pi as is, but many people prefer to protect it from spills and precariously stacked desk clutter by getting a case for it. The Pibow Coupe (`https://shop.pimoroni.com/collections/pibow`) is one of the most attractively designed cases, assembled from layers

of colored plastic (see Figure 1-5). It's designed by Paul Beech, who designed the Raspberry Pi logo. There are also official red-and-white cases for current Raspberry Pi models. The case for the Pi Zero includes three different tops, so you can either seal it, leave a camera hole, or have access to the GPIO pins. You don't have to buy a case, though. You can go without or make your own using cardboard or Lego bricks. Whatever case you go with, make sure you can still access the GPIO pins so that you can experiment with connecting your Pi to electronic circuits and try the projects in Part 5 of this book.

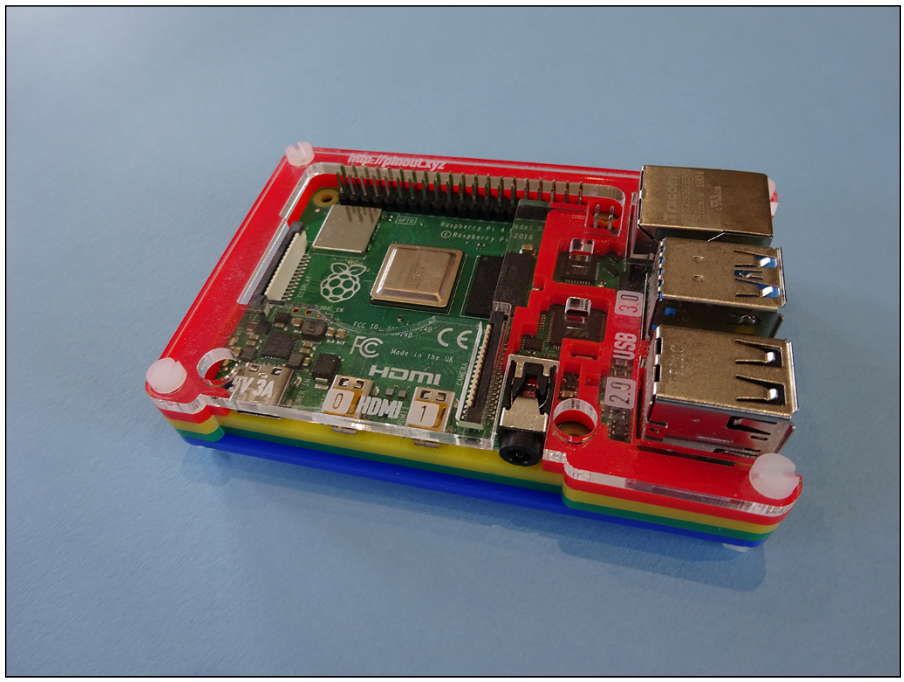


FIGURE 1-5: The Pibow Coupe case on the Raspberry Pi 4.

» **Raspberry Pi 4 Case Fan:** If you're really pushing the performance of your Raspberry Pi 4, you might find it gets a bit hot. The Raspberry Pi 4 Case Fan (see Figure 1-6) is an official accessory that fits inside the official Raspberry Pi case. It connects to your GPIO pins, and the fan spins to keep air flowing through the case. It's useful for power users, but most people won't need one.

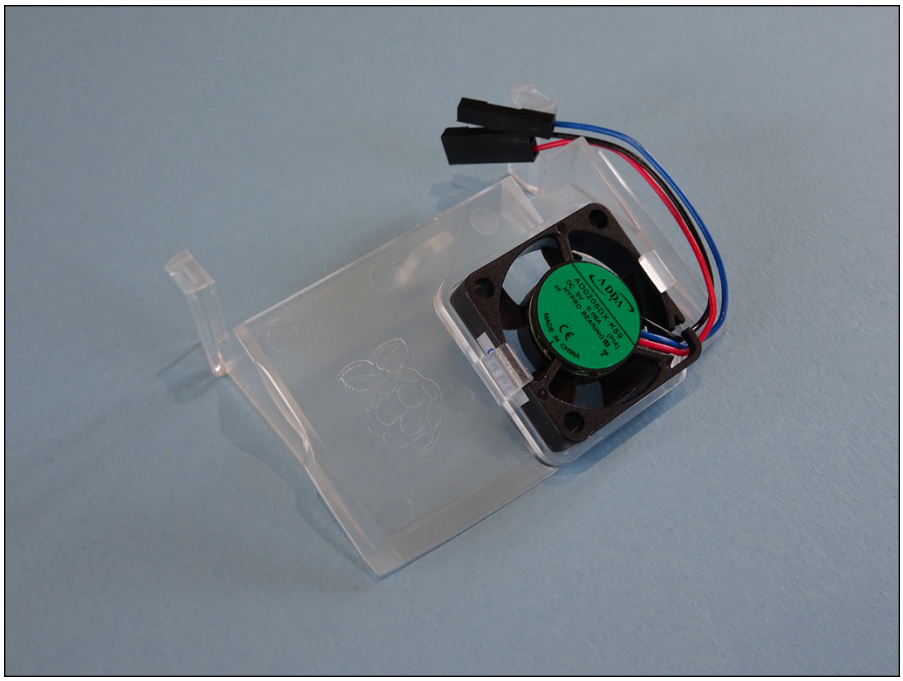


FIGURE 1-6: The Raspberry Pi 4 Case Fan.

4 Programming the Raspberry Pi

IN THIS PART . . .

Get familiar with the Scratch interface and how you can use it to create your own simple animations or games programs.

Use Scratch to build an arcade game, which you can customize with your own artwork.

Learn how to use Python to create a times table calculator and Chatbot, a program that simulates basic artificial intelligence.

Use Pygame Zero with Python to create a simple arcade game that you can customize with your own sounds and artwork.

Discover how you can use Python to build worlds in Minecraft, including a maze you can explore.

Use Sonic Pi to compose your own computer music on your Raspberry Pi.

IN THIS CHAPTER

- Starting Scratch
- Understanding the Scratch screen layout
- Making your sprite move
- Creating scripts
- Changing your sprite's appearance
- Adding sounds
- Adding extensions

Chapter 9
Introducing Programming with Scratch

The Raspberry Pi was created partly to inspire the next generation of programmers, and Scratch is the perfect place to start. With it, you can make your own cartoons and games and discover some of the concepts that professional programmers use every day.

Scratch is designed to be approachable for people of all ages. The visual interface makes it easy to see what you can do at any time, without having to remember any strange codes, and you can rapidly achieve great results. Scratch comes with a library of images and sounds, so it takes only a few minutes to write your first Scratch program.

In this chapter, we introduce you to Scratch so that you can start to experiment with it. In Chapter 10, we show you how to use Scratch to make a simple arcade game.

Understanding What Programming Is

Before we dip into Scratch, we should clear up some of the jargon surrounding it. A *program* is a repeatable set of instructions to make a computer do something, such as play a game. Those instructions can be extremely complicated because they have to describe what the computer should do in detail. Even a simple bouncing-ball game requires instructions for drawing the ball, moving it in different directions, detecting when it hits something, and then changing its direction to make it bounce.

Programming is the art and science of creating programs. You can create programs in lots of different ways, and Scratch is just one of them. In Chapter 11, you read about Python, another one.

Scratch and Python are both *programming languages*, different ways of writing instructions for the computer. Different programming languages are best suited for different tasks. Scratch is ideal for making games, for example, but it's not much use if you want to create a word processor. Using Python to create games takes longer, but it is more powerful than Scratch and gives you much more flexibility in the type of things you can get the computer to do.

Working with Scratch

There are two versions of Scratch in the recommended programs for Raspberry Pi OS:

- **Scratch:** This is the original version of Scratch, widely known as Scratch 1.4. This version was optimized to perform well on the early Raspberry Pi models. If you're using an old Raspberry Pi model, you may prefer to use this version because it's faster, although it doesn't have all the features of the latest Scratch version. The screen layout is similar to Scratch 3, except that the Code Area is called the Scripts Area, and the tabs for Scripts, Costumes, and Sounds are above the Scripts Area. Additionally, the buttons to select different parts of the Blocks Palette are above it, instead of to its left.

» **Scratch 3:** If you use the online version of Scratch (at `https://scratch.mit.edu`), Scratch 3 is the version you're familiar with. New features in this latest version of the language include Extensions. These enable you to add new capabilities to Scratch, including for text to speech, and for electronics on the Raspberry Pi. We recommend you use Scratch 3, so you can benefit from the latest improvements to Scratch.

In this chapter, we assume you're using Scratch 3.

To start Scratch, select your chosen version from the Applications menu in the upper left of the screen. You can find all three versions of Scratch in the `Programming` folder. If they aren't installed on your computer, you can add Scratch using the Recommended Software application in the Applications menu. It's in the `Preferences` folder.

Understanding the Scratch screen layout

Scratch divides the screen into four main areas, as you can see in Figure 9-1. The Stage is where you can see your game or animation take shape. There's a cat on it already, so you can get started straightaway by making it do things, as you see in a minute. The Stage is in the upper right.

Underneath the Stage is your Sprite List. You can think of sprites as the characters in your game. They're images that you can make do things, such as move around or change their appearance. For now, there's just the cat, which has the name Sprite1.

You create a Scratch program by snapping together *blocks*, which are short instructions. You find the blocks in the Blocks Palette, which is on the left. It displays the Motion blocks by default. They include instructions to move ten steps, rotate, go to a particular grid reference, and point in a particular direction.

The Code Area is where the magic happens! You assemble your program in this space by dragging blocks into it from the Blocks Palette. The Code Area is between the Blocks Palette and the Stage.

There are two buttons in the upper right of the screen (refer to Figure 9-1) to toggle the Stage between full size and small. When the Stage is small, the Code Area is bigger, so you may find that useful when you're writing scripts later in this chapter. You can also make the Stage fill the screen when you're running your program.

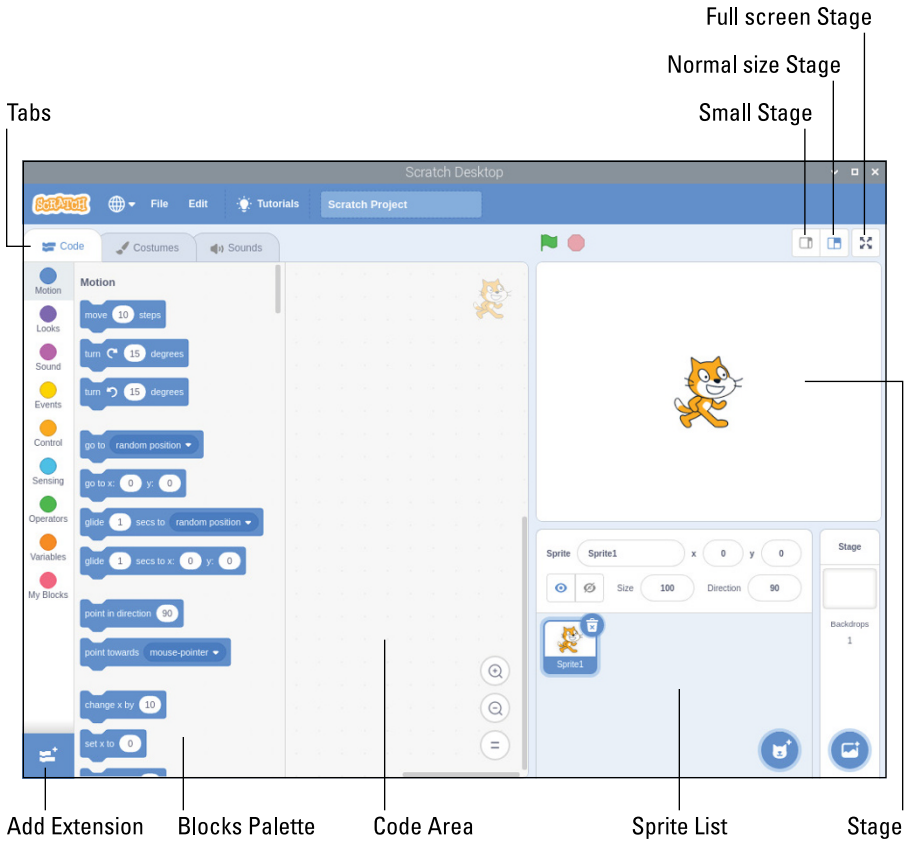


FIGURE 9-1: The screen layout in Scratch.

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See `https://scratch.mit.edu`.

Making your sprite move

You can drag and drop your sprite (the cat) around the Stage to position it where you would like it to be at the start of your program.

Experimenting with Scratch is easy. To try out different blocks, just click them in the Blocks Palette. For example, try clicking the Move 10 Steps block, and you should see the cat move in the direction it is facing, which is to the right. You can also turn the sprite 15 degrees in either direction by clicking the appropriate blocks.

TIP

If your cat goes somewhere that you don't want it to (don't they always?), you can click it on the Stage and drag it back to where you want it. In the Sprite List, you can also edit the sprite's *x* and *y* position, direction, and visibility (using the eye icons), if you're using Scratch 3. Set *x* to 0, *y* to 0, and direction to 90 to reposition your sprite in the middle of the screen, facing right.

Not all of the blocks will work at the moment because some of them need to be combined with other blocks. There's no harm in experimenting, however. Even if you click something that doesn't work, you won't cause any harm to Scratch or your Raspberry Pi.

Next, we talk you through the different Motion blocks you can use.

Using directions to move your sprite

You can use two different methods to position and move sprites. The first is to make your sprite "walk," and to change its direction when you want it to walk the other way.

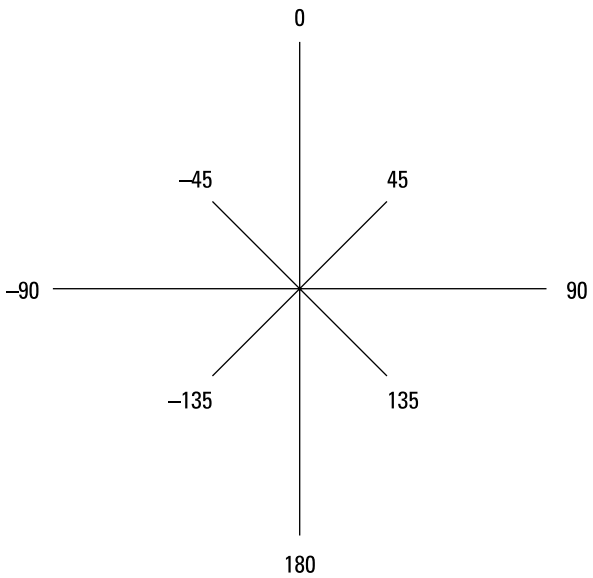
Here are the five blocks you use to move your sprite in this way:

- **Move 10 Steps:** This makes your sprite walk in the direction it is facing. If your sprite has been rotated, the steps taken could move it in a diagonal line across the Stage. You can click the number in this block and then type another number to increase or decrease the number of steps taken, but much bigger numbers spoil the illusion of animation. It stops looking like the sprite is walking across the screen when the number of steps taken is too big.
- **Turn Clockwise or Counterclockwise 15 Degrees:** These two blocks rotate your sprite. As with the number of steps, you can edit the number to change the degree by which your sprite is rotated. It walks in the direction it is facing when you use the Move 10 Steps block.
- **Point in Direction 90:** Whatever direction your sprite is facing, this block points it in the direction you want it to face. Use this block as is to reset your sprite to face right. You can change the number in this block to change the direction you want your sprite to face, and the numbers are measured in degrees from the position of facing up (see Figure 9-2). It helps to think of it like the hands of a clock: When the hand is pointing right, it's 90 degrees from the 12 o'clock position; when it's pointing down, it's 180 degrees from the top. To point left, you use –90. When you click the number box to type into it, a dial appears that you can use to select the angle.


TIP

You might be wondering whether you can use 270 to point left instead of –90. Try it. You'll see Scratch changes 270 to –90 if you type it into the block. It is possible to force the block to accept a value of 270 by using a variable. (You learn about variables in Chapter 10.) This can cause errors in your program, though. If you turn your cat to direction 270 and then ask Scratch which way the cat is facing, it tells you –90. To avoid any inconsistencies like this, keep direction numbers in the range from –179 to 180.

- **Point Towards:** You can also tell the sprite to point toward the mouse pointer or another sprite. Use the menu in this block to choose what you would like your sprite to point toward.
- **Set Rotation Style:** When the cat is facing left, it appears to stand on its head. You can change the rotation style to fix this. The style left-right keeps your sprite upright. The style all around will restore it to facing the direction it's moving in. There is also an option in this block to not rotate the sprite at all, even if its direction changes.
- **If On Edge, Bounce:** This block reverses the direction of your sprite if it's touching the edge of the Stage. It's useful for games or animations where you want sprites to bounce off the edges of the Stage.



Sean Mcmanus

FIGURE 9-2: The number of degrees used to face in different directions.

REMEMBER

If you're changing the number value in a block, you still need to click the block to run it.

Using grid coordinates to move and position your sprite

The second way you can move and position your sprite is to use grid coordinates. That makes it easy to position your sprite at an exact place on the screen, irrespective of where it is now.

Every point on the Stage has two coordinates: an X position (indicating where it is horizontally) and a Y position (indicating where it is vertically). The X positions are numbered from –240 at the far left to 240 at the far right. The Y positions are numbered from –180 at the bottom edge of the Stage to 180 at the top edge. That means the Stage is a total of 480 units wide and 360 units tall. The center point of the screen, where your cat begins its day, is where X equals 0 and Y equals 0. Figure 9-3 provides a quick visual reference of how the coordinates work.

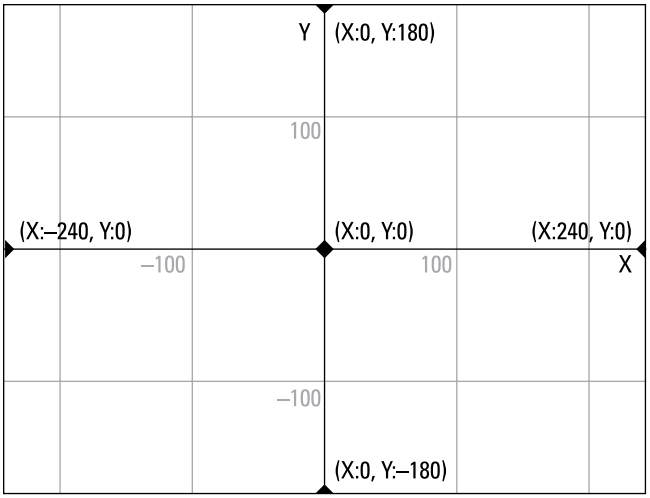


FIGURE 9-3: The grid coordinates on the Stage.

Sean McManus

Seven Motion blocks use the X and Y coordinates:

- **Go to x:0 y:0:** You can use this block to position your sprite at a specific point on the Stage.
- **Go to:** Use this block to move your sprite to a random position, the mouse pointer's location, or to the location of another sprite, if you have more than one.
- **Glide 1 secs to:** When you use the Go To block, your sprite just jumps to its new position. The Glide block makes your sprite float there smoothly instead. Like the Go To block, you can use this block to glide your sprite to a random position, the mouse pointer, or another sprite. You can change the number of seconds the glide takes, including using decimals for part of a second (for example, 0.5 for half a second).
- **Glide 1 secs to x:0 y:0:** Use this block to smoothly move your sprite to a new coordinate on the screen.

- **Change X by 10:** This moves your sprite ten units to the right. You can change the number of units and use a negative number if you want to move left instead. Note that this doesn't affect your sprite's vertical position and is independent of which way around your sprite is facing.
- **Set X to 0:** This changes the horizontal position of your sprite on the Stage, without affecting its vertical position. The value `0` returns it to the center of the screen horizontally, and you can edit the number to position it left or right of that. Use a negative number for the left half of the screen and a positive number for the right half.
- **Change Y by 10:** This moves your sprite ten units up the Stage, without affecting its horizontal position, and irrespective of which direction it is facing. You can change the number of units and use a negative number to move the sprite down the screen instead.
- **Set Y to 0:** This changes the vertical position of your sprite on the Stage without affecting its horizontal position, and without regard to which way it faces. Use a positive value for the top half of the Stage and a negative value for the lower half.

REMEMBER

You need to run a block to actually see its effect on your sprite. Do this by clicking it.

Showing sprite information on the Stage

It can be hard to keep track of where your sprite is and in which direction it's facing, but you can show the values for its X position, Y position, and direction on the Stage. Select the check boxes at the bottom of the Motion part of the Blocks Palette to do this, as shown in Figure 9-4. They clutter the screen a bit, but they can be essential tools for testing when you're creating a game.

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See `https://scratch.mit.edu`.

FIGURE 9-4: The blocks used to show sprite information on the Stage.

You can also refer to the Sprite List, where you can see a sprite's coordinates, size, and direction in Scratch 3.

Creating scripts

Clicking blocks in the Blocks Palette is one way to issue commands to Scratch, but if that's all you're doing, you're not really programming. The fact is, if you have to click each block every time you want to run it, you're doing all the hard work of remembering the instructions, and the computer can work only as fast as you can click the blocks.

A *program* is a reusable set of instructions that can be carried out (or *run*) whenever you want. To start to create a program, you drag blocks from the Blocks Palette and drop them in the Code Area in the middle of the screen. Most blocks mentioned so far in this chapter have a notch on the top and a lug on the bottom, so they fit together like jigsaw pieces. You don't have to align them perfectly: Scratch snaps them together for you if they're close enough when you release the mouse button.

You put your blocks in the order you want Scratch to run them, starting at the top and working your way down. It's a bit like making a to-do list for the computer.

A group of blocks in the Code Area is called a *script,* and you can run the script by clicking anywhere on it. Its border is highlighted, and you'll see the cat move around the Stage as you've instructed it to.

You can have multiple different scripts in the Code Area, so you could have one to make the cat walk left and another to make it walk right, for example. When you add multiple sprites (see Chapter 10), each sprite has its own Code Area and scripts there to control it.


TIP

If you want to tidy up the Code Area, you can move a script by dragging its top block. If you drag a block that is lower down in the script, it's separated from the blocks above it and carries with it all the blocks below it. If you want to delete a block or set of blocks, drag it back to the Blocks Palette on the left.

Figure 9-5 shows a script Sean built using some of the Motion blocks. Try building it in your Code Area, by dragging in the blocks and joining them together. Remember to change the numbers in the blocks. When you've finished, click the script to run it. The script makes the cat go to the middle of the screen, walk around the Stage in a square shape, and then point toward the mouse pointer. As you learn about new blocks in the rest of this chapter, you can try adding them to this script, or build your own new script.

Changing your sprite's appearance

As well as moving your sprite around the screen, you can change what it looks like.

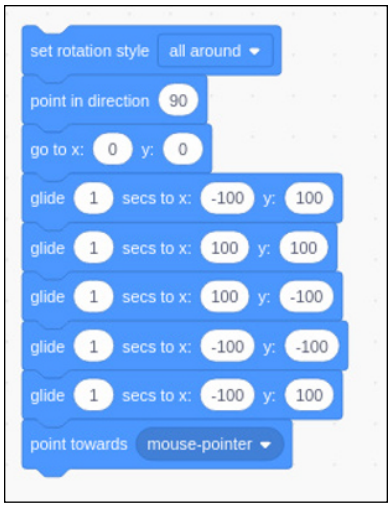


FIGURE 9-5: A simple script to make the cat walk around the Stage.

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

Using costumes

One way to think of sprites is as the characters in a game (although they can be used for lots of other objects, too, such as obstacles). Each sprite can have a number of *costumes*, which are different pictures of it. If the costumes look fairly similar, you can create the illusion of animation by switching between them. Your cat sprite comes with two costumes, and when you switch between them, it looks like the cat is running. You can think of a costume as being one image in an animation sequence (an *animation frame*).

You can see the costumes for your sprite by clicking the Costumes tab at the top of the Blocks Palette, as shown in Figure 9-6. If you want to modify the cat's appearance, you can click the costume on the left and use the editing canvas to its right. If you want to create a new animation frame, you can right-click the costume and choose duplicate from the menu that opens. You can then edit the bits you want to change.

TIP

It doesn't matter much when you're experimenting with sprites, but when you make your own games and animations, you can save yourself a lot of brain ache by giving your costumes meaningful names. It's much easier to remember that the costume with the name *game over* should be shown when the player is defeated than it is to remember it's called *costume7*. To rename a costume, click the Costumes tab to show the costumes, and then click the costume's current name (refer to Figure 9-6) and type its new name. The costume's name is shown above the editing canvas.

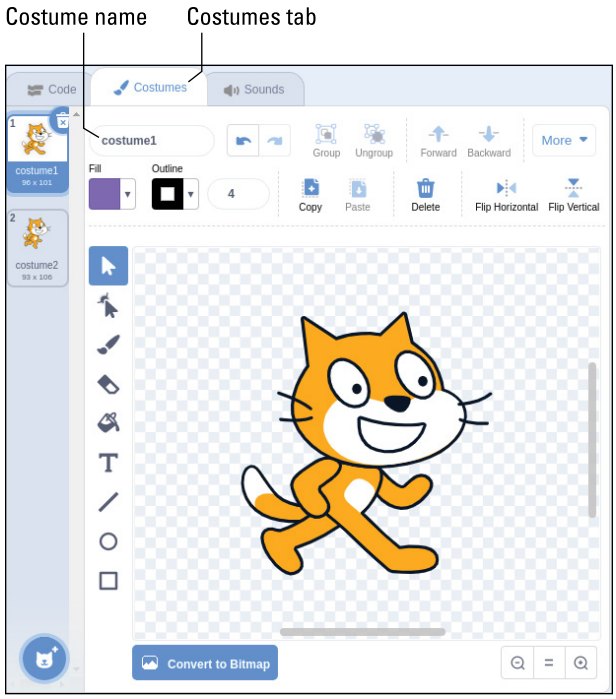


FIGURE 9-6: You can change a sprite's appearance by giving it a new costume.

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See `https://scratch.mit.edu`.

When you've finished using the Costumes Area, click the Code tab to get back to the Code Area.

In the Blocks Palette, there are two blocks you can use to switch between costumes (see Figure 9-7). Click the Looks button beside the Blocks Palette to show them:

- **Switch Costume to:** If you want to switch to a particular costume, choose its name from the menu in this block and then click the block.
- **Next Costume:** Each time you use this block, the sprite changes to its next costume. When the costumes run out, it goes back to the first one again.

TIP

You can show a sprite's costume number on the Stage, too, so that it's easier for you to work out what's going on. Just select the check box next to Costume # in the Blocks Palette. In that block, you can choose to show the costume's name instead.

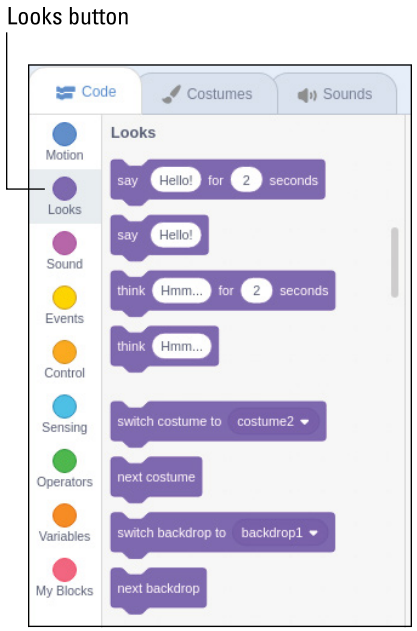


FIGURE 9-7: Some of the Looks blocks you can use to change your sprite's appearance.

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See `https://scratch.mit.edu`.

Using speech and thought bubbles

The Say blocks display a speech bubble, and the Think blocks show thought bubbles (see Figure 9-8). To see them, and to see the other blocks that change a sprite's appearance, click the Looks button beside the Blocks Palette. The speech and thought bubbles are great for giving a message to the player or viewer. You can edit the word in the block (*Hello!* or *Hmm . . .*) to change the text in the bubble. Figure 9-8 shows the speech bubbles (at the top) and thought bubbles (center and bottom) in action.

If you use one of the options with a length of time in it, the sprite pauses for that length of time and the bubble disappears when it has elapsed.

If you use a block without a length of time, you can make the bubble disappear again by using the Say or Think block again but editing the text so that the text box in the block is empty.

Using graphic effects

You can apply several graphic effects to your sprite using Looks blocks. In Figure 9-8, we've used eight sprites to demonstrate them on the Stage. The Color effect changes the sprite's Color Palette, turning orange to turquoise in the case of

the cat. The Fisheye effect works like a fish-eye lens, making the central parts of the sprite appear bigger. Whirl distorts the sprite by twisting its features around its middle. Pixelate makes the sprite blocky. Mosaic shrinks the sprite and repeats it within the space it usually occupies. The Brightness and Ghost effects can sometimes look similar, but the Brightness effect changes the intensity of the colors, and the Ghost effect fades out all colors evenly. In Figure 9-8, we've used a negative number with the Brightness effect to make the sprite darker.

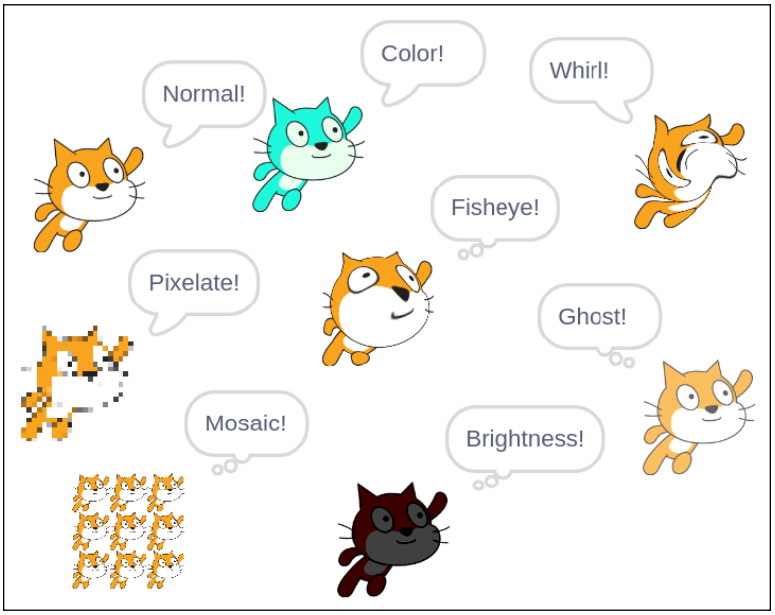


FIGURE 9-8: The different graphic effects you can apply to your sprite, with thought bubbles and speech bubbles used to describe them.

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

Here are the three blocks you use to control graphic effects:

- **Change Color Effect by 25:** You can select the effect you want to change (by default, it's the color effect) and enter the amount of it you want to add. You can use negative numbers to reduce the extent to which the effect is applied to your sprite. The color effect has 200 different levels (from 0 to 200), and the ghost effect has 100 different levels from 0 to 100. The other effects typically look best with levels in the range from –100 to 100. Experiment!
- **Set Color Effect to 0:** Use this block to set a chosen effect to a specific level. Choosing 0 turns the effect off again. You can use any of the seven effects with this block.
- **Clear Graphic Effects:** This block removes all graphic effects you've applied to a particular sprite so that it looks normal again.

Resizing your sprite

You can use blocks to change a sprite's size, so you could make it grow larger as the game progresses, for example.

There are two blocks you can use to resize your sprite:

- **Change Size by `10`:** This block enables you to change the size of your sprite by a certain number of units, relative to its current size. As usual, you can edit the number. If you want to decrease the sprite's size, use a negative number.
- **Set Size to `100%`:** This block sets the size to a percentage of its original size, so with the default value of 100 percent, it effectively resets any resizing you've done.


TIP

You can also select the check box beside the Size block to show the sprite's size on the Stage, in the same way you display other sprite information there. (See "Showing sprite information on the Stage," earlier in this chapter.) This can be useful for testing purposes.

Changing your sprite's visibility

Sometimes, you might not want your sprite to be seen on the Stage. If a spaceship is blown up in your game, for example, you want it to disappear from view. These two blocks give you control over whether a sprite is visible:

- **Hide:** Use this block to make your sprite invisible on the Stage. If a sprite is hidden, Scratch won't detect when it touches other sprites, but you can still move a hidden sprite's position on the Stage so that it's in a different place when you show it again.
- **Show:** By default, your sprite is visible, but you can use this block to reveal it again after you have hidden it.


TIP

Sometimes, sprites might get on top of each other. You can use the Go to Front Layer block to make a sprite appear on top of all the others, or use the menu in it to change it to Go to Back Layer, so you can force a sprite to go behind all the others. The Go Forward 1 Layers block enables you to move sprites forward and backward so you can precisely control which sprites are on top of which other sprites.

Adding sounds and music

As well as changing a sprite's appearance, you can give it some sound effects. Scratch comes with sounds, including slurps, sneezes, and screams; ducks, geese,

and owls; and pops, whoops, and zoops. You can find effects for most occasions, and many of them are natural partners for one of the sprites that Scratch provides.

To add a sound to your sprite, you have to do one task first: Import the sound to your sprite. Here's how you'd do that:

1. **Click the Sounds tab above the Blocks Palette, and then click the Choose a Sound button.**

   The button is in the lower left and it looks like a speaker.

2. **In the file browser that appears, browse the provided sounds.**

   There is a search box you can use if you know the name of the sound you're looking for, and there are category buttons that group the effects into loops, animal sounds, musical notes, and more.

3. **(Optional) Click the play button on a sound to hear it.**

4. **Click the sound to bring it into your sprite.**

   After you've imported a sound, you can preview it (see Figure 9-9). Click the play button in the Sounds Area. You can choose different sounds to preview using the panel on the left. Click the trashcan icon on a sound in this panel to delete it from your project.

   If you a delete a sound in this way, it remains on your SD card so that you can import it again later.

After a sound has been imported, you use one of the Sound blocks to play a sound. To see all available Sound blocks, click the Sound button beside the Blocks Palette first. Use the Code tab to return to the Code Area if you can't see any blocks.

The Play Sound block enables you to choose which sound you'd like to play from those you have imported. The Play Sound Until Done block stops the program from running any blocks joined underneath this one until the sound has finished playing.


REMEMBER

The sound is imported to a particular sprite, so if you can't see it as one of the choices in the Play Sound block, be sure you've imported it to the correct sprite. In Chapter 10, we cover how to use multiple sprites in a project.

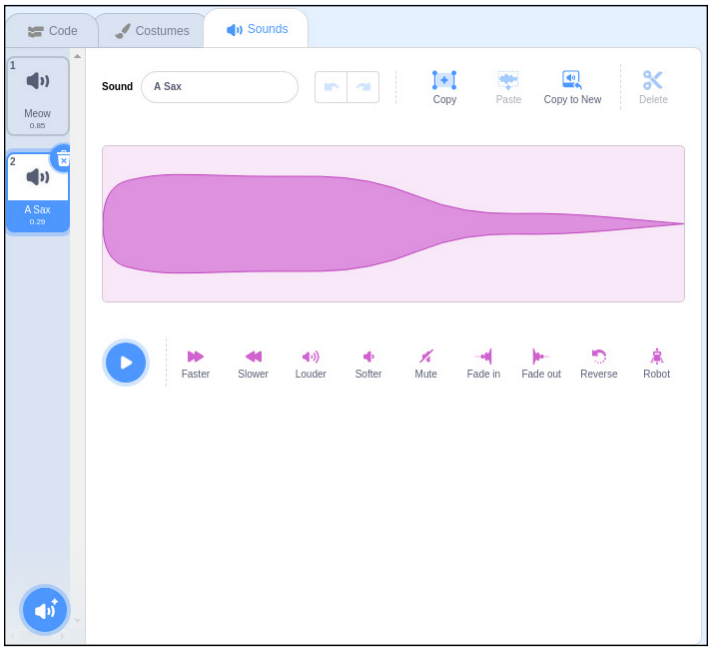


FIGURE 9-9: Adding sound effects to your sprite.

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See `https://scratch.mit.edu`.

Using the Wait block to slow down your sprite

As you put your script together, you might find that some movements happen so fast that you can hardly see what's going on.

If you click the Control button to the left of the Blocks Palette, you can find a set of blocks that are used to govern when particular things happen. You can read more about these in Chapter 10, but for now it's worth knowing that the Wait 1 Seconds block here enables you to wait for a certain number of seconds. Drag this into your script where necessary to introduce a delay so that you can see each of your blocks in action. The length of the delay is 1 second by default, but you can change it to whatever you want, including parts of a second (for example, 0.5 for half a second).

TIP

The Say Hello! for 2 Secs block can be also be used to force the script to pause before running any more blocks.

Using extensions in Scratch

The latest version of Scratch includes a feature to add new sections to the Blocks Palette, called Extensions. If you're familiar with earlier versions of Scratch, this is where you'll find the Music and Pen blocks that used to be part of the main Blocks Palette.

Adding extensions

The button to add extensions is at the lower left of the screen when you're in the Code Area (refer to Figure 9-1, earlier in this chapter). When you hover over it, it says Add Extension. It takes you to a screen where you can see the extensions available, and click to add one to your Blocks Palette.

We don't have space to cover all the extensions here, but here are some you may want to start experimenting with first.

Using the music extension

The music extension has blocks that let you use virtual drums and pitched instruments to create music using Scratch. Notes are numbered: C is 60, C# is 61, D is 62, and so on. There's a block called Play Note `60` for `0.5` Beats that plays a note with a particular number for a certain duration. When you click the menu in this block to specify which note to play, a piano opens that you can use to select the note.

There are also blocks you can use to change the sound of the instrument and set or change the tempo of the music.

TIP

If you're new to music, you can generally get a good result by starting with C, sticking to the white notes, and making sure no two consecutive notes are too far apart on the piano.

TIP

The note numbers used in Scratch are the same as those used in Sonic Pi (see Table 14-1, in Chapter 14).

Using the pen extension

The pen extension includes blocks for drawing on the Stage. As a sprite moves, it draws a line behind it in your chosen color. Try adding a Pen Down block at the top of the script in Figure 9-5 to see it in action. There are blocks to set the pen color and size (which determines the thickness of the line). You can use the Change Pen Color block to cycle through the colors in the pen's palette, by increasing the pen number by a certain number.

Using the electronics extensions

There are two electronics extensions for the Raspberry Pi. The Raspberry Pi Simple Electronics extension is ideal for getting light-emitting diodes (LEDs) and buttons working with your Raspberry Pi. The Raspberry Pi general-purpose input/output (GPIO) extension gives you more control over the default state of the circuit (pulled up or down). For most purposes, the simple electronics extension is all you need. For more on the electronics extensions, see Chapter 16.

Using the Sense HAT extension

The Sense HAT is an official add-on for the Raspberry Pi. It includes an 8 x 8 grid of LEDs, motion sensors, a joystick, and some environmental sensors. You can program it in Scratch using the Sense HAT extension.

If you don't have a Sense HAT, you can use Scratch together with the Sense HAT Emulator to test your Scratch project (see Figure 9-10).

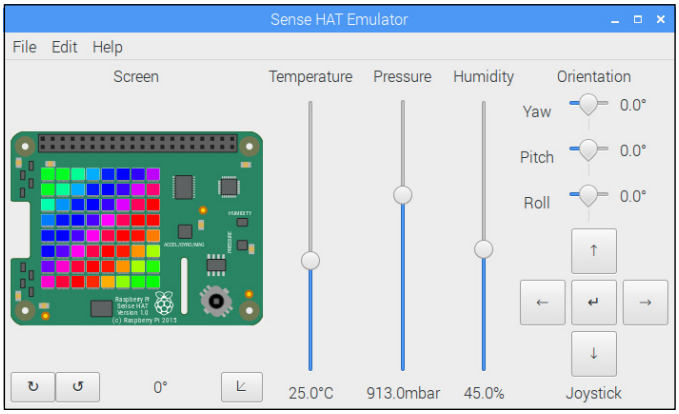


FIGURE 9-10: Emulating the Sense HAT.

The Sense HAT blocks include blocks to scroll text across the LEDs and to display a letter, a shape of your own design, a sprite, or the Stage. Given the extremely low resolution of the LED grid, the sprite and Stage are not always recognizable on the Sense HAT.

There are blocks to change the color used for text and shapes and the background color used for other LEDs. You can set the color of each LED individually, too.

There are blocks you can use to detect joystick movements and tilts and shakes of the device. You can also read the temperature, pressure, humidity, roll, pitch, and yaw sensors.

When you've made something you like, you can take it into the real world by buying a Sense HAT (which costs about $40).

Saving your work

Remember to save your work so that you can come back to it later. You can find the option to save on the File menu, at the top of the window. If you use Scratch online at the Scratch website instead of using the installed software on your Raspberry Pi, your work is saved automatically for you every few minutes. There is an option to Save Now in the upper right when you have unsaved changes.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

About the Authors

Sean McManus: Sean is an expert technology and business author. His other books include *Mission Python* (No Starch Press), *Coder Academy* (Kane Miller Books), *Cool Scratch Projects in Easy Steps* (In Easy Steps Limited), *Scratch Programming in Easy Steps* (In Easy Steps Limited), and *Web Design in Easy Steps* (In Easy Steps Limited). His novel for adults, *Earworm*, goes undercover in the music industry, exposing a conspiracy to replace bands with computer-generated music. His tutorials and articles have appeared in magazines including *The MagPi, Internet Magazine, Internet Works, Business 2.0, Making Music,* and *Personal Computer World.* He has been a Code Club volunteer, helping children at a local school to learn computer programming. Visit his website at `www.sean.co.uk` for bonus content from his books.

Mike Cook: Mike has been making electronic things since he was in school. A former senior lecturer in physics at Manchester Metropolitan University, he wrote more than 300 computing and electronics articles in the pages of computer magazines for 20 years starting in the 1980s. After leaving the university after 21 years when the physics department closed down, he got a series of proper jobs where he designed digital TV set-top boxes and access control systems. In 2015, he started writing a monthly column in *MagPi* magazine; he has covered 73 projects to date. His other books include *Raspberry Pi Projects* (Wiley), *Raspberry Pi Projects For Dummies* (Wiley), and *Arduino Music and Audio Projects* (Apress). He also works with Drake Music Labs North, a charity for disabled musicians, developing accessible music equipment. Now retired and freelancing, he spends his days surrounded by wires, patrolling the forums as Grumpy Mike.

Dedication

To my wife, Karen, with thanks for all her support throughout this project and always. And to Leo, our wonderful son.

—Sean

To my wife, Wendy, who always acts delighted whenever I show her yet another blinking LED. And also to the late Leicester Taylor, World War II radar researcher and inspirational supervisor of my post-graduate research at the University of Salford.

—Mike

Authors' Acknowledgments

Sean McManus: Thank you to my coauthor, Mike, for bringing his electronics expertise and fantastic project ideas. Thank you to Kelsey Baird, our acquisitions editor on this edition, and previous acquisitions editors Craig Smith and Katie Mohr. Thanks to Elizabeth Kuball, our editor on this edition, and previous editors Linda Morris, Paul Levesque, and Becky Whitney.

Our technical editors, Guy Hart-Davis (this edition), Jason E Geistweidt (3rd edition), Ryan Walmsley (2nd edition), and Paul Hallett (1st edition) cast a careful eye over the text and code and made much appreciated suggestions. Olivier Engler, who translated the first edition into French, provided helpful feedback, too. Thanks also to Lorna Mein and Natasha Lee in marketing, and to the *For Dummies* team for making it all happen.

Many people helped with research or permissions requests, including Karen McManus, Sam Aaron, Eben Upton, Liz Upton, Leo McHugh, Mark Turner, Peter Sayer, John Hartnup, Bill Kendrick, Simon Cox, Jon Williamson, Paul Beech, Peter de Rivaz, Michał Męciński, Ruairi Glynn, Stephen Revill, Lawrence James, Bram Stolk, Adam Kemeny, Will Jessop, David Bryan, Pimoroni, and Pi Supply. We wouldn't have a book to write if it weren't for the wonderful work of the Raspberry Pi Foundation, the manufacturers who took a gamble on it, and the many thousands of people who have contributed to the Raspberry Pi's software.

Mike Cook: I would like to thank Sean McManus for inviting me to contribute to this book and the staff at Wiley for making the process of producing this book as painless as possible.

Publisher's Acknowledgments

Acquisitions Editor: Kelsey Baird
Project Editor: Elizabeth Kuball
Copy Editor: Elizabeth Kuball
Technical Editor: Guy Hart-Davis
Production Editor: Vivek Lakshmikanth
Cover Image: Courtesy of Mike Cook; Raspberry Pi logo courtesy of Raspberry Pi Foundation

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.