Making Everything Easier!™

Raspberry Pi®

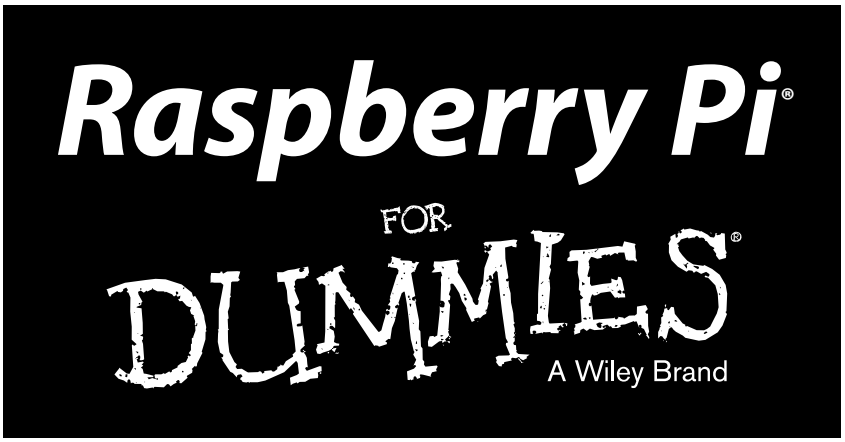
FOR DUMMIES®

A Wiley Brand

Learn to:

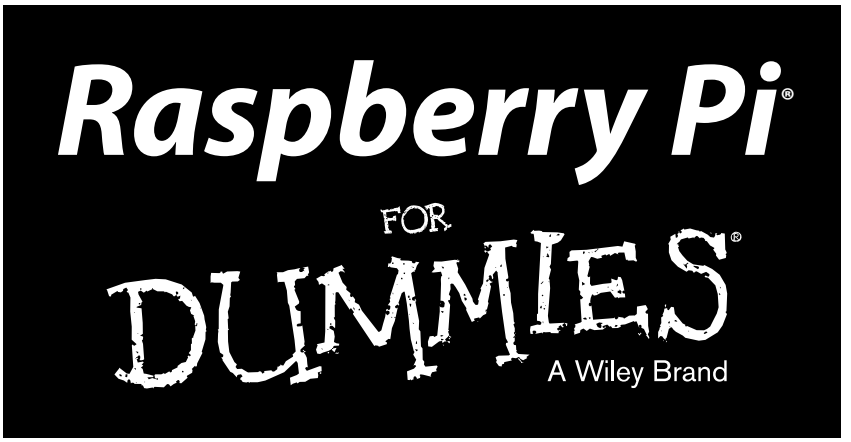
- Connect the Raspberry Pi and install the OS
- Get productive with word processing, spreadsheets, presentations, and images
- Learn simple programming with Scratch and Python
- Create electronics projects connected to the Raspberry Pi's GPIO port

Sean McManus
Mike Cook



01_9781118554210-ffirs.indd i 3/6/13 9:35 AM

01_9781118554210-ffirs.indd ii 3/6/13 9:35 AM



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

Copyright © 2013 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, the Wiley logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies Daily, The Fun and Easy Way, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Raspberry Pi and the Raspberry Pi logo are registered trademarks of the Raspberry Pi Foundation in the United Kingdom and other countries. *Raspberry Pi For Dummies* is not endorsed by the Raspberry Pi Foundation.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

ISBN 978-1-118-55421-0 (pbk); ISBN 978-1-118-55422-7 (ebk); ISBN 978-1-118-55423-4 (ebk);
ISBN 978-1-118-55424-1 (ebk)

Manufactured in the United States of America at Bind-Rite

10 9 8 7 6 5 4 3 2 1

About the Authors

Sean McManus is an expert technology and business author. His other books include *Microsoft Office for the Older and Wiser*, *Social Networking for the Older and Wiser*, *Web Design in Easy Steps*, and *iPad for the Older and Wiser*. His tutorials and articles have appeared in magazines including *Internet Magazine*, *Internet Works*, *Business 2.0*, *Making Music,* and *Personal Computer World*. His personal website is at `www.sean.co.uk`.

Mike Cook has been making electronic things since he was at school. Former Lecturer in Physics at Manchester Metropolitan University, he wrote more than three hundred computing and electronics articles in the pages of computer magazines for 20 years starting in the 1980s. 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. Now retired and freelancing, he spends his days surrounded by wires, patrolling the forums as Grumpy Mike.

01_9781118554210-ffirs.indd vi 3/6/13 9:35 AM

Dedication

Thank you to my wife, Karen, for all her support throughout this project. —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

Thank you to my co-author, Mike, for bringing his electronics expertise and fantastic project ideas. Thank you to Craig Smith for commissioning us to write this book, to Linda Morris for her editing support, and to Paul Hallett, our technical editor. 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, Leo McHugh, Mark Turner, Peter Sayer, Bill Kendrick, Simon Cox, Jon Williamson, Paul Beech, Peter de Rivaz, Michał Męciński, Ruairi Glynn, Stephen Revill, and Lawrence James.

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. —Sean

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. —Mike

Publisher's Acknowledgments

We're proud of this book; please send us your comments at http://dummies.custhelp.com. For other comments, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

Some of the people who helped bring this book to market include the following:

Acquisitions and Editorial

Project Editor: Linda Morris

Acquisitions Editor: Craig Smith

Copy Editor: Linda Morris

Technical Editor: Paul Hallett

Editorial Manager: Jodi Jensen

Editorial Assistant: Anne Sullivan

Sr. Editorial Assistant: Cherie Case

Cover Photo: © Dr. Andrew Robinson

Composition Services

Sr. Project Coordinator: Kristie Rees

Layout and Graphics: Carrie A. Cesavice, Jennifer Creasey, Joyce Haughey

Proofreader: Linda Seifert

Indexer: Potomac Indexing, LLC

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Kathleen Nebenhaus, Vice President and Executive Publisher

Composition Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

02_9781118554210-ftoc.indd x 3/6/13 9:36 AM

Table of Contents

Introduction

In recent years, computer education has focused largely on office skills, and not on understanding how computers work, or how you can use them to create new programs and inventions. The Raspberry Pi redresses the balance. It can be used for games, music, photo editing, and word processing, like any computer. But it can do so much more, providing a gateway into programming, electronics, and the mysterious world of Linux, the technically powerful (and free) rival to Windows and Mac OS.

Although the Raspberry Pi presents new opportunities to everyone, it can also be a daunting prospect. It comes as a bare circuit board, so to do anything with it, you'll need to add an operating system on an SD card and connect it up to a screen, mouse, and keyboard. To get started, you need to learn a few basics of Linux, or at least get acquainted with LXDE, the graphical desktop. 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* helps you to get started with your Raspberry Pi and teaches you about some of the many fun and inspiring things you can do with it.

About Raspberry Pi For Dummies

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

- Connect up your Raspberry Pi.
- Change its settings so it works optimally for you.
- Discover and install great free software you can use on your Raspberry Pi.
- Use the desktop environment to run programs, manage your files, surf the web, and view your 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.
- Build and publish your first website using the tools on the Raspberry Pi and free tools you can download.
- Create animations and arcade games with the child-friendly Scratch programming language.
- Write your own games and other programs using the Python programming language.
- Get started with electronics, from an introduction to soldering, to the design and creation of sophisticated electronic games, controlled by the Raspberry Pi.

Why You Need This Book

After you shake the Raspberry Pi out of the little electrostatic bag it comes in, what next?

This book answers that question. It enables you to get your Raspberry Pi up and running and also introduces you to some of the great things you can do with it, through satisfying practical projects. With this book as your companion, you can build websites, write games, and create your own electronic gadgets, all without any prior knowledge.

The Raspberry Pi is most likely a bit different compared to other computers you've used, so this book also helps you to do some of the things on your Pi that you expect of every computer, such as playing music and editing documents.

You can learn a lot of this through trial and error, of course, but that can be a frustrating way to spend your time. Using this book as a reference, you can more quickly start using your Raspberry Pi, whatever you plan to do with it.

Foolish Assumptions

Raspberry Pi For Dummies is written for beginners, by which we mean people who have never used a similar computer before. However, we do have to make a few assumptions in writing this book because we wouldn't have enough space for all the 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 the Internet or writing letters.
- 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 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.
- 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 at the world of possibilities that the Raspberry Pi can open up to you!

Other than those 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 adult) are using it for education and entertainment.

How This Book Is Organized

This book is organized into six parts:

- Part I shows you how to set up your Raspberry Pi, including guidance on what else you need; how you download the Raspberry Pi's operating system software and copy it to an SD card; and how you connect everything up. You'll learn how to use the configuration software and log in to your Raspberry Pi.
- Part II gets you up and running with Linux, the operating system that runs on the Raspberry Pi. You'll learn about the desktop environment, which you can use to run programs, manage your files, browse the web, and view your images. Many Raspberry Pi users spend most of their time in the desktop environment, but others want to dig deeper into Linux, learning how to enter text commands to manage the computer and its files. The book also shows you how to do this, so you can exploit the full power of Linux.
- Part III is all about using your Raspberry Pi for work and play. You can't use Windows or Mac OS software on your Raspberry Pi, so you need to find and install some new programs for work, photo-editing, and playing

music and video. You also learn how to build your first website, using HTML and CSS, the languages that underpin every website in the world.

- Part IV teaches you how to write your own programs for the Raspberry Pi, using the two programming languages that come with the operating system. Scratch is highly visual and ideal for making games and animations. After we introduce you to the concepts of Scratch, we show you how you can bring them together to make a shoot-'em-up game. After that, you learn Python, a more powerful programing language that comes with the Raspberry Pi. We'll show you how to create a basic Chatbot that analyzes what you type in and gives intelligent responses (sometimes, at least). After you've mastered the basics of Python, we show you how to write an arcade game using Pygame.
- Part V introduces you to some electronics projects you can undertake with your Raspberry Pi. You learn the basics of electronics theory, how to use a soldering iron, and how the Raspberry Pi can be connected to your own electronics circuits. This section builds on your knowledge of Python to show you how to make two electronic games controlled by the Raspberry Pi, Marble Slalom, and Copycat. The last chapter in this part shows you how to make an analog-to-digital converter that you can use for a wide range of your own electronics projects.
- Part VI is the Part of Tens, a unique feature of the *For Dummies* series. This part contains concise guides to great software you can install on your Raspberry Pi and inspiring projects you can make with it.
- Finally, Appendix A covers troubleshooting and more advanced configuration options of your Raspberry Pi. This gives you solutions for the most common problems people experience, and some guidance on directly editing the configuration files. You might not need this chapter, but it's good to know it's there if things go wrong! Appendix B provides a reference to the GPIO that you can consult when connecting your own electronics projects to the Raspberry Pi.

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, through learning the software that comes with it, to writing your own programs, and finally creating your own electronics projects. Some chapters build on knowledge gained in earlier chapters, especially the sections on Scratch, Python, and all of Part V.

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've included cross-references to help you quickly find what you might have missed. We've also included some signposts to future chapters too, so you can skip ahead to a later chapter if it provides the quickest answer for you.

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.

Arguably, the whole book is 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 we're doing. Sections highlighted with this icon might be worth re-reading to make sure you understand, or you might decide that you don't need to know that much detail. It's up to 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 take-aways, or they are fundamental to the project you're working on.

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

Visit the Book's Website

You can find the dedicated website for this book at `www.dummies.com/go/raspberrypifd`. You can download the files used in the website design, programming, and electronics projects there. That saves you having to retype them, and also gives you a sound base you can build on for your own projects.

Occasionally, we have updates to our technology books. If this book does have technical updates, they will be posted at `www.dummies.com/go/raspberrypifdupdates`.

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` and Sean's is at `www.sean.co.uk`.

Chapter 1

Introducing the Raspberry Pi

In This Chapter

- Getting familiar with the Raspberry Pi
- Figuring out what you can do with a Raspberry Pi
- Determining its limitations
- Getting your hands on a Raspberry Pi
- Deciding what else you need

The Raspberry Pi is perhaps the most inspiring computer available today. Although most of the computing devices we use (including phones, tablets, and games consoles) are designed to stop us from tinkering with them, the Raspberry Pi is exactly the opposite. From the moment you see its shiny green circuit board, it invites you to prod it, play with it, and create with it. It comes with the tools you need to start making your own software (or *programming*), and you can connect your own electronic inventions to it. It's cheap enough that if you break it, it's not going to break the bank, so you can experiment with confidence.

Lots of people are fired up about its potential, and they're discovering exciting new ways to use it together. Dave Akerman (`www.daveakerman.com`) and friends attached one to a weather balloon and sent it nearly 40 kilometers above the earth to take pictures of earth from near space using a webcam.

Professor Simon Cox and his team at the University of Southampton connected 64 Raspberry Pi boards to build an experimental supercomputer, held together with 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.

The Pi is also being used at the frontier of exploration. The FishPi project (`www.fishpi.org`) aims to create a vessel that can navigate across the Atlantic unmanned and take environmental measurements along the way, communicating with base by satellite. London Zoo is looking at using the Raspberry Pi in a device to detect and photograph animals in their natural habitats, called EyesPi.

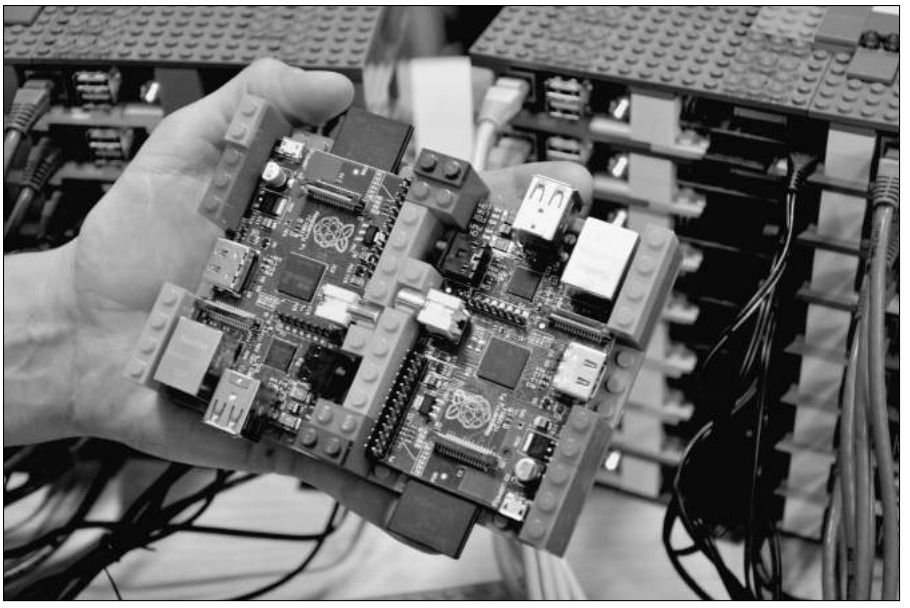


Figure 1-1: Two of the Raspberry Pi boards used in the University of Southampton's supercomputer, with the rest of the supercomputer in the background.

Courtesy of Simon Cox and Glenn Harris, University of Southampton

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 became friendlier, and we started to use mice and windows, we didn't need those skills any more, and we 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 of programming than students of the past did. Upton and his university colleagues hatched the idea of creating a computer that would come with all the tools needed to program it, and would sell for a target price of $25. It had to be able to do other interesting things too so that people were drawn to use it, and 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. Early in 2013, it reached the milestone of one million sales.

Getting Familiar with the Raspberry Pi

When your Raspberry Pi arrives, you'll see it's a circuit board, about the size of a credit card, 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. It's a big part of its appeal, though: most of the cases you can buy for the Raspberry Pi are transparent because people love the look of it.

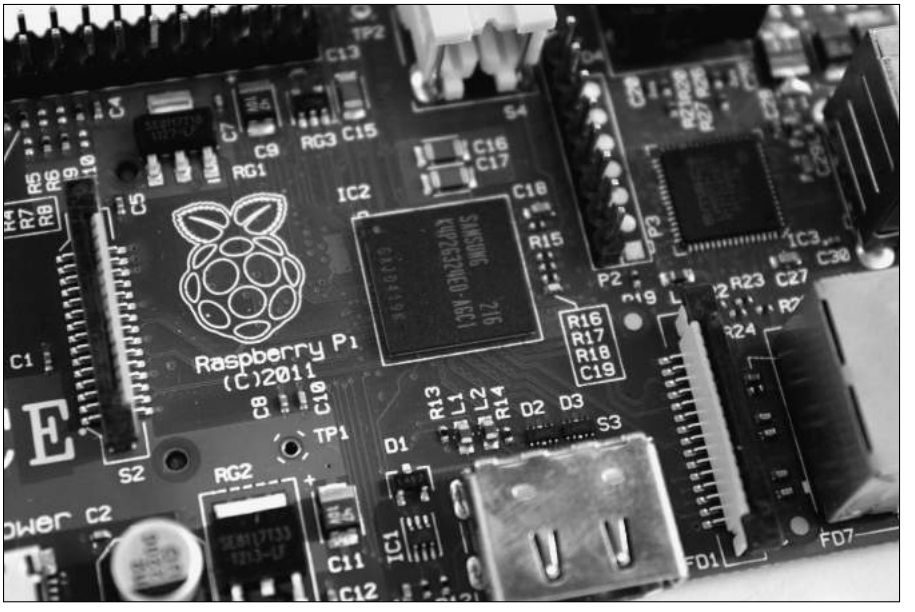


Figure 1-2: Up close with the Raspberry Pi.

There are two versions of the Raspberry Pi: the Model B (which was released first) and the Model A. The differences between the two are that the Model B has two USB sockets (whereas the Model A only has one), the Model B has an Ethernet socket, and editions of the Model B released after October 2012 contain twice the memory (512MB, compared to 256MB on the Model A and the first batches of the Model B). The Model A sells for $25, whereas the Model B sells for around $35.

The Raspberry Pi was made possible in part by the advances in mobile computer chips that have happened in recent years. At its heart is a Broadcom BCM2835 chip that contains an ARM central processing unit (CPU) and a Videocore 4 graphics processing unit (GPU). The CPU and GPU share the memory between them. The GPU is powerful enough to be able to handle Blu-ray quality video playback.

Instead of running Windows or Mac OS, 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. Instead of 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.

Unless you already use Linux, you won't be able to run the software you have on your other computers on your Raspberry Pi, but a lot of software for Linux is free of charge.

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.

When you switch it on, it has a text prompt (see Chapter 5), but you can use a graphical windows desktop to start and manage programs. You can use it for browsing the Internet (see Chapter 4), 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 9), or for playing games. You can use the built-in software to build a website (see Chapter 8). 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 10), which 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 12), 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 a General Purpose Input/Output (GPIO) port 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 V, we show you how to build some electronic games controlled by the Raspberry Pi.

Determining Its Limitations

For something that costs so little, the Raspberry Pi is amazingly powerful, but it does have some limitations. Although you probably use it as a desktop computer, its power is closer to a mobile device (like a tablet) than a modern desktop PC.

By way of example, the Raspberry Pi Foundation says the Pi's overall performance is comparable with a PC using a 300 MHz Pentium 2 processor, which you might have bought in the mid to late nineties, except that the Raspberry Pi has much better graphics. The memory of the Raspberry Pi is more limited than you're probably used to, with just 512MB or 256MB available. You can't expand that with extra memory in the way you can a desktop PC.

The graphics capabilities lag behind today's market somewhat too: The Raspberry Pi Foundation says the Pi's graphics are roughly the same as the original Xbox games console, which was released 10 years ago.

Both the Pentium 2 PC and the original Xbox were fine machines, of course, for their time. They're just not as snappy as we're used to, and that's where you might experience some problems. You might find that the Pi can't keep up with the demands of some modern software and that some programs don't run fast enough to be useful on it. However, it's easy to find programs, try them, and remove them if they're no good (see Chapter 5), and plenty of programs for work and play run well on the Raspberry Pi (see Chapter 18).

If you already have another computer, the Raspberry Pi is unlikely to usurp it as your main machine. But the Pi gives you the freedom to try lots of things you probably wouldn't dare to try, or wouldn't know how to try, with your main PC.

Getting Your Hands on a Raspberry Pi

The Raspberry Pi was created by the Raspberry Pi Foundation, a charity registered in the UK. The charity's six trustees funded the manufacture of the first large batch themselves, but it sold out rapidly so it quickly became clear that they needed something that would scale better.

The Foundation now licenses the design of the Raspberry Pi to RS Components (`www.rs-components.com`) and Premier Farnell, which uses the brand name Element 14 (`www.element14.com/community/groups/raspberry-pi`). Both companies fund and manage the manufacture of the Raspberry Pi, market and sell it, and look after their customers. They accept orders through their websites and are able to offer a number of the accessories you might also need.

It's possible that more companies will license the design of the Pi in the future, so check the Raspberry Pi Foundation's website at `www.raspberrypi.org` for current links to stores that sell the Pi.

Second-hand Raspberry Pis can be bought on eBay (`www.ebay.com`), but we would recommend getting a new one so you benefit from the customer support available, and have the peace of mind that it hasn't been damaged by the previous owner.

Deciding 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 about $25–$35, so you'll need to scavenge or buy a few other bits and pieces. I 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. That said, you might find they're not fully compatible with the Raspberry Pi and you need to buy replacements to use with the Pi.

Here's a checklist of what else you might need:

- **Monitor:** The Raspberry Pi has a high definition video feed and uses an HDMI (high definition multimedia interface) 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 aren't officially supported by the Raspberry Pi Foundation, but devices are available to convert the HDMI signal into a VGA one. If you're thinking of buying a converter, check online to see whether it works with the Raspberry Pi first. 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. If your monitor is connected using a blue plug, and the connector has three rows on 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 through an RCA cable, 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. If a TV is your only option, see Appendix A for advice on tweaking the settings to get the clearest possible picture. It's better to use a computer monitor if you can, though.

- **USB hub:** The Raspberry Pi has one or two USB sockets (depending on the model you get), but you should use a powered USB hub for two reasons. Firstly, you're going to want to connect other devices to your Pi at the same time as your keyboard and mouse, which use two sockets. And secondly, you should use a USB hub because it provides external power to your devices and minimizes the likelihood of experiencing problems using your Raspberry Pi. Make sure your USB hub has its own power source independent of the Raspberry Pi.
- **USB keyboard and mouse:** The Raspberry Pi only supports USB keyboards and mice, so if you're still using ones with PS/2 connectors (round rather than flat), you need to replace them.

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

- **SD card:** The Raspberry Pi doesn't have a hard disk built in to it, so it uses an SD card 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 would recommend a 4GB SD card as a minimum, and SD cards are cheap enough now that it's worth getting an 8GB or 16GB one. Even that isn't much space for your files and data compared to the hard drive on a modern computer, but you can use other storage devices such as external hard drives with your Raspberry Pi too. SD cards have different class numbers that indicate how fast you can copy information to and from them. Element14 sells a class 4 SD card with the operating system preloaded on it (see Figure 1-3), and RS Components recommends a class 6 SD card to use with the Raspberry Pi.
- **SD card writer for your PC:** Many PCs today have a slot for SD 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 card writer to connect to your computer. You'll use it to copy Linux 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. Alternatively, you can buy an SD card that has the recommended version of Linux already on it for use with the Raspberry Pi. That means you can avoid the expense of an SD card writer, but it doesn't enable you to experiment with the different operating systems available for the Pi (see Chapter 2).

Figure 1-3: A SD card preloaded with the Raspberry Pi operating system.

- **USB keys:** *USB keys* (also known as *flash drives* or *memory sticks*) are fairly cheap and high capacity now (a 64GB USB key is readily affordable), which makes them an ideal complement to your Raspberry Pi. You can transfer files between your PC and your Raspberry Pi using a USB key, too.
- **External hard drive:** If you want lots of storage, perhaps so 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.
- **Speakers:** The Raspberry Pi has a standard audio out socket, compatible with headphones and PC speakers that use a 3.5mm audio jack. You can plug your 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 goes between 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, but note that this doesn't work if you use a DVI monitor.
- **Power supply:** The Raspberry Pi uses a Micro USB connector for its power supply, and is theoretically compatible with a lot of mobile phone and tablet chargers. In practice, many of these can't deliver enough current (up to 700 milliamperes), which can make the Raspberry Pi perform unreliably. The resistance in the cables that connect the Pi to the power supply varies greatly too, and this can prevent peripherals like the

mouse from working. It's worth checking whether you have a charger that might do the job (it should say how much current it provides on it), but for best results, we recommend buying a compatible charger from the same company you got your Raspberry Pi from. Don't try to power the Pi by connecting its Micro USB port to the USB port on your PC with a cable, because your computer probably can't provide enough power for your Pi.

- **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. You can buy plastic cases on eBay (`www.ebay.com`), most of which are transparent so you can still admire the circuitry and see the Pi's LED lights. These cases typically come as simple kits for you to assemble. The Pibow (`www.pibow.com`) is one of the most attractively designed cases, with layers of plastic giving it a rainbow look, side-on (see Figure 1-4). It's designed by Paul Beech, who designed the Raspberry Pi logo. You don't have to buy a case, though. You can go without or make your own (see Chapter 3). Whatever case you go with, make sure you can still access the GPIO pins so you can experiment with connecting your Pi to electronic circuits and try the projects in Part V of this book.

Figure 1-4: The Pibow Raspberry Pi case.

Pibow™ Pimoroni Ltd (www.pibow.com)

- **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 television), an audio cable (if connecting the audio jack to your TV or stereo), and an Ethernet cable (for networking). You can get these cables from an electrical components retailer and might 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.

The Raspberry Pi has been designed to be used with whatever accessories you having lying around to minimize the cost of getting started with it but, in practice, not all devices are compatible. In particular, incompatible USB hubs, keyboards, and mice can cause problems that are hard to diagnose.

A list of compatible and incompatible devices is maintained at `http://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.

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 a cheap device, but buying a keyboard, mouse, USB hub, SD cards, and cables can easily double or triple your costs, and you might have to resort to that if what you have on hand turns out not to be compatible.

Chapter 10

Introducing Programming with Scratch

In This Chapter

- Starting Scratch
- Understanding the Scratch screen layout
- Positioning and resizing your sprite
- Making your sprite move
- Changing your sprite's appearance
- Adding sounds and music

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 only takes a few minutes to write your first Scratch program.

In this chapter, we introduce you to Scratch so you can start to experiment with it. In Chapter 11, we show you how to use Scratch to make a simple arcade game.

17_9781118554210-ch10.indd 173 3/6/13 9:41 AM

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 12, you'll learn 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 or do some sophisticated mathematics. 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.

Starting Scratch

You access Scratch from the desktop environment, so switch on your Raspberry Pi and then use `startx` to access it (see Chapter 4 for a guide to using the desktop environment).

To start Scratch, either double-click its icon on the desktop (which shows the head of a smiley orange cat), or select it from your Programs menu in the bottom left of the screen. You can find Scratch in the Programming folder.

Understanding the Scratch Screen Layout

Scratch divides the screen into four main areas, as you can see in Figure 10-1. In the top right is the Stage, where you can see your game or animation take shape. There's a cat on it already, so you can get started straight away by making him do things, as you'll see in a minute.

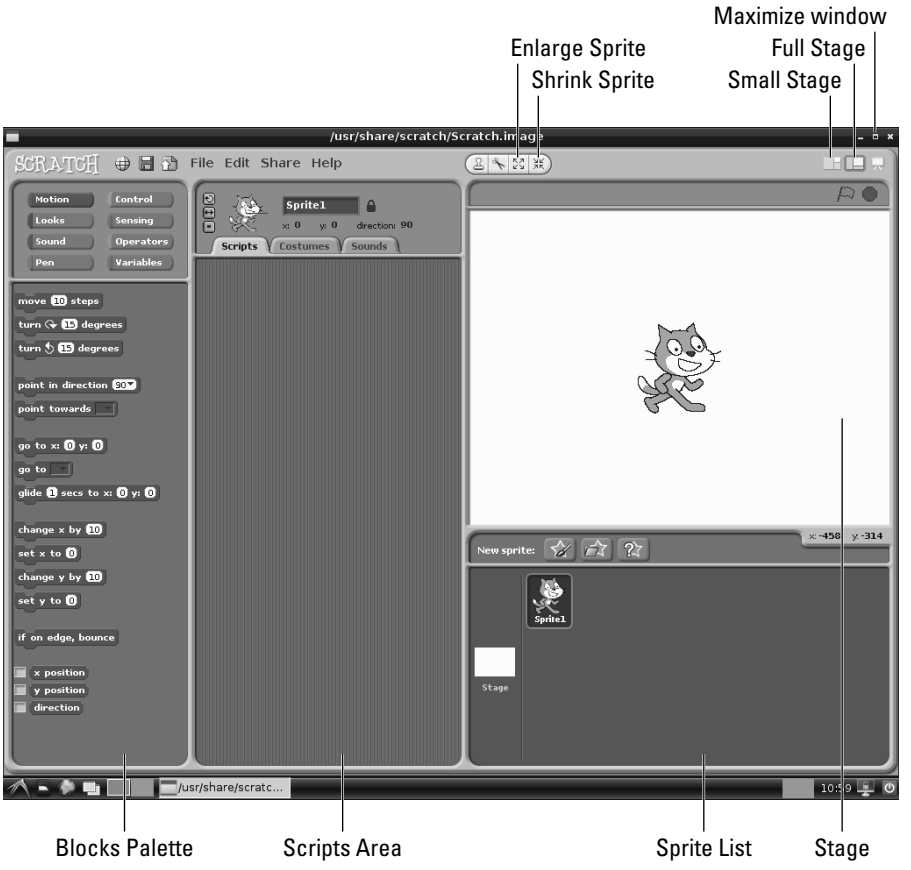


Figure 10-1: The screen layout when you first start Scratch.

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

The bottom right area 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. On the left, you can see the Blocks Palette, which currently shows the Motion blocks, which include instructions to move 10 steps, rotate, go to a particular grid reference, and point in a particular direction.

The tall middle panel is the Scripts Area. This is where the magic happens! You assemble your program in this space, by dragging blocks into it from the left.

You can use two buttons in the top right (indicated in Figure 10-1) to toggle the size of the Stage between full and small. When the Stage is small, the Scripts Area is bigger, so you might find that useful when you're writing scripts later in this chapter.

You'll find it easier to use Scratch if you maximize it so it fills the screen. Click the button in the top right of its window, as indicated on Figure 10-1.

Positioning and Resizing Your Sprite

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.

You can also resize it. Two buttons above the Stage (indicated in Figure 10-1) are used to enlarge or shrink a sprite. Click one of them, and your mouse pointer changes to arrows pointing outwards (for enlarging) or inwards (for shrinking). Click your sprite on the Stage repeatedly to change its size to what you want.

When you've finished resizing, click something that isn't a sprite to return the mouse pointer to normal and stop resizing.

Making Your Sprite Move

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

If your cat goes somewhere 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. You can fix rotation too by clicking the tiny cat at the top of the Scripts Area, holding down the mouse button, and rolling your mouse in a circle pattern on the desk.

Not all of the blocks will work at the moment. Some of them need to be combined with other blocks, or only make sense at certain times. There's no harm in experimenting, however. If you click something that doesn't work, you might get an error message, but 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 your 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 (see Figure 10-2):

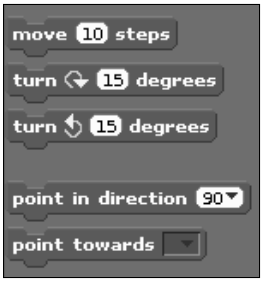


Figure 10-2: The directional movement blocks.

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

- **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 your sprite 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 bigger numbers spoil the illusion of movement.
- **Turn Right or Left 15 Degrees:** This block rotates your sprite. As with the number of steps, you can edit the number to change the degree by which your sprite is rotated. Your sprite 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 10-3). 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 arrow in the right of the number box, it gives you a menu from which you can select the four main directions, but you can enter any number. You might be wondering whether you can enter `270` to point left, and the answer is that it works, but it can cause errors in your programs. If you turn your cat to direction `270` and then ask Scratch which way your cat is facing, it tells you `-90`. To avoid any inconsistencies like this, keep your direction numbers in the range –179 to 180.

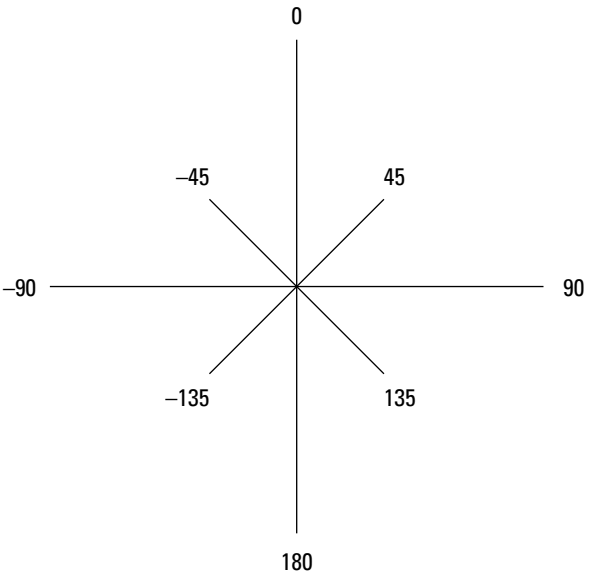


Figure 10-3: The number of degrees used to face in different directions.

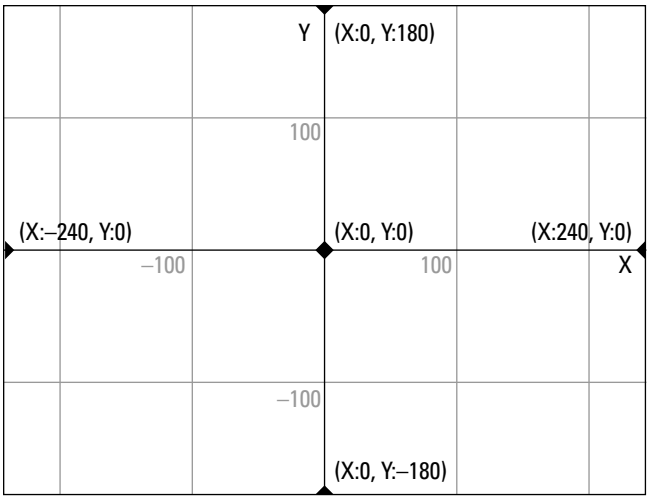
- **Point Towards:** You can also tell the sprite to point towards the mouse pointer or another sprite. Use the menu in this block to choose what you would like your sprite to point towards.

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

You can also move and position your sprite using grid coordinates. That makes it easy to position your sprite at an exact place on the screen, irrespective of where it currently is.

Every point on the Stage has two coordinates, an X position (for 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 his day, is where X equals 0 and Y equals 0. Figure 10-4 provides a quick visual reference of how the coordinates work.



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

Figure 10-4: The grid coordinates on the Stage.

When you move your mouse over the Stage, the grid reference of your mouse pointer is shown just underneath the Stage on the right.

Six Motion blocks use the X and Y coordinates (see Figure 10-5):

- **Go to x:0 y:0:** You can use this block to position your sprite at a specific point on the Stage. By default, it returns a sprite to the center of the screen (x=0, y=0). Edit the numbers for X and Y to position your sprite somewhere else.
- **Go to:** Use this block to move your sprite to the mouse pointer's location, or to the location of another sprite if you have more than one.
- **Glide 1 secs to x:0 y:0**: 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. 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).
- **Change X by 10:** This moves your sprite `10` units 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 10 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.

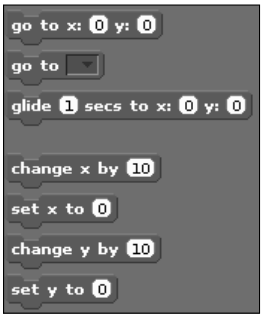


Figure 10-5: The blocks used for moving sprites using grid coordinates.

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

Don't forget that 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 has got to and which direction it's facing, but you can show the values for its X position, Y position, and direction on the Stage. Select the boxes at the bottom of the Blocks Palette to do this (see Figure 10-6). They slow your program down, and they clutter up the screen a bit, but they can be essential tools for testing when you're creating a game.

Figure 10-6: The blocks used to show sprite information on the Stage.

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

Changing Your Sprite's Appearance

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

Using costumes

One way to think of sprites is like 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 see the costumes for your sprite by clicking the Costumes tab at the top of the Scripts Area, as shown in Figure 10-7. If you want to modify the cat's appearance, you can click the button to edit one of the costumes, or if you want to create a new animation frame, you can click the Copy button beside a costume and then edit the bits you want to change.

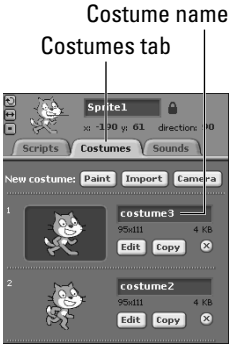


Figure 10-7: You can change a sprite's appearance by giving it a new costume.

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

It doesn't matter so 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 sprites meaningful names. It's much easier to remember that the costume with the name *gameover* 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 (see Figure 10-7) and type its new name.

In the Blocks Palette, there are two blocks you can use to switch between costumes (see Figure 10-8):

- **Switch to Costume:** 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 it runs out, it goes back to the first one again.

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

Using speech and thought bubbles

To see the blocks that affect a sprite's appearance, click the Looks button above the Blocks Palette (indicated in Figure 10-8).

Figure 10-8: 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 http://scratch.mit.edu.

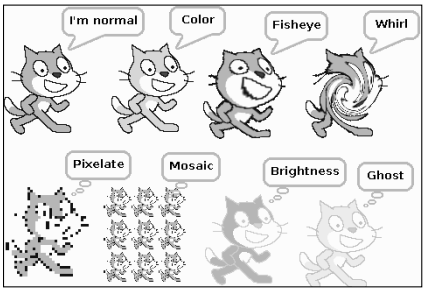
Scratch includes four blocks you can use to show a speech bubble or a thought bubble on screen, as you can see in Figure 10-8. These are great for giving a message to the player or viewer. You can edit the word in the block (`Hello` or `Hmm...`) to change the text in the bubble. Figure 10-9 shows the speech bubbles (top row) and thought bubbles (bottom row) 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's 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 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 10-9, I've used eight sprites to demonstrate them on the Stage. The Color effect changes the sprite's color palette, turning orange to green in the case of the cat. The Fisheye effect works like a fisheye 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 increases the intensity of the colors (turning the cat's black outline silver while softening the orange) and the Ghost effect fades all the colors out evenly.



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

Figure 10-9: The different graphic effects you can apply to your sprite.

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

- **Change Color Effect by 25:** You can select which effect you want to change (by default, it's the color effect), and enter the amount of it you want to apply, as a percentage (by default, 25 percent). You can use negative numbers to reduce the extent to which the effect is applied to your sprite.
- **Set Color Effect to 0:** Use this block to set a chosen effect to a specific percentage. 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 the graphic effects you've applied to a particular sprite, so it looks normal again.

The graphic effects look great, but they are quite slow. They're best used in moderation for special moments in your animation or game; otherwise, they make it appear unresponsive.

Resizing your sprite

Earlier in this chapter, we showed you how to change the starting size of your sprite on the Stage. You can use blocks to issue instructions to change its size too, so you could make it get 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 your size to a percentage of its original size, so with the default value of 100 percent, it effectively resets any resizing you've done.

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 displayed other sprite information (see "Showing sprite information on the Stage" earlier in this chapter) there. 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 space ship 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 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.

Sometimes sprites might get on top of each other. You can use the Go to Front block to make a sprite appear on top of all the others. To push a sprite backwards and allow others to appear on top of it, use the Go Back `1` Layers block.

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. There are effects there for most occasions, and many of them are a natural partner for one of the sprites that Scratch provides.

At the time of writing, some of the sounds provided are in MP3 format, but Scratch can only play those that are in WAV format. If you get a message saying a sound is in an unrecognized format, try another sound.

Here are the two steps to using sounds in your Scratch project:

1. **Import the sound to your sprite. To do this, click the Sounds tab above the Scripts Area, as shown in Figure 10-10, and then click the Import button. Browse the provided sounds. You can click a file once to hear a preview of it, and click it twice to bring it into your sprite.**

   After you've imported a sound, click the speaker beside it to preview it, or click the X button to delete it from your project. If you a delete a sound in this way, it remains on your SD card so you can import it again later.

2. **Use one of the blocks to play a sound. To see the Sound blocks, click the Sound button at the top of the Blocks Palette first.**

   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 any movement or other blocks on the same sprite until the sound has finished playing.

In Chapter 11, we cover how to use multiple sprites in a project. 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.

There are also blocks you can use to create music using Scratch, using drums and pitched instruments. Notes are numbered, with C being 60, C# being 61, D being 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. 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. There is also a block called Set Instrument to `1` which you can use to change the instrument, although at the time of writing, this doesn't work on the Raspberry Pi.

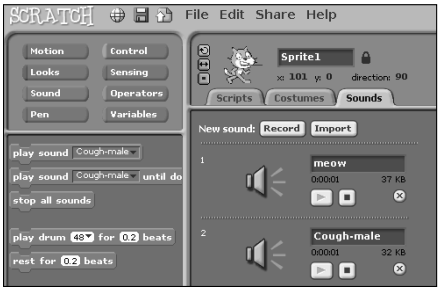


Figure 10-10: Adding sound effects to your sprite.

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

Creating Scripts

Clicking blocks in the Blocks Palette is one way to issue commands to Scratch, but you're not really programming. 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 only work 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 Scripts Area in the middle of the screen. Most blocks mentioned so far have a notch on the top of them and a lug on the bottom of them, 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.

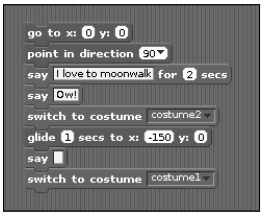
You put your blocks in the order you want Scratch to run them, starting at the top and working your way down. It's a bit like making a to-do list for the computer.

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

You can have multiple different scripts in the Scripts 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 11), each sprite has its own Scripts Area and scripts there to control it.

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

The moonwalk is the dance popularized by Michael Jackson where the dancer looks like he's walking forwards, but actually moves backwards. Figure 10-11 shows an example script to make our cat moonwalk across the Stage. The first two lines in the script reset the cat to the middle of the screen, facing right. She tells us she loves to moonwalk and then lets out a little whoop like Michael Jackson, which she keeps up for the duration of the dance. The costume switch changes the position of the cat's legs, and it then glides 150 units to the left. We close the speech bubble by using the Say block with nothing in it, and then switch back to the other costume, which makes the cat's legs move back to their default position. Give it a go!



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

Figure 10-11: This is how you make a cat moonwalk. Ow!

Using the Wait Block to Slow Down Your Sprite

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

If you click the Control button at the top of the Blocks Palette, you can find a set of yellow blocks that are used to govern when particular things happen. You'll learn more about these in Chapter 11, but for now, it's worth knowing that there is a block here that enables you to wait for a certain number of seconds. Drag this into your script where necessary to introduce a delay so 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).

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

Saving Your Work

Remember to save your work so you can come back to it again later. You can find the option to save in the File menu at the top of the screen, or you can click the floppy disc icon in the top left.

When the Save dialog box opens (see Figure 10-12), you'll see buttons on the left to choose from various places you could save your file, although you might not have permission to use all of them (see Chapter 5 for more on permissions). I recommend you use the Scratch folder inside your Pi directory.

On the right, you can add your name and some project notes to remind you what the project was about later. You can see and edit the project notes associated with a file by going through the File menu when you're working on a program.

Figure 10-12: Saving your work so you can come back to it later.

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

Index

• Symbols •

• A •

• B •

• C •

• D •

• E •

• F •

• G •

• H •

• I •

• J •

• K •

• L •

• M •

• N •

• O •

• P •

• Q •

• R •

• S •

• T •

• U •

• V •

• W •

• X •

• Y •

• Z •

Join the Raspberry Revolution and start having fun with your Raspberry Pi — today!

As small as a credit card and ultra-affordable, the Raspberry Pi packs plenty of power into a pocket-size computer. But getting the hang of the Raspberry Pi can be tricky. *For Dummies* is here to help. You'll discover how to use your Raspberry Pi to work and play, edit photos, watch videos, and even craft your own programs! No programming experience? No problem. This friendly guide makes mastering this cool, compact computer as easy as pie.

- ***Ready, set, go* — *download the operating system software, hook everything up, and configure your Raspberry Pi***
- ***You'll love Linux* — *find your way around the Linux shell, make the most of its desktop environment, and use it to manage your new computer***
- ***Have a little fun* — *play videos, listen to your favorite music, create documents, and even build a brilliant website***
- ***The Pi's the limit* — *create your own software with Scratch and Python programming***
- ***Conquer the world* — *master Raspberry Pi electronics in no time — from soldering your own circuits to creating games, musical instruments, and interfaces***

Open the book and find:

- **How to download and install Raspbian, the Raspberry Pi OS**
- **Help connecting a keyboard, mouse, and monitor**
- **Tips on editing images and creating web pages**
- **How to play music and videos on your Raspberry Pi**
- **Advice on finding and installing free software**
- **How to create your own arcade games and animations**
- **Exciting electronics projects to try out your skills**

Cover image: © Dr. Andrew Robinson

Sean McManus is the author of many technology and business books and a former staff writer at *Internet Magazine*. He has written tutorials and articles for, among others, *Internet Works*, *Business 2.0*, and *Personal Computer World*. **Mike Cook** is a veteran tech author and freelance consultant specializing in all things physical computing.

Go to Dummies.com®
for videos, step-by-step examples, how-to articles, or to shop!

ISBN 978-1-118-55421-0

FOR DUMMIES®
A Wiley Brand

Also available as an e-book