The Definitive Guide to ImageMagick

Michael Still

Apress®

The Definitive Guide to ImageMagick

ISBN (Hardback): 1-59059-590-4

Library of Congress Cataloging-in-Publication data is available upon request.

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Matt Wade
Technical Reviewer: Doug Jackson
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser
Project Manager: Kylie Johnston
Copy Edit Manager: Nicole LeClerc
Copy Editor: Kim Wimpsett
Assistant Production Director: Kari Brooks-Copony
Production Editor: Linda Marousek
Compositor and Artist: Kinetic Publishing Services, LLC
Proofreader: Kim Burton
Indexer: Carol Burbo
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code section and also at `http://www.stillhq.com`.

For my ever-patient and loving family—Catherine, Andrew, and Matthew—who put up with me being distracted by random projects, including this one. Thanks to my friends who encouraged me along the way and all those people who asked great questions that I didn't find enough time to answer. I hope your answers are somewhere in here.

Oh, and thanks, Dad, for coming over to provide moral support. What would I have done without you to mind the cat and help me drink all the port? Thanks to Mum as well for all the support during my childhood; it positioned me well to undertake a project like this.

Contents at a Glance

Contents

Foreword

I swear by my life and my love of it that I will never live for the sake of another man, nor ask another man to live for mine.

—John Galt in *Atlas Shrugged*, by Ayn Rand

Like many software projects, ImageMagick lacks good documentation. I designed it to be as intuitive as possible so most users without the benefit of this book could surmise that the following command converts an image in the JPEG format to one in the PNG format:

```
convert image.jpg image.png
```

However, few would realize that the next command turns a flat, two-dimensional label into one that looks three-dimensional with rich textures and simulated depth:

```
convert -background black -fill white -pointsize 72 label:Magick +matte ➥
\( +clone -shade 110x90 -normalize -negate +clone -compose Plus -composite \) ➥
\( -clone 0 -shade 110x50 -normalize -channel BG -fx 0 +channel -matte \) ➥
-delete 0 +swap -compose Multiply -composite button.gif
```

ImageMagick has been in development for nearly 20 years, and for 20 years users of the project have rightly complained about its lack of documentation. I have never had the opportunity to write a book, because I am perpetually consumed with answering ImageMagick questions, fixing bugs, and adding enhancements. So when Matt Wade from Apress approached me about writing a book on ImageMagick, I did the proverbial happy dance.

Apress did well finding Michael Still to present ImageMagick to you. I know of Michael because of some articles he wrote on ImageMagick for IBM DeveloperWorks. I often refer ImageMagick users to those articles when they want a gentle introduction to using ImageMagick from the command line.

ImageMagick started with a request by my DuPont supervisor, Dr. David Pensak, to display computer-generated images on a monitor capable of showing only 256 unique colors simultaneously. In 1987, monitors that could display 24-bit true-color images were rare and quite expensive. There were a plethora of chemists and biologists at DuPont but few computer scientists to confer with. Instead, I turned to Usenet for help and posted a request for an algorithm to reduce 24-bit images to 256 colors. Paul Raveling of the USC Information Sciences Institute responded with not only a solution, but one that was already in source code and available from his FTP site. Over the course of the next few years, I had frequent opportunities to get help with other vexing computer science problems I encountered in the course of doing my job at DuPont. Eventually, I felt compelled to give thanks for the help I received from the knowledgeable folks on Usenet. I decided to freely release the image-processing tools I developed to the world so that others could benefit from my efforts.

In 1990 few freely available image-processing tools existed, so I expected an enthusiastic reception. Before a release was possible, Dr. Pensak had to convince upper management at

DuPont to give away what they might have perceived as valuable intellectual property. I suspect they agreed simply because ImageMagick was not chemically or biologically based, so they did not understand its value to the company. Either way, ImageMagick would not be available today without DuPont's permission to distribute it. ImageMagick was posted to Usenet's `comp.archives` group on August 1, 1990.

After ImageMagick's release, I got the occasional request for an enhancement, a report of a bug, or a contribution to the source base. In the mid-1990s, I released the culmination of these efforts, ImageMagick 4.2.9. At the time, I thought ImageMagick was complete. Thousands of folks worldwide were using it, and it was even showing up as part of a new operating system being distributed freely called Linux.

The next generation of ImageMagick, version 5, started when Bob Friesenhahn contacted me and began suggesting ways to improve it. Bob had seemingly boundless energy, questions, and ideas. He suggested I revamp ImageMagick 4.2.9, so in addition to the command-line tools, it should have a usable application programming interface (API) so users could leverage the image-processing algorithms from other languages or scripts. Bob also wrote a C++ wrapper for ImageMagick called Magick++ and began contributing enhancements such as the module loader facility, automatic file identification, and test suites. In the meantime, the project picked up a few other notable contributors: Glenn Randers-Pehrson, William Radcliffe, and Leonard Rosenthol. By now, ImageMagick was being utilized by tens of thousands of users, who reacted gruffly when a new release broke an existing API call or script. The other members of the group wanted to freeze the API and command line, but I was not quite ready, since ImageMagick was not quite what I had envisioned it could be. Bob and others created a fork of ImageMagick called GraphicsMagick. I alone continued to develop ImageMagick.

I did not work alone for long. Anthony Thyssen contacted me about deficiencies in the ImageMagick command-line programs. He pointed out that the command line was confusing when dealing with more than one image. He suggested an orderly, well-defined method for dealing with the command line, and this became ImageMagick 6 (the current release). His efforts are detailed at his Web pages, Examples of ImageMagick Usage, at `http://www.cit.gu.edu.au/~anthony/graphics/imagick6/`. In addition to this book, I highly recommend you peruse his site. He has illustrated the power of ImageMagick in ways even I did not know were possible.

It has been nearly 20 years since ImageMagick was first conceived, and it looks likely that it will be here for another 20 and beyond. The command line and the API are stable, but there is still work to do. We're currently working on improving the Scalable Vector Graphics (SVG) support and adding better support for video formats. And, of course, I always have questions from the community to keep me busy. In fact, I better get back to work—while I was writing this foreword I received several e-mails with ImageMagick questions. I am grateful that in the future, I'll be able to answer most ImageMagick questions simply by pointing people to this book.

Cristy
Principal ImageMagick Architect
November 2005

About the Author

MICHAEL STILL has been working on imaging applications for eight years and has been programming for many more. His interest in imaging applications started with his employment at IPAustralia, the Australian patent, trademark, and designs office, where he was tasked with modifying an open source PDF-generation library to support TIFF images. This developed into a long-term series of projects using custom imaging code and ImageMagick to implement a line of business systems.

During this time, Michael was responsible for imaging databases, including a database of nine million TIFF files for the Australian Patent Office and a database of all images associated with trademarks in Australia. He also wrote his Panda PDF-generation library, as well as a variety of other open source imaging tools, in this time. You can find his open source code at `http://www.stillhq.com`.

Michael has written a number of articles on ImageMagick for IBM DeveloperWorks (`http://www.ibm.com/developer/`). He has also presented at a variety of conferences and was previously the maintainer of the `comp.text.pdf` frequently asked questions (FAQ) document.

Michael has recently accepted a job with Google as a systems administrator. His experience involves developing large-scale systems, performing systems administration of vertical systems (many of which he developed), and solving other interesting-sounding engineering problems. His previous employer was TOWER Software, developers of a leading Electronic Document and Records Management (EDRM) product, where he worked on imaging problems, as well as a variety of server functionality, including several Web-based products.

About the Technical Reviewer

DOUG JACKSON has worked in the IT industry since 1985 in fields ranging from hardware design, communications, programming, systems administration, and IT security to project management and consulting. During this time, he has become fluent in a number of programming languages, including C/C++, Java, Assembler, and Forth, on both Microsoft and Unix systems. Doug first encountered ImageMagick in 1997 while writing large-scale image-processing and image-viewing software for the Australian Patent Office.

When Doug isn't being an information security consultant, he enjoys teaching the fine art of sailing to Cub Scouts, playing guitar, and solving hardware puzzles with PIC microprocessors. He is married to Megan, arguably the most wonderful and patient lady on the planet, and has two terrific daughters (Cate and Siân).

Acknowledgments

It always seemed corny to me that authors thank the usual suspects for helping with the production of their book. They normally thank the publisher's editorial team, their families, and perhaps their workmates. My problem is that I now discover these sentiments are genuinely true.

If it weren't for Matt Wade initially contacting me and pitching the project, I wouldn't have ever started. If it weren't for the able assistance of Tina Nielsen and my project manager, Kylie Johnston, the project would have faltered along the way. If it weren't for the able review of the manuscript by Doug Jackson and Matt Wade, then this book's content would have suffered. There is also, of course, the layout team, which has produced such a wonderful-looking finished project, especially Linda Marousek, who was my contact point with that team.

Then there's my family, who have gone out of their way to make my life easier while writing the book. Be it leaving Daddy alone for a bit to hack on some sample code or just understanding when I was dazed and confused after a day of writing—thanks, Catherine, Andrew, and Matthew.

My workmates were instrumental, too; without the encouragement of Gordon Taylor, Anthony Drabsch, Simon Dugard, Chris Crispin, Grant Allen, and Lindsay Beaton, I probably wouldn't have let Matt talk me into writing the book.

Andrew Pollock deserves a special mention for providing the hosting for my site and the blog for this book. Many thanks for your patient support and advice.

I want to save two final special acknowledgments to last—Kim Wimpsett was my copy editor, and I never imagined that having my own personal English grammar coach would be such fun. American English isn't my first language (I'm an Australian, and we do English the British way, which is of course better), and I didn't appreciate all the subtle differences until Kim helped me out. The book flows better and makes more sense because of Kim's input.

Finally, this book would have nothing to talk about if it weren't for Cristy and all the other contributors to ImageMagick over the years. ImageMagick is an incredibly deep product, which makes it wonderful to write about. The efforts to which the team has gone to make a product that works cannot be underestimated. Cristy recommended Anthony Thyssen, Bob Friesenhahn, Glenn Randers-Pehrson, and William Radcliffe as being instrumental in the development of ImageMagick and thus deserving of my thanks. Thanks, guys.

I'm sure I've forgotten to thank some people here, and I apologize to them for that. Thanks, folks.

Introduction

The ideal reader of this book is someone with immediate imaging needs who is prepared to either use command-line tools or use the ImageMagick programmer's interface to write code. Many of the concepts demonstrated are also available in the ImageMagick graphical tools, but almost all the examples in this book focus on the command-line tools.

This book provides hundreds of working examples of how to use ImageMagick for everyday problems, as well as the theory necessary to understand what's happening in those examples. I recommend you install ImageMagick before reading this book so you can work along with the examples provided. (Chapter 1 covers how to install ImageMagick for the first time.)

How This Book Is Structured

This book starts by describing how to install ImageMagick on your system and then covers how to configure it. After that, I launch into covering the ImageMagick command-line tools. Complete coverage isn't possible, however, because ImageMagick is so rich. After I've covered the command-line tools, I show working examples of four applications developed with some of the ImageMagick APIs.

The chapter breakdown is as follows:

Chapter 1, "Installing and Configuring ImageMagick": Chapter 1 discusses how to install and configure ImageMagick on Microsoft Windows and Unix machines, including how to install binary versions, what those packages are likely to be called in your Linux distribution, and how to compile ImageMagick from source on both Unix and Microsoft Windows operating systems.

Chapter 2, "Performing Basic Image Manipulation": Chapter 2 covers simple image manipulations such as resizing, sample, cropping, scaling, thumbnailing, and so forth. This chapter contains information about all the ImageMagick transformations used to create smaller or larger versions of an image. To discuss these topics, the chapter also introduces the differences between raster and vector image formats and how raster formats are encoded.

Chapter 3, "Introducing Compression and Other Metadata": In Chapter 3, I discuss compression options for image files, show how to use ImageMagick to change the compression used for a file, and provide recommendations about which file format to use in various scenarios. I'll also discuss file formats that can contain more than one image per file, show how to handle animations, and discuss the metadata you can associate with image files.

Chapter 4, "Using Other ImageMagick Tools": Chapter 4 covers the various other ImageMagick tools that aren't covered extensively in the rest of the book. Five chapters in the book cover the `convert` command; this chapter covers the others: `compare`, `composite` (previously known as `combine`), `conjure`, `identify`, `import`, `mogrify`, `montage`, `animate`, and `display`. The rationale behind the focus on the `convert` command is that most of the functionality offered by these commands in this chapter can also be accessed via `convert`.

Chapter 5, "Performing Artistic Transformations": Chapter 5 is my chance to show off the more artistic transformations that ImageMagick can apply; these include blurring images, adding charcoal effects, imploding images, adding noise to images, making an image look like it was painted by hand, adding beveled edges, creating shadows, spreading pixels randomly, and so forth.

Chapter 6, "Performing Other Image Transformations": Finally for the command-line image transformations, there is Chapter 6. This chapter mops up all the command-line operations that haven't been demonstrated in earlier chapters, apart from those used to draw or annotate images (which are covered in the next chapter). These operations are the more routine of those offered by ImageMagick, such as adding borders, rotating images, manipulating contrast in the image, dithering an image, and so on.

Chapter 7, "Using the Drawing Commands": Chapter 7 is the last chapter that documents the `convert` command. In this chapter, I discuss how to create and annotate images using the drawing commands that ImageMagick implements. Also, I discuss how to specify colors and then walk you through each of the drawing and annotation commands available.

Chapter 8, "PerlMagick: ImageMagick Programming with Perl": Chapter 8 is the first of the programming chapters, and it covers a Web photo management system written in Perl using the PerlMagick ImageMagick interface.

Chapter 9, "Implementing Your Own Delegate with C": ImageMagick implements support for new image formats with delegates. This chapter demonstrates how to write a simple delegate to support your own image format using the C programming language.

Chapter 10, "RMagick: ImageMagick Programming with Ruby": Chapter 10 demonstrates a simple command-line interface to build batch conversion jobs written in Ruby. The code allows you to interactively apply ImageMagick operations to an image and then apply all the operations you used on that image to all the images in a specified directory with a specified filename filter.

Chapter 11, "MagickWand: ImageMagick Programming with PHP": Chapter 11 demonstrates a PHP implementation of an on-the-fly graph-generation page using ImageMagick. The graphs use image composition to provide nice-looking output.

Chapter 12, "Where to Go from Here": The final chapter of the book covers those final little issues that are always handy to know, such as where to find information about topics not covered in this book, how to join the ImageMagick community, and how to report bugs.

Prerequisites

This book discusses ImageMagick 6.2.3. The concepts discussed are applicable to future and previous releases, however. Further, the book's content is relevant regardless of the platform on which ImageMagick is installed.

You can download ImageMagick from its Web site at `http://www.imagemagick.org`.

Contacting the Author

You can e-mail Michael Still at `imagemagick@stillhq.com`, and you can find his Web site at `http://www.stillhq.com`. You can find the ImageMagick blog for the book at `http://www.stillhq.com/imagemagick/`, and you can find the Apress page for the book at `http://www.apress.com/book/bookDisplay.html?bID=10052`.

For all the examples in this book, the figures are available online at `http://www.apress.com` in the Source Code section and at `http://www.stillhq.com/imagemagick/book/`. The online figures are full-color images, so you can download them if you need to further understand an example.

CHAPTER 1

Installing and Configuring ImageMagick

This chapter will give you detailed instructions on how to obtain, install, and configure ImageMagick. It also will discuss the architectural design of ImageMagick and explain how you can use that architecture to expand and customize ImageMagick. Finally, it will discuss how to get online help and debug problems you might have with ImageMagick.

If you already have ImageMagick installed on your machine, then you can skip the "Installing Precompiled Versions" and "Installing from Source" sections of this chapter.

Installing Precompiled Versions

By far the quickest and easiest way to install ImageMagick is to install the precompiled binary version, which is probably packaged by either your operating system provider or the ImageMagick team. Too many Linux distributions exist to cover all of them here, so I have limited this discussion to the two main packaging formats—apt and RPM.

If your chosen operating system isn't covered in this chapter, then fear not—you have two options for installing ImageMagick. First, it's quite possible that your operating system provider has packaged ImageMagick, so you should check in the normal place for your operating system. Second, failing that, you can refer to the "Installing from Source" section of this chapter to install ImageMagick from source. To do this, you'll need a compiler installed on your machine, though.

Debian and Ubuntu Linux

On both Debian and Ubuntu Linux, the name of the package to install is `imagemagick`. I personally run Debian Unstable on my laptop, and this installed, until recently, the same version of ImageMagick that Ubuntu 5.04 (Hoary Hedgehog) installs, which is 6.0.6. This is quite old compared with the latest version of ImageMagick at the time of writing of this book, which is 6.2.3.

Debian Unstable has now upgraded to the latest upline ImageMagick, so you should see that new version flow through to the next release of Ubuntu as well.

Red Hat Linux

Fedora Core 3 has packaged ImageMagick as well. The name of the package to install is `ImageMagick`, and when I did a default install, it was already installed on the system, which was nice. The version of ImageMagick currently packaged with Fedora Core 3 is 6.0.7, which is a little out of date.

Older ImageMagick Versions

As discussed in the previous sections, several of the more common Linux distributions currently install older versions of ImageMagick by default. You can cope with this problem in a couple of ways. The first option is that you could of course just download the source for ImageMagick and compile and install it yourself. You'll find instructions on how to do that later in the "Installing from Source" section if you're interested. Another option is to find someone else who has compiled the latest version and has already packaged it for your chosen distribution. Instructing you on how to do this, however, is outside the scope of this book. Finally, you should find that most of the features discussed in this book also work with the older versions of ImageMagick that are still shipping.

Microsoft Windows

Installing ImageMagick on Microsoft Windows machines is fairly trivial. The first step is to download the installer from `http://www.imagemagick.org`. You'll find a link to the download page on the left side of the home page. Download the installer, and run it. Figure 1-1 shows the first screen you'll see.

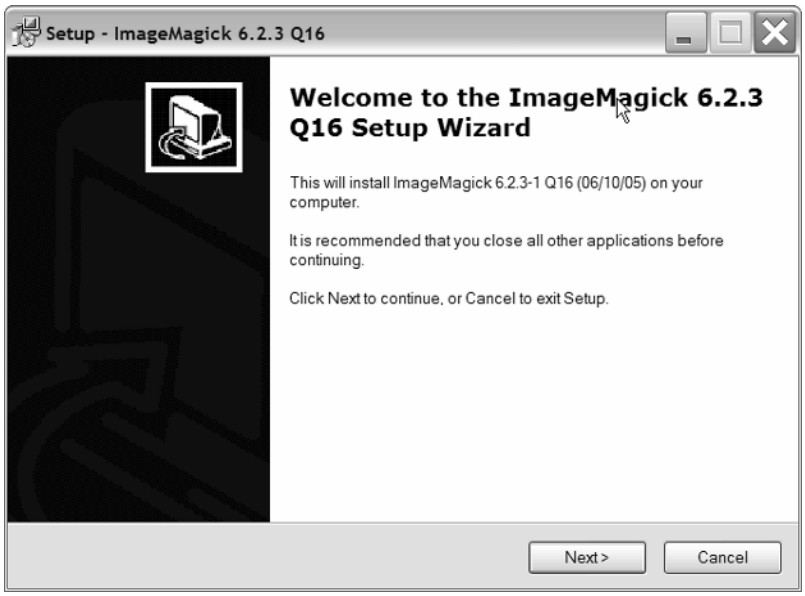

Figure 1-1. *Viewing the welcome screen for the installer*

The installer recommends that you close all other applications that are running on the machine before proceeding, which is a fairly common requirement. I recommend you do indeed do this, which will minimize the risk of ending up in an inconsistent state with the dynamic link libraries (DLLs) on your system. Click Next. You're now presented with a screen that asks you to agree with the license agreement for ImageMagick, which is something you'll need to do for any of the versions discussed in this chapter. Figure 1-2 shows a sample of what this screen looks like.

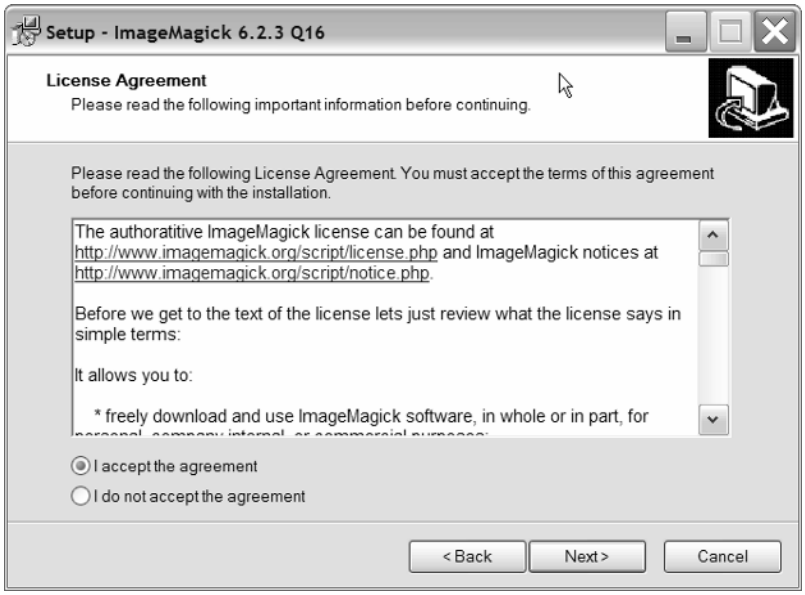

Figure 1-2. *Reading the ImageMagick license agreement*

If you do agree with the license agreement, then select Accept the Agreement, and then click Next. If you don't accept the agreement, then ImageMagick will not be installed. Next, you'll see a screen explaining some of the administrative requirements of the installation. If you've previously installed a version of ImageMagick and are attempting to upgrade instead of having two versions side by side on the machine, then you'll need to uninstall that version before proceeding with the installer, as shown in Figure 1-3. Additionally, if you want to install ImageMagick so that any user on the machine can use it, then you'll need to run the installer from the Administrator account. Remember, however, that older versions of Microsoft Windows don't necessarily have the concept of an Administrator account, which means that all users of the machine will get ImageMagick by default.

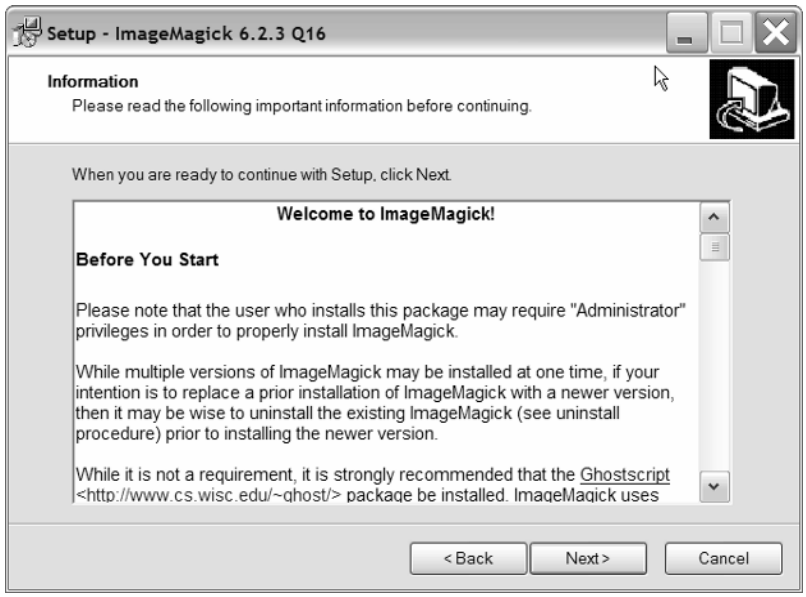

Figure 1-3. *Getting some reminders about the install process*

After clicking Next, you'll be asked where to install ImageMagick on your disk, as shown in Figure 1-4. I like selecting the default here so that all my applications are together in the `Program Files` directory, but if you're low on disk space on one partition, then you can install ImageMagick to another partition.

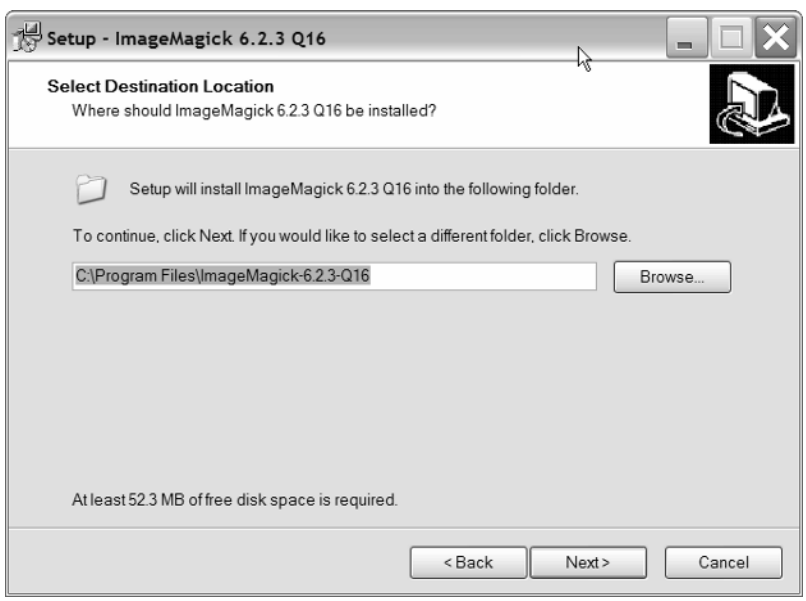

Figure 1-4. *Selecting the directory in which to install ImageMagick*

After clicking Next, you'll be asked for the name of the folder in the Start ➤ Programs menu for Windows. The default name is pretty sensible, but you can change it if you want, as shown in Figure 1-5.

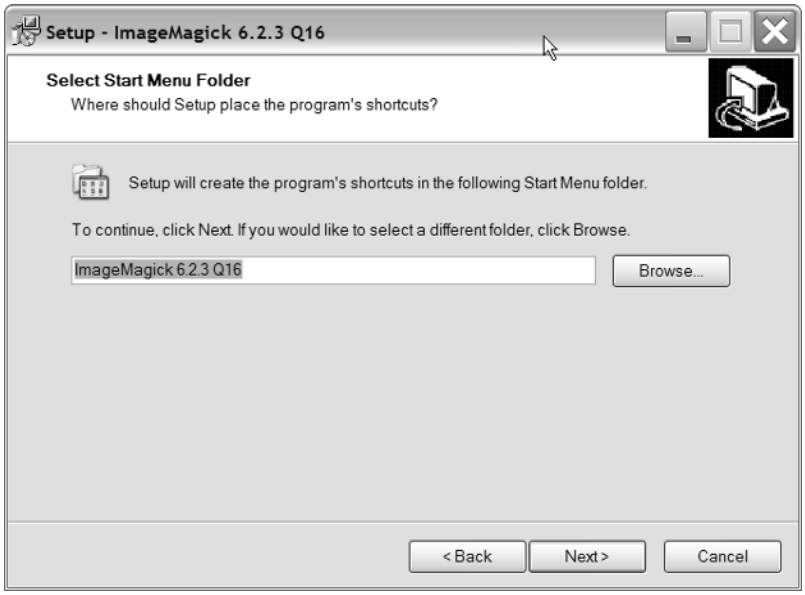

Figure 1-5. *Selecting a name for the entry in the Programs menu*

After you click Next, the subsequent screen asks questions about the rest of the install process. Note that if you want to use ImageMagick from the command line, as discussed in most of this book, then you're best off updating the executable search path so that the Windows command shell can find the ImageMagick executables. I've also chosen to use ImageMagick as my viewer, so I associated the file extensions, which isn't the default, as shown in Figure 1-6.

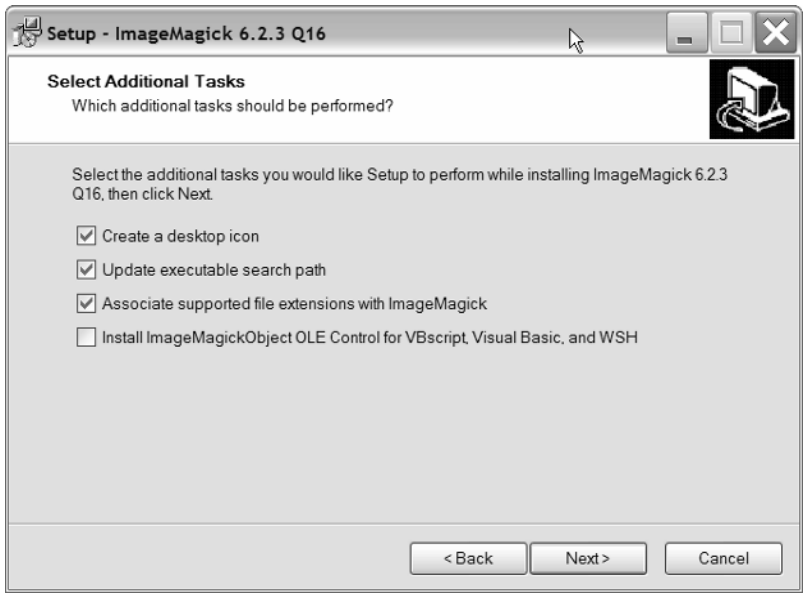

Figure 1-6. *Selecting installation options*

ImageMagick's installer now has enough information to proceed. After you click Next, the installer will show you the final configuration screen, which confirms the installation settings, as shown in Figure 1-7.

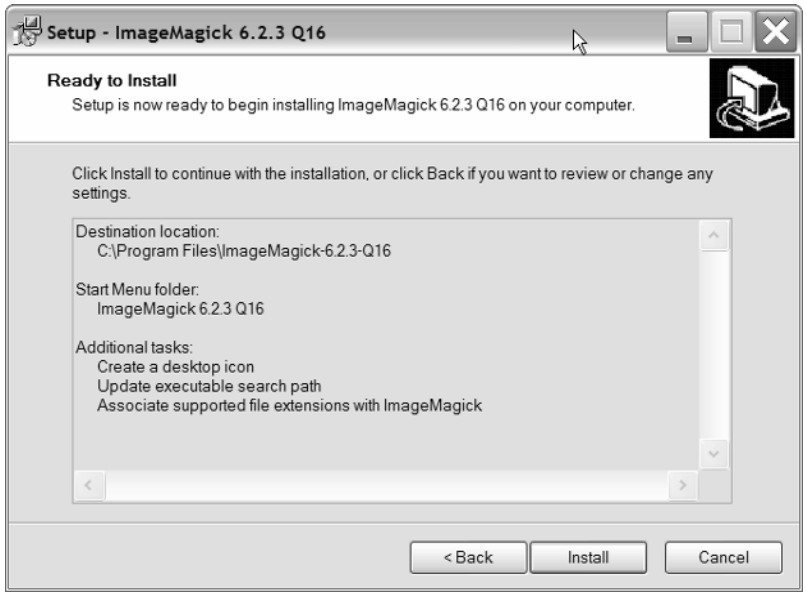

Figure 1-7. *Confirming installation settings*

Click Install to start the installation. You'll see a progress bar, as shown in Figure 1-8, even though the install doesn't take long (at least on my machine).

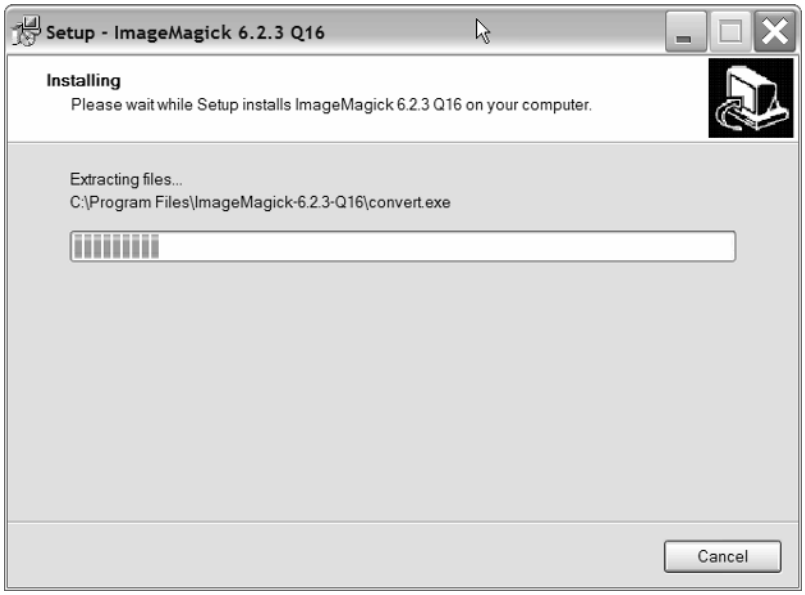

Figure 1-8. *Watching the installation progress*

The installer now provides some advice about how to make sure your installation worked, as shown in Figure 1-9. I recommend you follow these instructions, because if the installation has failed, then you'll be confused when you try to work along with the examples in the book and they don't work.

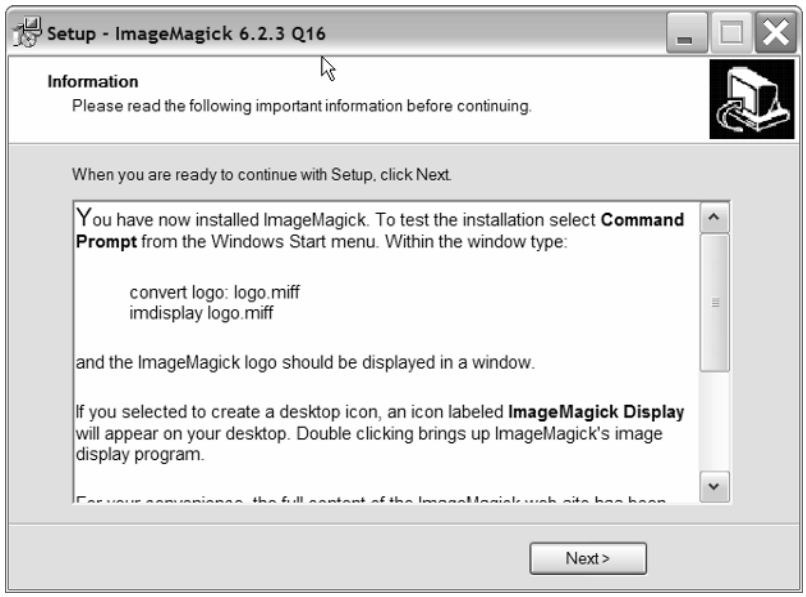

Figure 1-9. *Reviewing information about how to test the ImageMagick installation*

The final screen in the installer offers to take you to the ImageMagick documentation, as shown in Figure 1-10. Uncheck the box if you don't want the documentation to open in your default browser.

Figure 1-10. *Do you want to see some documentation?*

You've now installed ImageMagick for Microsoft Windows and tested the installation, so you're set to go.

Installing from Source

If a packaged version of ImageMagick for your operating system doesn't exist, or if you want more control over configuring and installing ImageMagick, then you might consider installing the software from source. The following sections of the chapter outline how to do this, but it's important to remember that I will assume that you already have a compiler installed and working on your machine. Depending on the operating system, this might mean you need to purchase compiler software from your vendor or install an open source alternative.

Introducing Dependencies

You'll need to install a number of dependencies in addition to ImageMagick in order to have a fully functional ImageMagick installation. It's important that these dependencies are installed before you start configuring and compiling ImageMagick, because the `configure` script for ImageMagick will disable functionality that isn't available because of missing dependencies at compile time.

In other words, if `libpng` (which is needed for supporting the PNG image format) were missing at the time that you ran the `configure` script, then this functionality would be missing from your ImageMagick installation. This is true even if you installed `libpng` after compiling ImageMagick. In that case, you'd need to reconfigure and recompile ImageMagick for the new functionality to become available.

Several classes of dependencies exist, each of which is discussed in turn in the following sections.

Introducing Delegates

For some of its work, ImageMagick uses command-line tools called *delegates* to encode and decode the image file in a format that ImageMagick can use. That intermediate format can then be further processed before being saved into the format that you want. This means the ImageMagick team can implement significantly fewer file format conversion routines without losing any functionality. You can see an example of the delegate detection process in the section "Compiling on Unix Operating Systems." You can also add your own delegates to the mix by using the delegate configuration file, which is discussed in the "Using Configuration Files" section. Chapter 9 also contains an example of a custom delegate.

For now, I'll stick to listing the delegates that ImageMagick supports so that you know what you might want to install before you compile ImageMagick from source (see Table 1-1).

Table 1-1. *Delegates Used by ImageMagick*

Delegate Name	Used For	URL to Download the Delegate From
bzlib	Bzip compression in MIFF files	`http://sources.redhat.com/bzip2/`
DPS	Display PostScript, which is used only for Postscript files if Ghostscript is unavailable	

(Continued)

Table 1-1. *(Continued)*

Delegate Name	Used For	URL to Download the Delegate From
FlashPIX	FlashPIX format	`ftp://ftp.imagemagick.org/pub/ImageMagick/delegates/libfpx-1.2.0.9.tar.gz`
FreeType	TrueType fonts	`http://www.freetype.org`
GhostPCL	PCL page description language	`http://www.artifex.com/downloads/`
Ghostscript	PostScript and PDF document formats	`http://www.cs.wisc.edu/~ghost/`
Graphviz	Graphviz visualization	`http://www.graphviz.org`
JBIG	JBIG lossless, black-and-white compression format	`http://www.cl.cam.ac.uk/~xml25/jbigkit/`
JPEG	JPEG files	`ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz`
JPEG 2000	JPEG 2000 files (the next version of the JPEG compression standard)	`http://www.ece.uvic.ca/~mdadams/jasper/`
LCMS	ICC CMS color management	`http://www.littlecms.com/`
PNG	Support for the PNG image format	`http://www.libpng.org/pub/png/pngcode.html`
TIFF	Support for the TIFF image format	`http://www.libtiff.org`
WMF	Support for Windows metafiles	`http://sourceforge.net/projects/wvware/`
zlib	Support for deflate compression	`http://www.gzip.org/zlib/`

Each of these delegates is open source and can be separately downloaded and installed before ImageMagick is configured if you need the facilities it implements. Details for how to install each of these dependencies is outside the scope of this chapter, but each of these delegates comes with excellent documentation about how to perform the installation steps needed.

Compiling on Unix Operating Systems

The following instructions apply to Linux, the various BSDs (including FreeBSD, OpenBSD, and NetBSD), Solaris, Mac OS X, AIX, and many other Unix variants. ImageMagick is identical to most other open source projects in its installation methodology. For those of you who haven't done this before, don't worry, because I'll walk you through the process.

The first step is to download the source code from the ImageMagick Web site at `http://www.imagemagick.org`. On the current site, the download link is on the left side of the screen and leads you to a page where you can download the source code.

Once you have the source code, you'll need to uncompress it. As I mentioned earlier, the current version of ImageMagick at the time of writing this book is 6.2.3, so that's what I'll use in these examples. Anyway, here's how to decompress the source code:

```
tar -xvzf ImageMagick.tar.gz
```

You should see output like this:

```
ImageMagick-6.2.3/
ImageMagick-6.2.3/Install-mac.txt
ImageMagick-6.2.3/depcomp
ImageMagick-6.2.3/ImageMagick.spec.in
ImageMagick-6.2.3/PerlMagick/
ImageMagick-6.2.3/PerlMagick/Makefile.PL
ImageMagick-6.2.3/PerlMagick/Makefile.nt
ImageMagick-6.2.3/PerlMagick/.gdbinit
ImageMagick-6.2.3/PerlMagick/Makefile.PL.in
ImageMagick-6.2.3/PerlMagick/Makefile.am
ImageMagick-6.2.3/PerlMagick/demo/
ImageMagick-6.2.3/PerlMagick/demo/Turtle.pm
ImageMagick-6.2.3/PerlMagick/demo/lsys.pl
ImageMagick-6.2.3/PerlMagick/demo/demo.pl
ImageMagick-6.2.3/PerlMagick/demo/tree.pl
ImageMagick-6.2.3/PerlMagick/demo/shapes.pl
ImageMagick-6.2.3/PerlMagick/demo/yellow_flower.gif
ImageMagick-6.2.3/PerlMagick/demo/Generic.ttf
ImageMagick-6.2.3/PerlMagick/demo/composite.pl
ImageMagick-6.2.3/PerlMagick/demo/red_flower.gif
ImageMagick-6.2.3/PerlMagick/demo/steganography.pl
ImageMagick-6.2.3/PerlMagick/demo/smile.gif
ImageMagick-6.2.3/PerlMagick/demo/shadow_text.pl
ImageMagick-6.2.3/PerlMagick/demo/annotate.pl
ImageMagick-6.2.3/PerlMagick/demo/src.png
ImageMagick-6.2.3/PerlMagick/demo/Makefile
...
```

The output shown here from that command is an example of what you'll see. I've truncated the listing here because it would fill several pages and not be particularly interesting to read.

■ **Note** You can find out more about the `tar` command, and the arguments it takes, by reading the `tar` man page. If manual pages have been installed on your system, you can access the man page by typing `man tar`. If manual pages aren't installed, then you'll find many examples of them online.

Now that you've extracted the source code, change directories into the new source code directory that `tar` extracted for you, and configure the code, like so:

```
cd ImageMagick-6.2.3
./configure
```

The name of the directory will change if you've extracted a different version of ImageMagick. The output from the `tar` command will tell you the directory name, however. You'll see output like this:

```
configuring ImageMagick 6.2.3
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking whether build environment is sane... yes
checking for a BSD-compatible install... /usr/bin/install -c
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for a sed that does not truncate output... /bin/sed
checking for egrep... grep -E
checking for ld used by gcc... /usr/bin/ld
checking if the linker (/usr/bin/ld) is GNU ld... yes
checking whether gcc and cc understand -c and -o together... yes
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets $(MAKE)... (cached) yes
checking maximum warning verbosity option...  -Wall for C
checking whether ln -s works... yes
checking for gcc... (cached) gcc
checking whether we are using the GNU C compiler... (cached) yes
checking whether gcc accepts -g... (cached) yes
checking for gcc option to accept ANSI C... (cached) none needed
checking dependency style of gcc... (cached) gcc3
checking if malloc debugging is wanted... no
...
```

Again, I've truncated the output so as to not fill the entire book with command output. I'll show the last few lines from the output, though, because they're important:

```
ImageMagick is configured as follows. Please verify that this configuration
matches your expectations.

Host system type : i686-pc-linux-gnu
```