THE EXPERT'S VOICE®

Digital Image Compositing Fundamentals

Wallace Jackson

Apress®

Digital Image Compositing Fundamentals

Wallace Jackson

Apress®

Digital Image Compositing Fundamentals

Copyright © 2015 by Wallace Jackson

ISBN-13 (pbk): 978-1-4842-1639-2

ISBN-13 (electronic): 978-1-4842-4060-1

Digital Image Compositing Fundamentals is dedicated to everyone in the open source community who is working so diligently to make professional new media application development software and content development tools freely available to rich application developers so that they can utilize them to achieve our creative dreams and financial goals. Last but not least, I dedicate this book to my father, Parker Jackson, my family, my life-long friends, and my production ranch neighbors for their constant help, assistance, and those relaxing, late-night BBQs!

Contents at a Glance

About the Author xiii

About the Technical Reviewer xv

Acknowledgments xvii

Introduction xix

■ Chapter 1: The Foundation of Digital Imaging: The Pixel 1

■ Chapter 2: The Size of Digital Imaging: Resolution 7

■ Chapter 3: The Shape of Digital Imaging: Aspect Ratio 15

■ Chapter 4: The Color of Digital Imaging: Color Theory 23

■ Chapter 5: The Digitization of Digital Imaging: Color Depth 31

■ Chapter 6: The Transparency of Digital Imaging: Alpha Channel 39

■ Chapter 7: The Isolation of Digital Imaging: Masking Tools 49

■ Chapter 8: The Organization of Digital Imaging: Using Layers 65

■ Chapter 9: The Algorithms of Digital Imaging: Blending Modes 79

■ Chapter 10: The Modality of Digital Imaging: Modal Operation 95

■ Chapter 11: The Automation of Digital Imaging: Plug-In Filters 111

■ Chapter 12: The Work Process of Digital Imaging: Compositing 123

■ Chapter 13: The Data Footprint of Digital Images: Optimization 135

■ Chapter 14: The Automation of Digital Imaging: Programming 151

Index 163

Contents

About the Author xiii

About the Technical Reviewer xv

Acknowledgments xvii

Introduction xix

■Chapter 1: The Foundation of Digital Imaging: The Pixel................ 1

Downloading and Installing GIMP.. 1

A Foundation for Digital Images: The Pixel.. 5

Raster vs. Vector: Imaging vs. Illustration 6

Rendering: Turning Convert Vector Art into Raster Imagery 6

Summary.. 6

■Chapter 2: The Size of Digital Imaging: Resolution 7

Resolution: The Number of Pixels in 2D 7

Doing the Math: Calculating Total Image Pixels...................................... 8

Matching Resolution to Target: Device Resolutions................................ 8

Using Imaging Software: Determining Resolution 9

Summary.. 12

Chapter 3: The Shape of Digital Imaging: Aspect Ratio 15
Aspect Ratio: The Ratio of Pixels in 2D 15
Screen Shapes: Common Display Aspect Ratios 16
Doing the Math: How to Arrive at the Aspect Ratio 16
Using Imaging Software: Maintaining Aspect Ratio 17
Summary 21
Chapter 4: The Color of Digital Imaging: Color Theory 23
Color Theory: Using Pixel Color Channels 23
Subtractive Color: Print Using CMYK Color Plates 24
Additive Color: Display Using RGB Color Channels 26
Summary 29
Chapter 5: The Digitization of Digital Imaging: Color Depth 31
Color Depth: Bit Levels that Define Color 31
Indexed Color: Using Palettes to Hold 256 Colors 32
High Color: Using a 15-Bit and 16-Bit Color Depth 34
True Color: Using a 24-Bit Color Image Backplate 35
True Color Plus Alpha: Using 32-Bit Digital Images 36
Summary 37
Chapter 6: The Transparency of Digital Imaging: Alpha Channel....... 39
Alpha Channels: Defining Transparency 39
Selection: Using Alpha for Selection Sets 43
Selecting Alpha Channel Data in Photoshop CS 43
Selecting Alpha Channel Data in GIMP 45
Summary 48

■ **Chapter 7: The Isolation of Digital Imaging: Masking Tools 49**

Masking: Isolating and Extracting Pixels 49

Photoshop Magic Wand: Tolerance and Sampling 50

Adding to a Selection Set: Using the Shift Modifier 52

Selection Editing: Using an Alpha to Edit Selection 53

Mask Fine-Tuning: Alpha Edge Editing Techniques 55

GIMP Scissors Tool: Intelligent Masking Algorithm 58

Adjusting the Scissors Tool: Intelligent Spline Nodes 59

Summary 63

■ **Chapter 8: The Organization of Digital Imaging: Using Layers 65**

Layer Differentiation: Photoshop vs. GIMP 66

Photoshop Layers: Organization and Effects 66

GIMP Layers: Organization and Masking 67

Using Layers in Photoshop: Compositing 68

Adding a Text Object: Using the Vertical Text Tool 69

Special Effects Layers: Adding Effects to the Text 70

Adjustment Layer: Adjusting Image Characteristics 71

Using Layers in GIMP: Basic Compositing 73

Summary 78

■ **Chapter 9: The Algorithms of Digital Imaging: Blending Modes 79**

Pixel Blending: Porter Duff Modes Theory 79

Photoshop Blending: Blending Modes Supported 81

GIMP Blending: Blending Modes Supported 81

Using Modes in GIMP: Multiplying a Shadow 82

Photoshop Blending Modes: Cool Effects 86

Soft Light: Shining a Light on Your Digital Imagery 87

Hard Light: Adding a Glass Overlay Effect to Text 88

Linear Burn: Turning a 3D Metal Hoop into Glass 88
Luminosity: Turning Your Metal Hoop into Chrome 90
Difference: Using a Layer to Invert Pixel Colors 90
Summary 92

Chapter 10: The Modality of Digital Imaging: Modal Operation 95

Modal Operation: Part of the Pipeline 95
Processing Pipeline: Tool, Options, and Settings 96
Practice Makes Perfect: Keeping It All in Your Mind 96
Modal Operation: A Photoshop Example 96
Photoshop Magic Wand Tool: Modal Selection Sets 97
Photoshop Layer Mask: Delete Default Layer Mask 98
Custom Layer Mask: Add Selection as Layer Mask 98
Modal Operation: The GIMP Example 100
GIMP Layer Mask: Adding a Layer Mask 102
GIMP Special Effects: Creating a Ring Drop Shadow 103
Summary 109

Chapter 11: The Automation of Digital Imaging: Plug-In Filters 111

GIMP Plug-In Filters: Enhancing GIMP 2.8 111
GIMP's Filters Menu: Using Plug-In Filters 113
Photoshop's Filter Menu: Using Plug-In Filters 116
Summary 121

Chapter 12: The Work Process of Digital Imaging: Compositing 123

GIMP Compositing: Using Layer Features 123
Photoshop Compositing: Using Layers 128
Summary 133

■Chapter 13: The Data Footprint of Digital Images: Optimization......135
Photoshop Optimization: Save for Web.. 135
Indexed Color Format: Using Less Color to Reduce File Size...............................137
Lossy Compression: Using JPEG to Reduce File Size 139
Layer Visibility: Selecting Which Pixels Will Be Saved..141
GIMP Optimization: File ➤ Export As... 144
Creating a Masked PNG-32 Object in GIMP.. 147
Summary... 150
■Chapter 14: The Automation of Digital Imaging: Programming.......151
Internal Scripting Languages: Automation 151
Photoshop Scripting: JavaScript and ExtendScript 152
GIMP Scripting: Scheme, Script-Fu, and Python...153
Java and JavaFX: javafx.scene.effect API... 155
HTML5 and CSS3: Markup Compositing.. 157
Android Studio: The PorterDuff Class 159
Summary... 161
Index.. 163

About the Author

Wallace Jackson has been writing for several leading multimedia publications about work in the new media content development industry, after contributing a piece about advanced computer processing architectures for the centerfold (a removable "miniissue" insert) of an original issue of *AV Video Multimedia Producer* magazine that was distributed at the SIGGRAPH trade show. Wallace has written for a large number of popular publications about his work in interactive-3D and new-media-advertising campaign design, including: *3DArtist* magazine, *Desktop Publisher Journal*, *CrossMedia* magazine, *Kiosk* magazine, *AV Video Multimedia Producer* magazine, *Digital Signage* magazine, and many other publications.

Wallace has authored a dozen Apress book titles, including four titles in its popular Pro Android series, Java and JavaFX game development titles, digital image compositing titles, and new media content production titles.

In the current book on digital image compositing, he focuses on the GIMP and Photoshop CS6 digital image compositing software packages, and uses them to demonstrate digital image editing and compositing fundamentals to beginners who wish to become digital imaging professionals.

Wallace is currently the CEO of MindTaffy Design, an agency specializing in new media content production and digital campaign design and development, located in Northern Santa Barbara County, halfway between its clientele in Silicon Valley to the north and Hollywood, the "OC," West LA, and San Diego to the south.

MindTaffy Design has created open source, technology-based (HTML5, JavaScript, Java, JavaFX, and Android 5.3) digital new media i3D content deliverables for more than a quarter century (since 1991).

The company's clients consist of a significant number of international branded manufacturers, including Sony, Tyco, Samsung, IBM, Dell, Epson, Nokia, TEAC, Sun Microsystems, Micron, SGI, KDS USA, EIZO, CTX International, KFC, Nanao USA, Techmedia, EZC, and Mitsubishi.

Wallace received his undergraduate BA degree in business economics from the University of California at Los Angeles (UCLA) and his graduate degree in MIS business information systems design and implementation from University of Southern California in Los Angeles (USC). Wallace also received a postgraduate degree in marketing strategy from USC and completed the USC Graduate Entrepreneurship Program. He earned the two USC degrees while at USC's nighttime Marshall School of Business MBA Program, which allowed him to work full time as a COBOL programmer while completing his degrees.

About the Technical Reviewer

Chád ("Shod") Darby is an author, instructor, and speaker in the Java development world. As a recognized authority on Java applications and architectures, he has presented technical sessions at software development conferences worldwide (in the United States, UK, India, Russia, and Australia). In his fifteen years as a professional software architect, he's had the opportunity to work for Blue Cross/Blue Shield, Merck, Boeing, Red Hat, and a handful of start-up companies.

Chád is a contributing author to several Java books, including *Professional Java E-Commerce* (Wrox Press), *Beginning Java Networking* (Wrox Press), and *XML and Web Services Unleashed* (Sams Publishing). Chád has Java certifications from Sun Microsystems and IBM. He holds a BS in computer science from Carnegie Mellon University.

You can visit Chád's blog at `www.luv2code.com` to view his free video tutorials on Java. You can also follow him on Twitter at `@darbyluvs2code`.

Acknowledgments

I would like to acknowledge all my fantastic editors and their support staff at Apress, who worked those long hours and toiled so very hard on this book to make it the ultimate digital image compositing fundamentals book title currently on the market.

I thank:

Steve Anglin for his work as the Acquisitions Editor for the book and for recruiting me to write development titles at Apress covering widely popular open source content development platforms (Android, Java, JavaFX, HTML5, CSS3, JS, GIMP, etc.).

Matthew Moodie for his work as the Development Editor on the book and for his experience and guidance during the process of making the book one of the leading image compositing titles.

Mark Powers for his work as the Coordinating Editor for the book and for his constant diligence in making sure that I either hit my chapter delivery deadlines or far surpassed them.

Jana Weinstein for her work as the Copy Editor on this book, for her careful attention to minute details, and for conforming the text to current Apress book writing standards.

Chád Darby for his work as the Technical Reviewer on the book and for making sure that I didn't make technical mistakes.

Anna Ishchenko for her work as a Book Cover Designer for the book and for creating the graphic design for the book.

Finally, I'd like to acknowledge Oracle for acquiring Sun Microsystems and continuing to enhance Java and JavaFX, which allows their Java and JavaFX to remain the premiere open source programming languages, and allows digital image compositing pipelines to be written in Java code, taking this industry to the next level.

Introduction

Digital Image Compositing Fundamentals is intended for digital photographers; multimedia producers; 2D-application developers; 2D or 3D web site developers; user interface or user experience designers; social media users of image-centric web sites and/or apps, including Instagram, Pinterest, Facebook, LinkedIn, XING, YouTube, and similar; and just about anyone interested in generating high-quality digital imagery or special effects delivered in popular PNG, JPEG, GIF and animGIF file formats.

The book covers digital image editing and compositing; in the early chapters, that equates to fundamentals: proper terms, topics, concepts, and definitions. The subsequent chapters each build on the knowledge of the previous chapter. So, by the time we get to the later chapters in the book, our readers should be creating some fairly advanced digital image compositing pipelines, by using alpha channels, masks, and blending modes, special effects, adjustment layers, and the like.

There is even coverage at the end of this book on data footprint optimization as well as on creating digital image compositing pipelines that use open source platforms such as Java, JavaFX, HTML5, CSS3, JavaScript, Python, and Android.

Chapter 1 focuses on one of the foundations of digital imaging, the pixel, and Chapter 2 looks at the size of digital imaging, or resolution. Chapter 3 covers aspect ratio and the shape of digital imaging, and Chapter 4 takes on digital imaging color theory.

As we get into Chapter 5, we take on the digitization of digital images using color depth, and Chapter 6 goes on to cover digital image transparency and the alpha channel. Chapter 7 looks at isolating areas in digital images using masking tools, and Chapter 8 addresses organizing the digital image composite by using layers.

Chapter 9 starts to get into more advanced concepts, like blending modes, and Chapter 10 discusses the modal operational nature of digital image compositing software packages, such as GIMP and Photoshop CS6. Chapter 11 covers the installation and usage of plug-ins, and Chapter 12 discusses the finer points of the digital image compositing pipeline using both Photoshop CS6 and GIMP 2.8.14.

In Chapter 13, we look at data footprint optimization and, finally, in Chapter 14 we review how computer programming languages factor into image compositing, both inside of GIMP and Photoshop and with popular open source platforms such as Python, Java, JavaFX, Android, JavaScript, CSS3, and HTML5.

If you're interested in digital photography or in digital imaging and want to learn the fundamentals, and how everything works in the digital domain from pixel to compositing pipeline, this is the digital image editing and special effects book for you. I even show how to do everything with free digital imaging software, GIMP 2.8.14, so your only expense will be this book.

Chock full of tips, tricks, tools, topics, terminology, techniques, and work processes, *Digital Image Compositing Fundamentals* can help you to transition from a digital imaging amateur to a knowledgable professional where your digital image compositing pipeline is concerned.

CHAPTER 1

The Foundation of Digital Imaging: The Pixel

Welcome to *Digital Image Compositing Fundamentals*. This book will take you through the foundation of digital imaging as well as multilayered image compositing. It starts with the lowest-level concepts—in this chapter that's the **pixel**—and builds upon each of those concepts in subsequent chapters until we have a comprehensive understanding of image compositing concepts, pipelines, work flows, and terminology.

I will show you what the concepts, techniques, and terms look like as we progress through this book by reproducing them, using two of the most widely used digital image compositing software packages, Adobe Photoshop CS6 and the open source GIMP software package; the latter is free for commercial use.

Assuming you might not want to pay for the brand name digital imaging software, the first part of this chapter will cover how to download and install your very own open source GIMP software package. If you do want to use paid software, I suggest using Photoshop at $10 per month. The rest of the chapter discusses the foundational element of digital image compositing: the pixel.

How the pixels are composited is what this book is all about, and each chapter will build on the knowledge from the chapters that preceded them until you have a clear picture (no pun intended) regarding precisely how digital image compositing pipelines work.

Downloading and Installing GIMP

In order to follow along with this book, all readers are going to need to have a digital imaging software package of one kind or another, whether that is Adobe Photoshop or Corel Painter or PaintShop Pro. If you do not own any of these, you can use GIMP 2.8, which is free for commercial use.

To download GIMP 2.8.14, which is the current stable version of GIMP (the next version, GIMP 2.9, the predecessor to the much anticipated GIMP 3.0, is due out some time during 2016), go to `www.gimp.org` and click the orange **Download** button, seen in Figure 1-1, or alternately click the **Downloads** link on the right side of the screen.

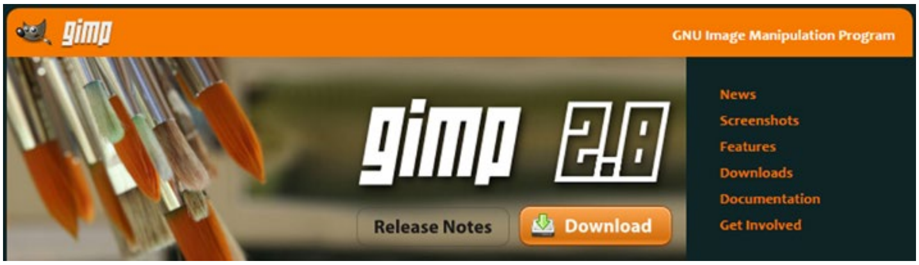
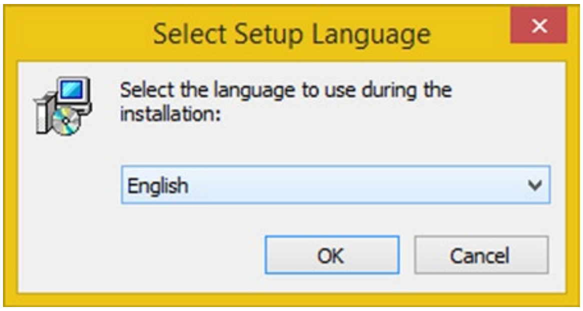


Figure 1-1. *Go to the gimp.org site; click the* ***Download*** *button*

Then, download the GIMP-2.8.14.exe installer for your OS to start the installation. The installer will determine whether you need a 32-bit or a 64-bit version, so all you have to do is select the language that you want to use for the installation of the software and click the **OK** button, as shown in Figure 1-2.


Select Setup Language

Select the language to use during the installation:

English

Figure 1-2. *Select a language you want to use for the install*

Once you click the **OK** button, you'll get a **GIMP Setup** dialog, as shown in Figure 1-3, where you can click the **Install** button to start the installation process.

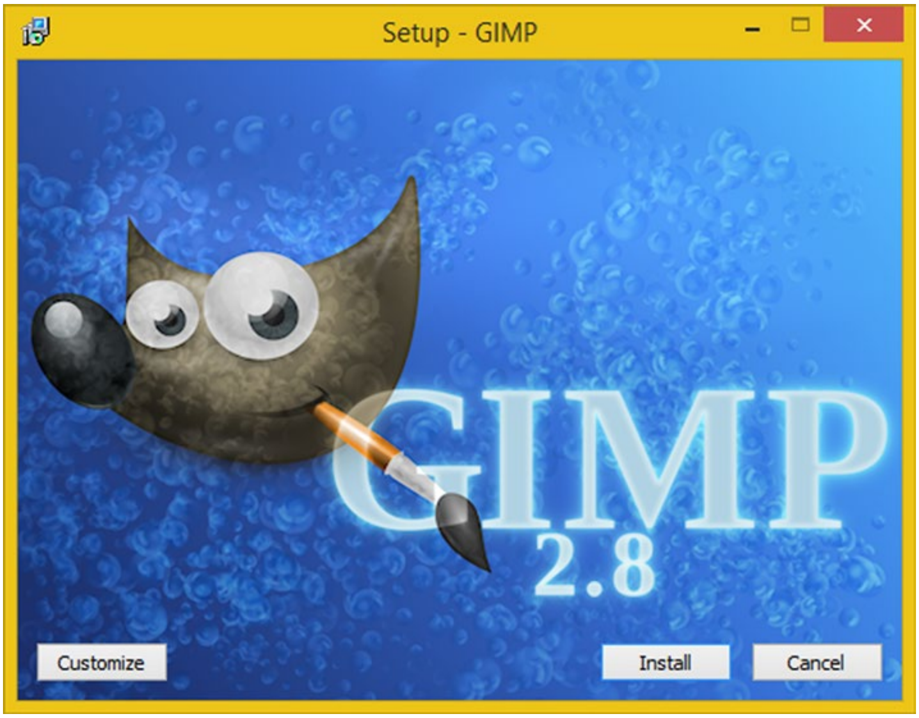


Figure 1-3. *Click the **Install** Button to start the install*

If you want to customize the installation, you can click the **Customize** button and select exactly what components you want installed on your system.

I recommend that you use the (full) Install button, which will give you the default GIMP software installation with the plug-ins, filters, and file support that is now part of the stable GIMP version, which is the fully completed (finished) package architecture. GIMP is an digital imaging-software package that rivals paid image editing software packages.

As you can see in Figure 1-4, GIMP will install itself.

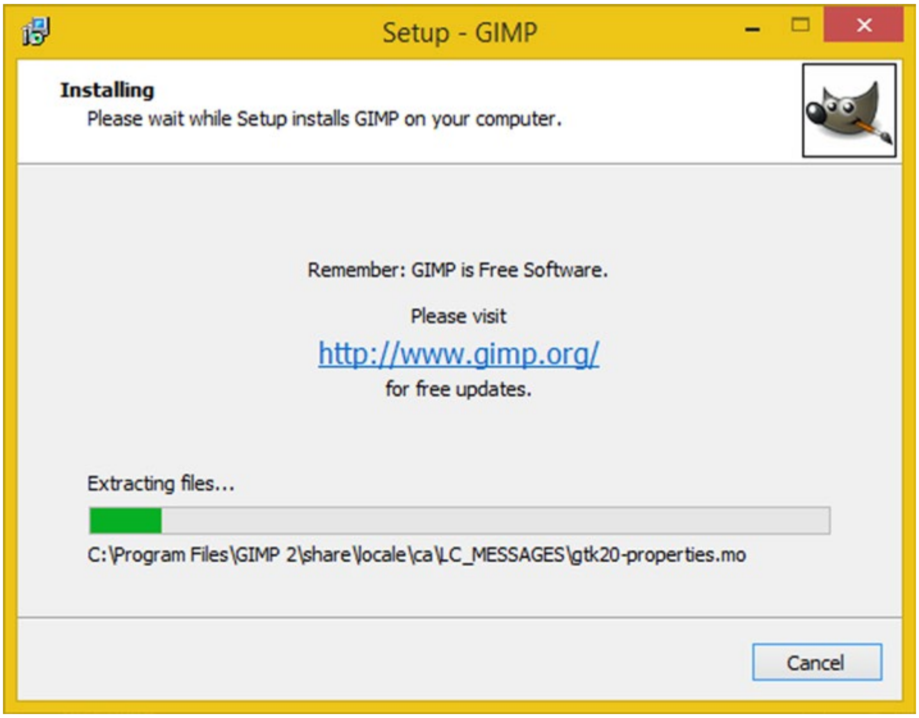
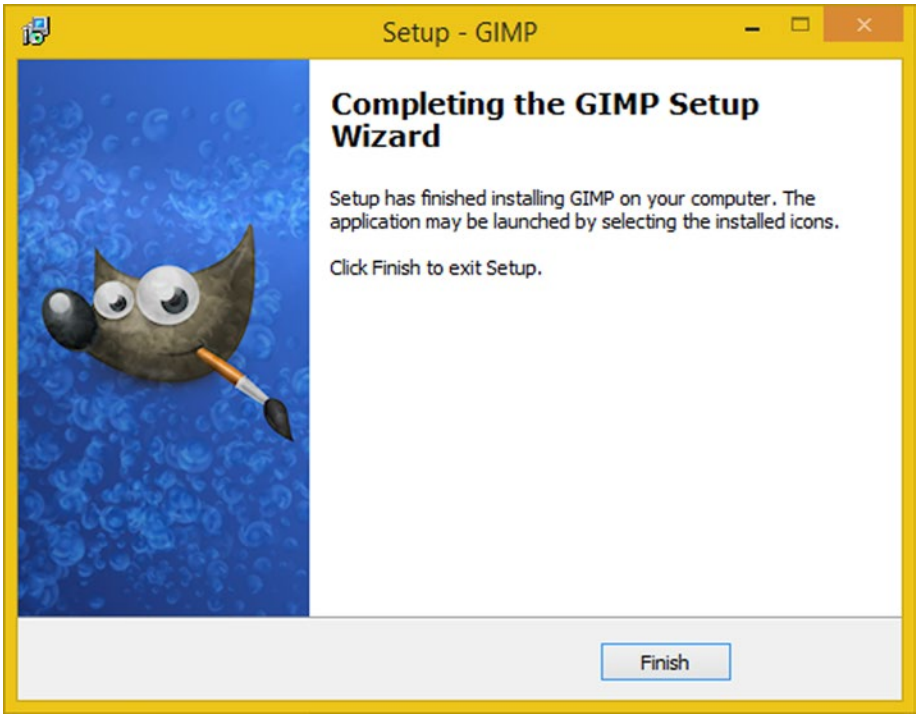

Setup - GIMP
Installing
Please wait while Setup installs GIMP on your computer.
Remember: GIMP is Free Software.
Please visit
http://www.gimp.org/
for free updates.
Extracting files...
C:\Program Files\GIMP 2\share\locale\ca\LC_MESSAGES\gtk20-properties.mo
Cancel

Figure 1-4. *GIMP tells you which file it's installing*

Once the install process has completed, click the **Finish** button, shown in Figure 1-5, and then create a shortcut icon for the quick launch taskbar on your OS so that you can launch GIMP if you need it by using just a single mouse click.



Completing the GIMP Setup Wizard

Setup has finished installing GIMP on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

Figure 1-5. *Click the* ***Finish*** *button to finish the install*

Now that you have installed a digital imaging software package, we can take a look at a foundational element for all digital image processing work flows, the pixel.

A Foundation for Digital Images: The Pixel

Digital images are made up of two-dimensional, or 2D, arrays (or grids) containing something called pixels. **Pixel**, an industry term, is a conjugation of two words: **pictures** (which are commonly called "pix") and **elements** (shortened to be just "els"). So the foundation for any digital image that you'll be working with is its picture elements.

These elements dictate everything about a digital image, including its file size, dimension, color, translucency, shape, format, and many other characteristics. I will devote a chapter to each characteristic over the course of the book as I build logical concept on top of logical concept. There are also **digital illustrations**, which are not made up of pixels.

Raster vs. Vector: Imaging vs. Illustration

Besides the term pixel, let's get into some other terminology related to pixels to make sure we cover everything thoroughly in this first chapter of the book. Pixel-based digital imagery is also called "raster imagery." This is because the array of pixels can be said to be "rasterized" by the device that is displaying these pixels. These devices include everything from an iTV to a tablet to a smart phone to a smart watch.

There is another type of digital image called a "vector" image, which is defined using **mathematics** rather than pixels. A vector image uses **points**, called "vertices," and **lines** or **curves** to draw out your "digital illustration," which is what a **vector image** is commonly referred to as in the multimedia industry today.

Vector imagery has its own genre of 2D software that is called digital illustration software. You may be familiar with Inkscape, an open source digital illustration software package, or Adobe Illustrator, or Macromedia Freehand, or Corel Draw.

Rendering: Turning Convert Vector Art into Raster Imagery

Vector imagery can be converted into raster imagery with a process called "rendering." A **rendered image** is inherently a raster image, and both 2D vector artwork and 3D vector artwork can be rendered into raster imagery, which, as you now know, is pixel-based imagery.

At this point, all of the concepts that are outlined in this book can be applied, or inherently would be applied, to this pixel-format raster imagery. Remember to keep a **back-up** of your vector artwork, so that you can enhance it and render it at any time, because once it is rasterized, or rendered, it is no longer vector artwork anymore, so keep your vector format.

The primary advantage of vector illustration over raster imagery is that since it's defined using math it can be **scaled** up or down to any size. Scaling raster imagery causes what is known as "pixelation."

Summary

In this first chapter, we made sure that you had a digital image compositing software package installed and ready to explore and also took a look at the foundational element of a digital image, the pixel. We also covered the distinction between raster, or pixel-based imagery, and vector, or mathematics-based imagery.

Next, we looked at the concept of rendering, or taking a vector math-based digital illustration and turning it into a raster pixel-based digital image representation.

In the next chapter, we will take a look at how pixels are stored in X,Y arrays and the concept of **image resolution**.

CHAPTER 2

The Size of Digital Imaging: Resolution

Now you have an understanding of the pixel building blocks that make up raster images as well as what the difference is between a raster image that is 2D and pixel based, and a vector image that is 2D (or 3D), and vertex and line (or curve) based. In this chapter, we build on these concepts by looking at the *x* and *y* dimensions of digital image compositing. Since you now have GIMP downloaded onto your computer, I will illustrate concepts in the book using both Photoshop and GIMP, as those are the most widely used digital image packages.

Whereas a single pixel is just **one point** in space, which is called **1D** or **one dimensional** (do not call your spouse this), an image is a **2D** array, or grid, of pixels that uses both an *x* and a *y* dimension.

There are many implications of what this means, which is why I dedicate the entire chapter to this **two-dimensional array** of pixels and what its implications include.

Having a 2D array involves having an **image size**, also called a **data footprint**, and a **resolution**, which defines how many pixels will be used to form the digital image, and will define how much **detail** your digital image will contain.

In the following sections, you'll be learning about the mathematics used to calculate the resolution for your images as well as the details about a wide range of the standard consumer electronics device display resolutions currently available in the industry, which you will be designing images for during Chapters 6 through 13.

Resolution: The Number of Pixels in 2D

The number of pixels contained in your digital image assets is expressed using a term called **resolution**. This term refers to the number of pixels contained in the image, incorporating both the **width**, which is usually denoted by using a *w*, or alternatively an *x* for the ***x* axis**, and the **height**, which is usually denoted using an *h*, or a *y* for the ***y* axis**. The resolution gives your digital imagery its 2D **dimensions**.

The resolution of your image asset is expressed using two numbers (the width, or *x*, and the height, or *y*) with an *x* in the middle. For instance, the common VGA resolution would be expressed as **640 x 480**. Alternately, you could also use the word "by," such as **640 by 480** pixels. Now let's get into some basic mathematics, so that you can see how to calculate how many pixels are contained in your image.

Doing the Math: Calculating Total Image Pixels

To find the total number of pixels that are contained in any 2D image, you will want to **multiply** the pixel width by the pixel height. Hopefully you remember the equation for the **area of a rectangle** from school; well here it is again, in a professional imaging example. Since I didn't realize that there would be professional applications for what was being taught in school, I didn't listen, and I had to go back and relearn math and physics.

As an example, by multiplying the width and the height of a true HD resolution, 1920 x 1080 image, you will get the result of **2,073,600** pixels. If you are a digital photographer, you will be familiar with the phrase **two megapixels**, referring to 2 million pixels, which is essentially what true High Definition, or "HD" resolution will give you in pixel count.

The more pixels that are used within a digital image, the higher its resolution is said to be. Higher resolution gives a viewer more **detail** or image subject matter **definition**, which is why an HDTV is called **high definition television**, and why these new 4K resolution UHDTVs are now called **ultra high definition television**.

Next, let's take a look at a few of the most popular consumer electronics device resolutions, so that you will know what pixel resolutions you should use for your digital image compositing pipelines, so your images will go full screen.

Matching Resolution to Target: Device Resolutions

One of the objectives in digital image compositing is to match the number of pixels in your digital image to the hardware device that it is going to be viewed on. There used to be dozens of devices with different resolutions in the marketplace, but recently devices have been conforming to HDTV resolutions, since the pixel pitch (dot size) has been getting smaller with new display technologies such as OLED (organic light-emitting diodes), allowing printer-like resolution on a display screen.

The first HDTV screens were what I like to call "**quasi-HD**" and were **1280 by 720** resolution; 1280 x 720 = **921,600** pixels. This 1280 resolution is still a common resolution and I see it used on a lot of entry-level smart phones or entry-level tablets, netbooks, and laptops. The reason for this is because a lot of film, video, and TV content continues to be produced in **Blu-ray** 1280 x 720 format, and matching content resolution to device resolution yields the best visual result, because **zero pixel scaling** is necessary. We will be covering pixel scaling in Chapter 3.

The next type of HDTV display screen available in the market today is what the industry calls **True HD**; it features a **1920 by 1080** resolution. As you know, this gives you 2 million pixels, in contrast to the 1 million pixels that you get with 1280 HD resolution. You see True HD resolution on entry-level iTV sets, mainstream smart phones, medium-size tablets, e-book readers, and high-end laptops (also called notebook computers).

A recent type of HDTV display screen that is available in the market today is what the industry calls **Ultra HD,** or **UHD**, and consists of a **4096 x 2160** resolution. If you do the math, this display screen resolution contains **8,847,360** pixels, or nearly nine megapixels, in contrast to the 2 million pixels that you get with 1920 HD resolution. You see Ultra HD resolution on high-level iTV sets, cutting-edge smart phones, high-end tablets, and e-book e-readers, as well as the latest UHD laptops (also called UHD notebook computers).

There's one other category of consumer electronic device with lower resolutions. Smart watches have resolutions of 320 by 320, and Huawei's smart watch is at 400 by 400. Look for higher pixel density (pixel pitch) in the smart watch space by 2016, affording 480-by-480 pixel screens, and, hopefully, 560-by-560 or 640-by-640 pixel screens by late 2016 or early 2017.

Using Imaging Software: Determining Resolution

To find your digital image resolution in Photoshop, you use the **Image ➤ Image Size** menu sequence, which is shown in Figure 2-1.

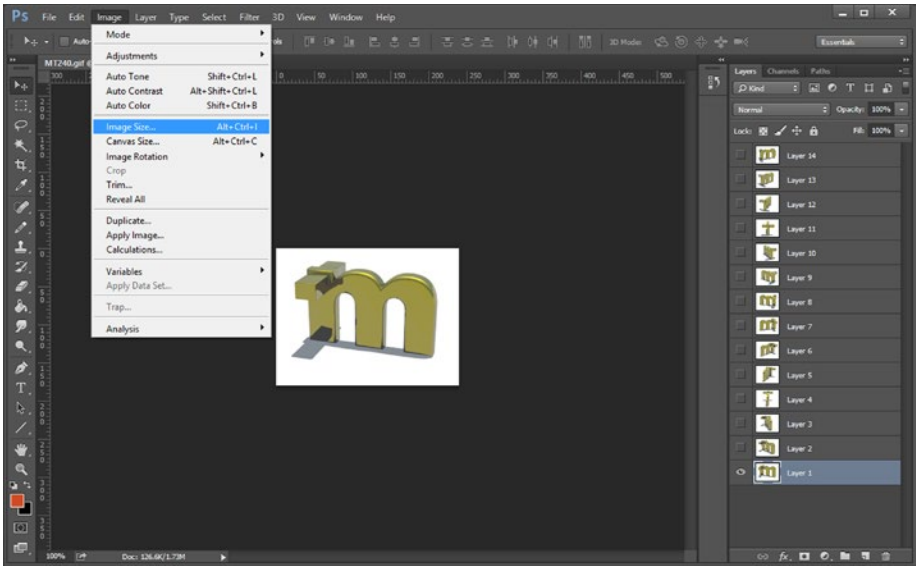


Figure 2-1. *In Photoshop use an Image ➤ Image Size menu sequence*

The GIMP has a different **Image Size dialog**, which is shown in Figure 2-2. This dialog will tell you the **image size** (the data footprint), **dimension**, **width**, **height**, **DPI** (pixels per inch) and gives you options for **resampling** (scaling), which we cover in Chapter 3.

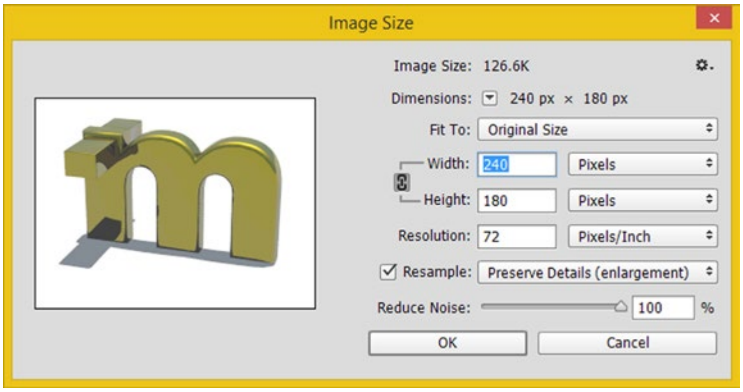


Image Size

Image Size: 126.6K

Dimensions: 240 px × 180 px

Fit To: Original Size

Width: 240 Pixels

Height: 180 Pixels

Resolution: 72 Pixels/Inch

[✓] Resample: Preserve Details (enlargement)

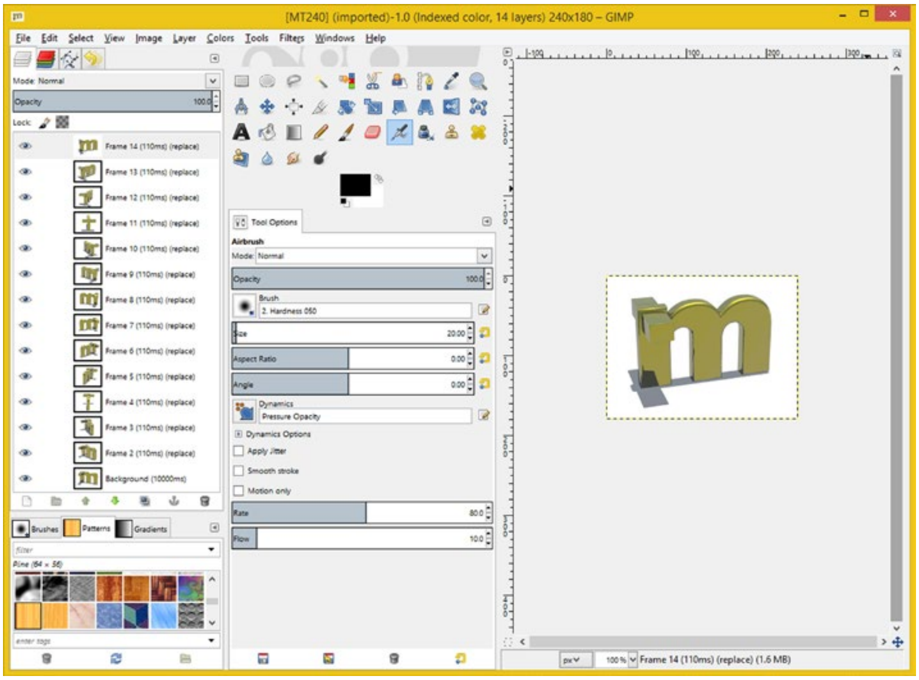
Reduce Noise: 100 %

Figure 2-2. *The GIMP Image Size dialog shows image size and dimension*

There's also an option to **Reduce Noise** which will always be introduced by scaling, which supports my contention earlier in the chapter that you want to match the content resolution to the screen size of the device dimensions, or the physical hardware screen pixel resolution.

Doing this will ensure the highest quality visual result for your digital image compositing projects and digital images.

Now, let's take a look at the image resolution features in the open source GIMP 2.8.14 software package, as shown in Figure 2-3. As you can see, the statistics for the image are right there in the title bar at the top of the GIMP digital image compositing software package, without having to open up a dialog at all.


[MT240] (imported)-1.0 (Indexed color, 14 layers) 240x180 – GIMP

Mode: Normal
Opacity 100.0

Frame 14 (110ms) (replace)
Frame 13 (110ms) (replace)
Frame 12 (110ms) (replace)
Frame 11 (110ms) (replace)
Frame 10 (110ms) (replace)
Frame 9 (110ms) (replace)
Frame 8 (110ms) (replace)
Frame 7 (110ms) (replace)
Frame 6 (110ms) (replace)
Frame 5 (110ms) (replace)
Frame 4 (110ms) (replace)
Frame 3 (110ms) (replace)
Frame 2 (110ms) (replace)
Background (10000ms)

Airbrush
Mode: Normal
Opacity 100.0
Brush 2. Hardness 050
Size 20.00
Aspect Ratio 0.00
Angle 0.00
Dynamics Pressure Opacity
Apply Jitter
Smooth stroke
Motion only
Rate 80.0
Flow 10.0

px 100 % Frame 14 (110ms) (replace) (1.6 MB)

Figure 2-3. *GIMP shows your image statistics in its title bar*

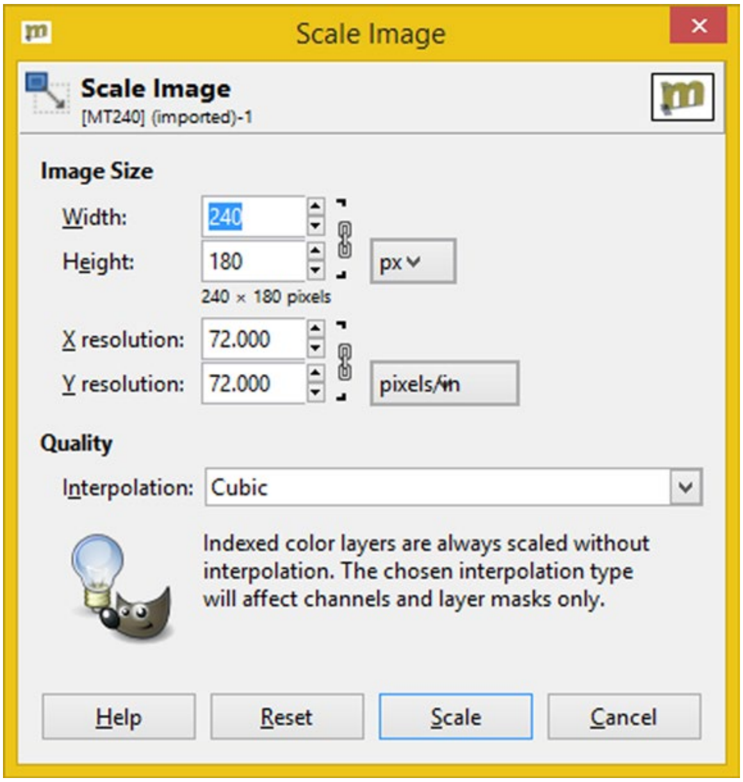
These statistics include the color depth, which we'll be covering in Chapter 5; the number of layers in this composite, which we cover in Chapter 8; and the resolution of your image, using the WxH format, which in this case is 240x180 pixels.

If you are wondering why these screenshots contain 14 layers, the file that I opened is in the **animated-GIF**, or **aGIF**, format, and thus each of my animation cels are on a different layer. In this case the digital image, or 3D animation, file data format contains my 3D animated Mind Taffy Design logo.

GIMP does have a dialog that can be used to see what the resolution for your imagery is, as well as to scale it to a new resolution, called the **Scale Image** dialog.

This can be found under the **Image** menu, on the top left side of the screen, as shown in Figure 2-3. If you want to take a look at it, follow the **Image ➤ Scale Image** menu sequence.

GIMP also calls its Image Size dialog the **Scale Image** dialog, as you can see in Figure 2-4. It allows you to select **Quality** via several different **Interpolation** (scaling) algorithms.


Scale Image

Scale Image
[MT240] (imported)-1

Image Size
Width: 240
Height: 180 px
240 × 180 pixels
X resolution: 72.000
Y resolution: 72.000 pixels/in

Quality
Interpolation: Cubic

Indexed color layers are always scaled without interpolation. The chosen interpolation type will affect channels and layer masks only.

Figure 2-4. *GIMP's Scale Image dialog*

We're going to cover scaling in Chapter 3, when we cover **aspect ratio**. Scaling is a very important consideration, which is why we match pixel resolution for an image to target device screen resolution, so the device or platform does not scale it for us, as doing so decreases quality.

You want to always have control over the scaling, also called **re-sampling**, of your digital image assets, so that you can assure the highest level of quality for your viewers.

As you might imagine, there's a relationship between the resolution of your image and its storage size (data footprint). Since your image pixels need to be stored, the more there are of them, the more data will be required to hold them in system memory. This seems kind of obvious, but I thought I'd state it.

Summary

In this second chapter, we took a look at the *x,y* pixel array display structure that holds your image pixels, and you learned about the concept of digital image resolution. We looked at the mathematics behind calculating the number of pixels that are contained in the digital image and looked at how you ascertain the image resolution in the two most widely used digital image compositing software packages, Photoshop CS6 and GIMP 2.8.14. We learned that both of these software packages also allow you to change the resolution of your image using a process called **resampling**, or scaling, which uses interpolation algorithms. We are going to be taking a closer look at this process in the next chapter.

We will also take a look at image **shape** defined by the resolution parameters for an *x,y* pixel array, or the concept of **image aspect ratio**. Last, we will also cover how image scaling can be done correctly, without distortion, and how your image can be cropped instead of scaled to control aspect ratio.

CHAPTER 3

The Shape of Digital Imaging: Aspect Ratio

Now that you have an understanding of the image resolution and how to calculate the number of pixels that make up your raster image, it is time to consider the **shape** (rectangular or square) of your digital imagery. Your digital imagery's shape, known in the industry as its "aspect ratio," is more important than you might think, especially when you're changing it using image scaling (known as resampling) or image cropping.

The aspect ratio should not be manipulated or changed in any way, in order to prevent **image distortion**. Developing full knowledge about the aspect ratio will allow you to make sure that you do not distort your digital imagery, and that is why this chapter is extremely important to your digital image compositing knowledge base.

After we look at the mathematics involved in calculating your aspect ratio, we will take a look at how to "lock" the ratio. **Locking the aspect ratio** allows you to eliminate distortion from the image-scaling process, whether that be up-sampling, or scaling the image resolution up, or down-sampling, or scaling the image resolution down. You can also avoid distortion by using **cropping**, which is removing pixels from the top, bottom and sides of your digital image.

Aspect Ratio: The Ratio of Pixels in 2D

Whereas the number of pixels contained in your digital image is called **resolution**, the **ratio of** *x* **to** *y* in your digital image is called the **aspect ratio**. Aspect ratio is a more complicated aspect, no pun intended, of image resolution, because it is the **ratio** of width to height, or *w*:*h*, within your image resolution specification, or *x*:*y* **when speaking of** an image's *x* and *y* axis.

The aspect ratio defines the shape of your digital image and also applies to the shape of the device display screen. For instance, smart watches have a square screen and widescreen iTV sets have a rectangular screen. We will be looking at how you can change the aspect ratio according to these different screen shapes over the course of the chapter.

A **1:1 aspect ratio** digital image (or display screen) can be said to be **perfectly square**. By its very definition, a 1:1 aspect ratio is the same as a **2:2** or **3:3** aspect ratio. It is important to note that it is the **ratio** between these two numbers, not the numbers themselves, that defines the **shape** of the image or screen, and that's why it is called an aspect ratio, although it's often called an **aspect** for short.

A 2:1 aspect ratio would use a **widescreen** aspect ratio, and a 3:1 aspect ratio could be referred to as a **panoramic** aspect ratio.

Screen Shapes: Common Display Aspect Ratios

Many HDTV resolution display screens, discussed in Chapter 2, use an HDTV **widescreen** aspect ratio of **16:9**. However, some displays use a less wide, or taller, **16:10** (or 8:5, if you prefer to go by lowest common denominators) aspect ratio. Even-wider screens will surely appear on the market soon, so look for **16:8** (which is also **2:1**) **ultrawide** screens, which will have a 2160 x 1080 screen resolution.

The original television screens were almost square, using a **3:2** aspect ratio. Computer screens featured a **4:3** aspect ratio, such as Macintosh's 512-by-384 screen resolution, or 640 by 480 for the PC. Once you learn how to calculate the aspect ratio in the next section of this chapter, you can check my math!

As time went on, PC displays kept getting wider, until a **2:1-widescreen** 2160 x 1080 display appeared in 2013. It won't be long before the Ultra High Definition, or UHD, 4K iTV Set 2:1 display (4096 by 2048 pixels) will show up in the market.

The most recent aspect ratio change was introduced by the smart watch industry, with square **1:1**-aspect-ratio displays showing up in 2015. Custom-screen aspect ratios can get fairly extreme; we've all seen the **9:1** aspect ratio screens in sports stadiums, ringing around the top and ends of the stadium walls.

Doing the Math: How to Arrive at the Aspect Ratio

An image aspect ratio is usually expressed as the smallest pair of numbers that can be achieved on either side of the aspect ratio colon. If you paid attention in high school when you were being taught about lowest (or least) common denominators, then this aspect ratio mathematics should be fairly easy to calculate.

I usually do the **mathematical matriculation** (say it five times rapidly, to make what you are about to do seem easier) by continuing to divide each side by a factor of two. Let's take a fairly oddball 1280 x 1024 (it's called *SXGA*) resolution as the example. Half of 1280:1024 is 640:512, and half of that, should be 320:256. Half of that is 160:128, and half of that is 80:64. Half of that's 40:32, and another half is 20:16. From 10:8, our last half gets us to **5:4**. Therefore, an *SXGA* resolution uses a 5:4 aspect ratio.

Interestingly, all the above ratios were the same aspect ratio, and all were valid. Thus, if you want to take the really easy way out, replace that *x* in your image resolution with the colon and you have an aspect ratio for the image. The industry standard involves distilling an aspect ratio down to its lowest format, as we've done here, as that is a far more useful ratio.

Using Imaging Software: Maintaining Aspect Ratio

To change a digital image's resolution in Photoshop you use the **Image ➤ Image Size** menu sequence, which was shown in Figure 2-1 of the last chapter. Notice the chain icon in the Image Size dialog in Figure 3-1 and how it connects the width and height data fields when it is turned on. This "locks" the aspect ratio, so that when you change one value, the other value will change, based on the current image aspect ratio, keeping the scaling **uniform** across both the *x* and the *y* axes.

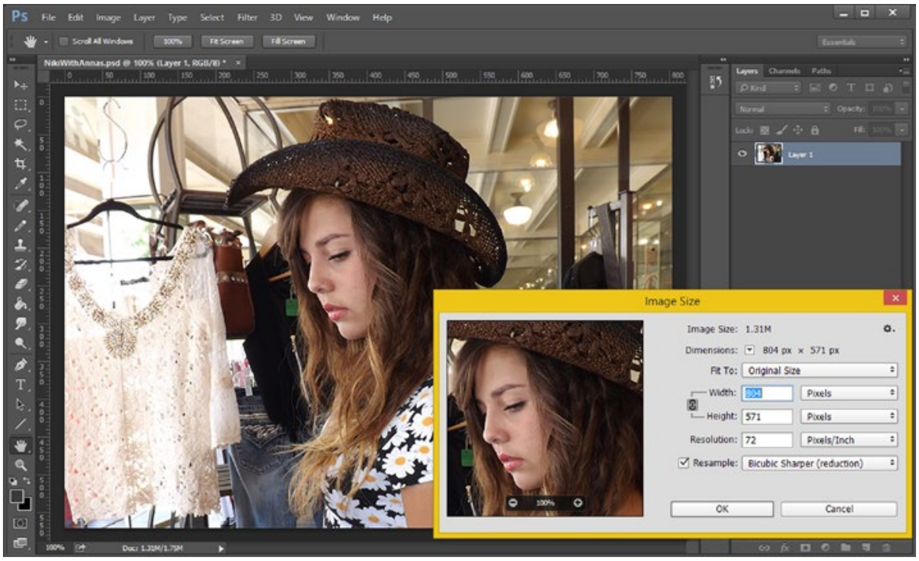


Image Size: 1.31M
Dimensions: 804 px × 571 px
Fit To: Original Size
Width: 804 Pixels
Height: 571 Pixels
Resolution: 72 Pixels/Inch
Resample: Bicubic Sharper (reduction)

Figure 3-1. *In Photoshop use an Image ➤ Image Size menu sequence*

If you look ahead a little bit in the chapter at Figure 3-2, you can also see the unlocked aspect ratio icon, and see how these two lines (connecting to Width and Height fields) disappear when the chain icon is clicked on and unlocked.

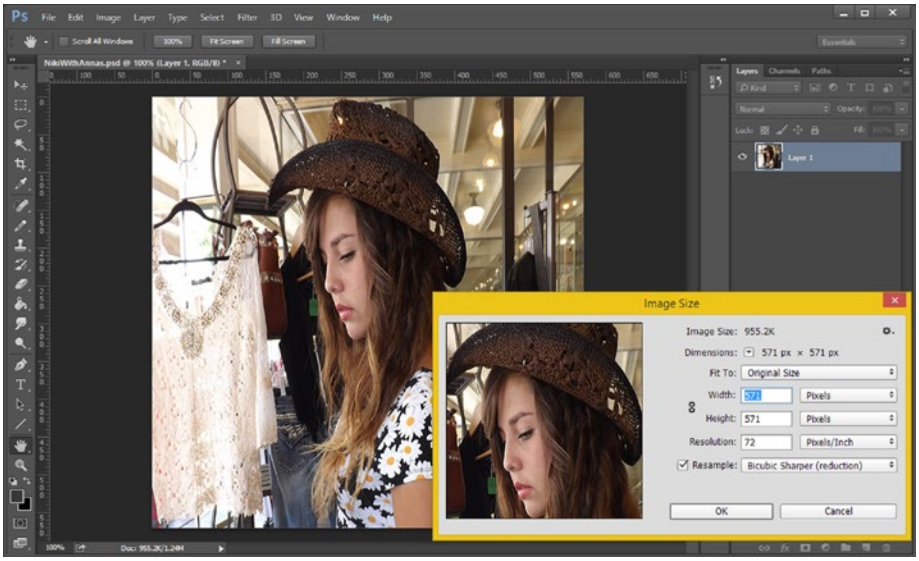

Ps File Edit Image Layer Type Select Filter 3D View Window Help
Scroll All Windows 100% Fit Screen Fill Screen
NikiWithAnnas.psd @ 100% (Layer 1, RGB/8) *
Layers Channels Paths
Layer 1
Image Size
OK
Cancel

Image Size: 955.2K
Dimensions: 571 px × 571 px
Fit To: Original Size
Width: 571 Pixels
Height: 571 Pixels
Resolution: 72 Pixels/Inch
☑ Resample: Bicubic Sharper (reduction)

Figure 3-2. *Image Size dialog showing the w:h aspect ratio unlocked*

The user interface design of this Image Size (Image Scaling) dialog demonstrates how important the concept of locking the aspect ratio in order to prevent distortion truly is to your digital image editing and compositing work process.

Let's unlock the aspect ratio in this dialog, and resize the image, so that it's a square 571-by-571-pixel image, so that you can see this distortion that is caused by changing the aspect ratio for yourself.

Figure 3-2 shows this image along with another preview image with the width distorted along the *x* axis. At some point you will see this mistake happen in images on the Internet. It is quite common, if you see a face that is unnaturally thin, or one that is unnaturally pudgy, you will likely be looking at x-axis and y-axis image aspect ratio distortion.

The more drastically you change the aspect ratio, the more noticeable the distortion will become. It is also much more noticeable with people and animals than it is with landscapes or textures such as pictures of stone or foliage, for instance.

The way to achieve this change in aspect ratio without a distortion occurring is to use image cropping rather than image scaling. Image cropping changes the number of pixels by cutting some of them off of the top, bottom or sides, rather than by scaling or re-sampling the same pixels to different resolution specifications.

Figure 3-3 shows the Photoshop **crop tool**, as selected on the left side of the screen, underneath the Magic Wand Tool.

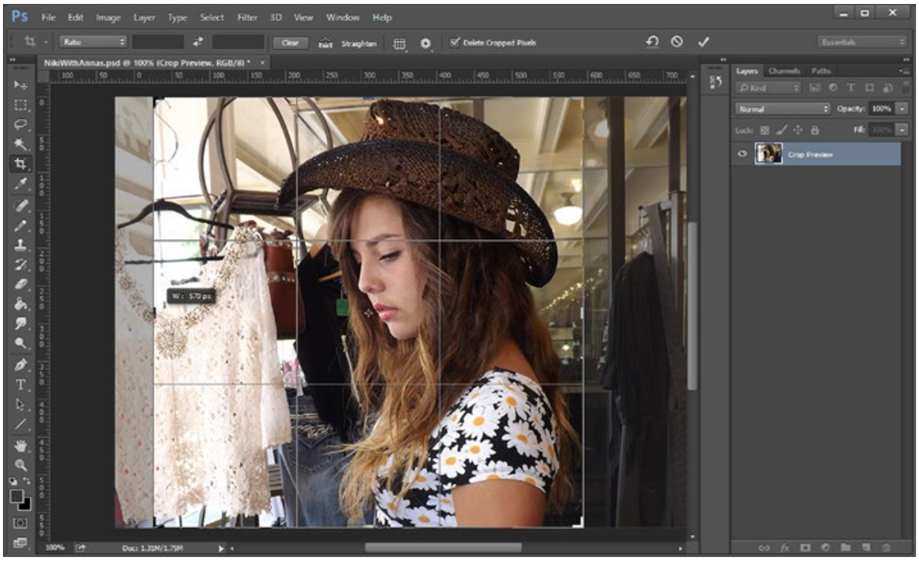
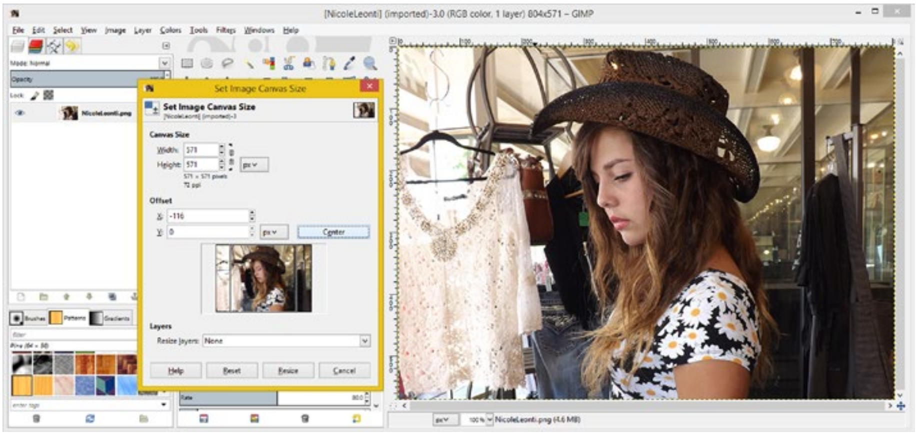


Figure 3-3. *Use Photoshop's crop tool to create an image with a 1:1 aspect ratio*

As you can see, the Photoshop tool provides you with **cropping grid guidelines** as well as with a **current image width in pixels** indicator, both overlaid over the top of the image.

These allow you to ascertain in real-time exactly what you are doing to your image and what you will get when you are finished with the tool. To complete a crop tool operation, you will need to double-click on the inside of your cropping area.

GIMP features a **Set Image Canvas Size dialog** that offers an image-cropping function, as you can see in Figure 3-4. Setting a new canvas size allows you to crop the digital image to your desired aspect ratio without introducing any distortion. It is important to note that Photoshop also has a **Canvas Size dialog** that can be used instead of the crop tool to achieve the same result. You can also click the **Center** button for your crop operation.


[NicoleLeonti] (imported)-3.0 (RGB color, 1 layer) 804x571 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Set Image Canvas Size
Help Reset Resize Cancel

Set Image Canvas Size
[NicoleLeonti] (imported)-3

Canvas Size
Width: 571
Height: 571 px
571 × 571 pixels
72 ppi

Offset
X: -116
Y: 0 px Center

Layers
Resize layers: None

Figure 3-4. *GIMP's Set Image Canvas Size dialog allows you to crop an image*

The result of the crop dialog is seen in Figure 3-5.

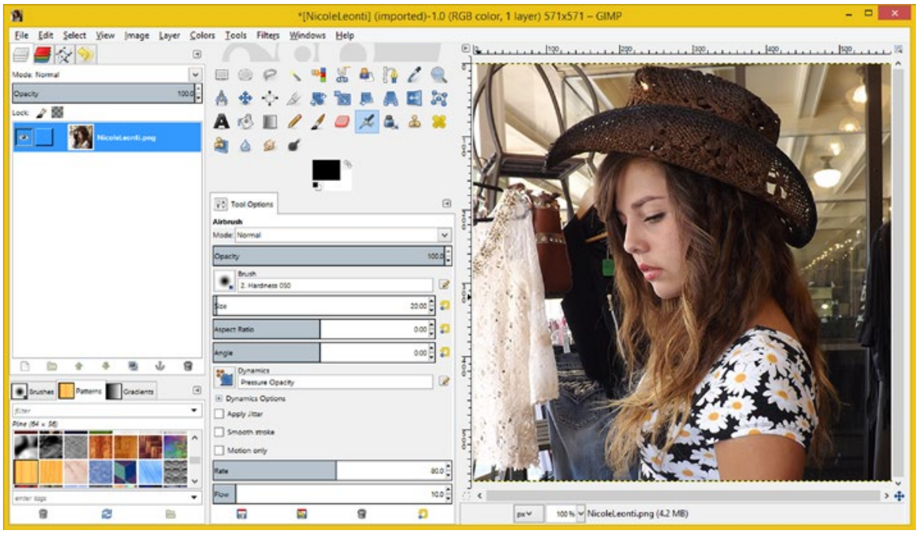
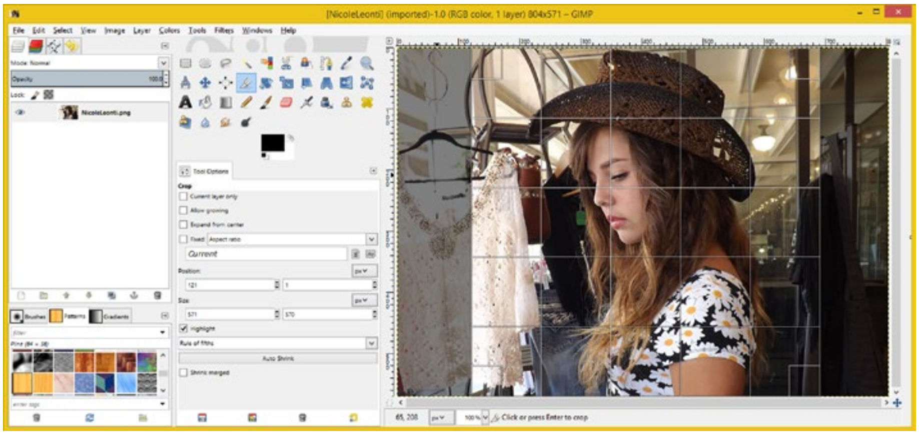

*[NicoleLeonti] (imported)-1.0 (RGB color, 1 layer) 571x571 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Tool Options
Airbrush
Brushes Patterns Gradients
NicoleLeonti.png (4.2 MB)

Figure 3-5. *The result of the Image ➤ Canvas Size operation*

Just like Photoshop CS has both an Image ➤ Canvas Size function and a crop tool, GIMP also has a crop tool. Look for the knife icon in the tool palette, shown as selected in blue in Figure 3-6. This tool has a plethora of options that can be seen underneath the tools palette.


[NicoleLeonti] (imported)-1.0 (RGB color, 1 layer) 804x571 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Tool Options
Crop
Click or press Enter to crop

☐ Current layer only
☐ Allow growing
☐ Expand from center
☐ Fixed: Aspect ratio
Current
Position: 121, 1 px
Size: 571, 570 px
☑ Highlight
Rule of fifths
Auto Shrink
☐ Shrink merged

__Figure 3-6.__ As highlighted in blue, GIMP has a crop tool (knife icon) with many options

The Photoshop options are shown along the top of the screen, as you can see in Figure 3-3. In GIMP, I selected the **Rule of Fifths** option to use a fine cropping grid. I positioned the left edge at position 121 on the x axis (I used 116 in Photoshop).

GIMP also shows me the size of the 571 pixel-image so that I can get the precise result I'm trying to achieve. As you can see, these two imaging tools provide different approaches to the cropping operation, but both are able to achieve the same professional level end result.

As you can see, both GIMP and Photoshop provide powerful image-cropping and image-scaling tools, if you use them wisely.

Summary

In the third chapter, we took a look at the concept of aspect ratio, which is used to define the rectangular shape of an image. You learned about some of the most popular aspect ratios used for today's consumer electronics hardware devices as well as the mathematics behind calculating your aspect ratio. Finally, we looked at how to avoid distortion when changing the resolution, or the aspect ratio, of the imagery, by either locking the aspect ratio during scaling or using the **Crop tool** or **Canvas Size** dialog. We went over how to access and use the crop tool and Canvas Size dialog in both Photoshop CS6 and GIMP 2.8.14.

In the next chapter, we'll take a close look at digital image **color theory**, and delve into how colors are represented in digital image compositing pipelines and in digital imagery.

CHAPTER 4

The Color of Digital Imaging: Color Theory

Now that you have an understanding of the positioning of pixels using an *x,y* grid, which defines the image resolution as well as the image aspect ratio, it is time to look inside of each pixel, and see how it defines its **color space**. Whereas a digital image holds its resolution and aspect ratio, from a data perspective, each pixel contains its own characteristics, such as the *x,y* grid (array) **position**, **color composition**, **transparency or opacity**, **blending mode**, and the like. We will be looking at all of these areas individually, each in their own chapters within this book.

The color of the pixels that make up the digital imagery ultimately define what an image looks like, and so color theory is an important topic here. Photoshop CS6 and GIMP 2.8.14 both feature a plethora of tools that will allow us to adjust color, as well as to select pixels using their color values.

If you're using modern-day consumer electronics devices, then you're using an **RGB**, or red, green, and blue, pixel color model. If you are working with a modern printer, then you are using a **CMYK**, or cyan, magenta, yellow, and black, color model.

If you're wondering why the acronym is not CMYB, it is because its black plate is known as the "key" plate, which the other color plates are aligned to (or aligned with).

In this chapter, we will be looking at these two color models as well as the best color model, or "mode," to use with your digital image compositing pipeline. Now let's get started!

Color Theory: Using Pixel Color Channels

Whether you are compositing digital imagery to be used on display screens or to be printed using inks on printers, your pixels will be holding color values that use one or more "channels." These are sometimes referred to as "plates" in the printing industry, continuing the tradition of older printing presses that used metal plates, some of which are still in use today. In Photoshop and GIMP, the **color channels** have their own floating palette and allow you to work on just one color channel (or plate), which can be quite useful for special effects, or other advanced image operations.

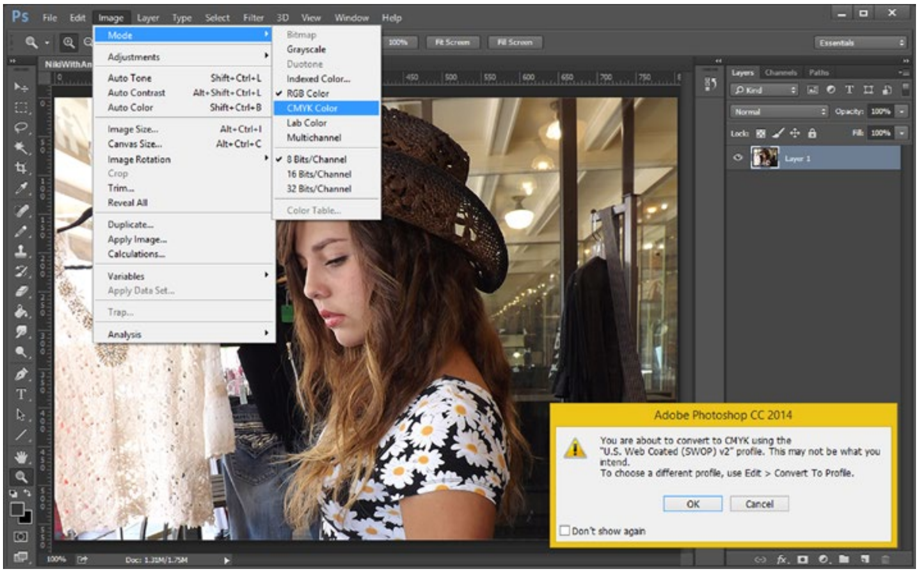
Subtractive Color: Print Using CMYK Color Plates

Modern color printers' use at least the four ink colors in CMYK allowing for the creation of a wide range of different colors. High-end inkjet printers are coming out with six- or eight-color ink cartridge models that create more color variations.

Printer inks use the **subtractive color** model, according to which inks remove color from each other—that is, two colors **darken** the other (by removing the light inks) when combined, which is why it is called a "subtractive" color process.

For example, the cyan and magenta inks create purple, which is darker than either source color ink is by itself.

A subtractive color model produces fewer colors than an additive color model, which I will discuss next. This is quite logical if you think about it, because subtraction takes away, whereas addition adds more. In Photoshop, as well as GIMP, your color modes can be accessed using an **Image ➤ Mode** menu sequence and a submenu that contains all of the supported color modes. The path is shown in Figure 4-1 along with a CMYK warning dialog. The warning dialog in the lower-right corner advises you that you are about to convert from a RGB to a CMYK **color profile** and suggests that more control over the conversion can be exercised using an **Edit ➤ Convert to Profile** menu sequence.



Adobe Photoshop CC 2014

You are about to convert to CMYK using the "U.S. Web Coated (SWOP) v2" profile. This may not be what you intend.
To choose a different profile, use Edit > Convert To Profile.

OK Cancel

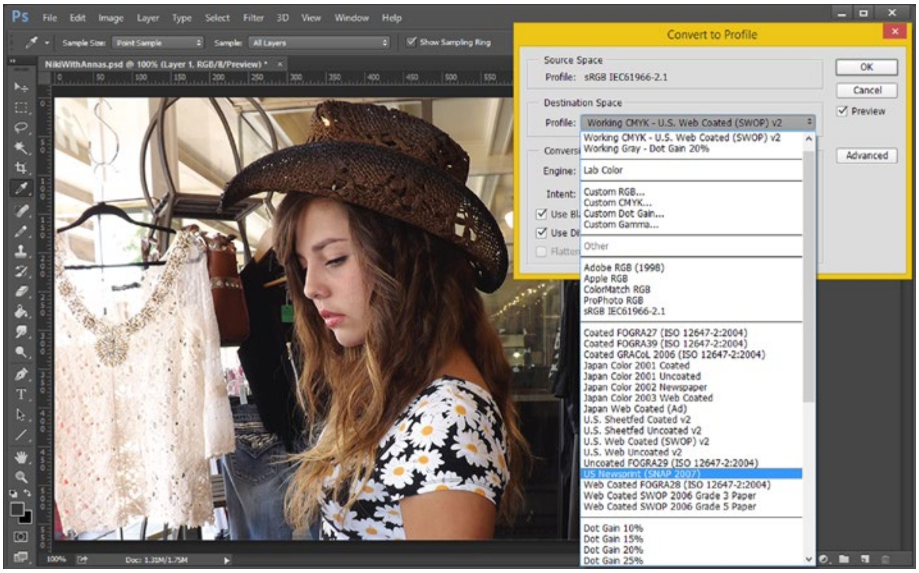
☐ Don't show again

Figure 4-1. *You can use an Image ➤ Mode menu sequence to set color mode*

Generally you will want to work in an RGB color profile as RGB has 16,777,216 colors to work with, whereas CMYK would have significantly fewer colors, and because smart phones, iTV sets, tablets, e-book e-readers, laptops, notebooks or netbooks do not support the CMYK color model.

Just like you want to match resolution and aspect ratio pixel for pixel, within each of these pixels, you also want to match those color channels which are supported for that device (printer or display) that you are creating your digital imagery for.

As you can see in Figure 4-2, the dialog accessed by this menu sequence allows you to select predefined color profiles, which are a standard in the industry to create different applications such as newspaper print (as selected) or sheet-fed print.



Convert to Profile

Source Space

Profile: sRGB IEC61966-2.1

Destination Space

Profile: Working CMYK - U.S. Web Coated (SWOP) v2

Ps File Edit Image Layer Type Select Filter 3D View Window Help
Sample Size: Point Sample
Sample: All Layers
Show Sampling Ring
NikiWithAnnas.psd @ 100% (Layer 1, RGB/8/Preview) *
OK
Cancel
Preview
Advanced
Engine:
Intent:
Working CMYK - U.S. Web Coated (SWOP) v2
Working Gray - Dot Gain 20%
Lab Color
Custom RGB...
Custom CMYK...
Custom Dot Gain...
Custom Gamma...
Other
Adobe RGB (1998)
Apple RGB
ColorMatch RGB
ProPhoto RGB
sRGB IEC61966-2.1
Coated FOGRA27 (ISO 12647-2:2004)
Coated FOGRA39 (ISO 12647-2:2004)
Coated GRACoL 2006 (ISO 12647-2:2004)
Japan Color 2001 Coated
Japan Color 2001 Uncoated
Japan Color 2002 Newspaper
Japan Color 2003 Web Coated
Japan Web Coated (Ad)
U.S. Sheetfed Coated v2
U.S. Sheetfed Uncoated v2
U.S. Web Coated (SWOP) v2
U.S. Web Uncoated v2
Uncoated FOGRA29 (ISO 12647-2:2004)
US Newsprint (SNAP 2007)
Web Coated FOGRA28 (ISO 12647-2:2004)
Web Coated SWOP 2006 Grade 3 Paper
Web Coated SWOP 2006 Grade 5 Paper
Dot Gain 10%
Dot Gain 15%
Dot Gain 20%
Dot Gain 25%
100%
Doc: 1.31M/1.75M

Figure 4-2. *A Convert to Profile dialog offers a profile selector*

It is important to note that digital cameras use the RGB color mode to capture digital image data using something called a charge-coupled device, or CCD for short. This means that data coming into your digital image editing and compositing software package is in the RGB color mode, so it is logical to also edit it in this color mode, as a conversion to CMYK will cause your image to lose some colors.

For these reasons, most digital image compositors use an RGB color model for doing their image compositing work and then convert to CMYK before they print. In this way, you can produce both a CMYK and an RGB representation of your current digital image compositing project. This is an advantage, as your image compositing work can be used on displays as well as printed.

This conversion process to print being used less and less by image compositors might account for GIMP not having moved to add a full CMYK color model to its software package. This is especially true since the age of digital cameras and display devices has greatly reduced the public's reliance on analog and print media forms. This trend continues each and every day, as more smart phones, smart watches, tablets, e-readers, and iTV sets explode into the current marketplace.

As you can see in Figure 4-3, GIMP's **Image ➤ Mode** submenu does not have subtractive color mode support. If you do want to use GIMP for CMYK color models and print work, though, there's a way around this. A plug-in called **Separate** is available for GIMP that adds the CMYK color mode support, and AUR offers an advanced version of this plug-in called **Separate+ (Plus)** for Linux users. It can be found at the following URL: `https://aur4.archlinux.org/`.

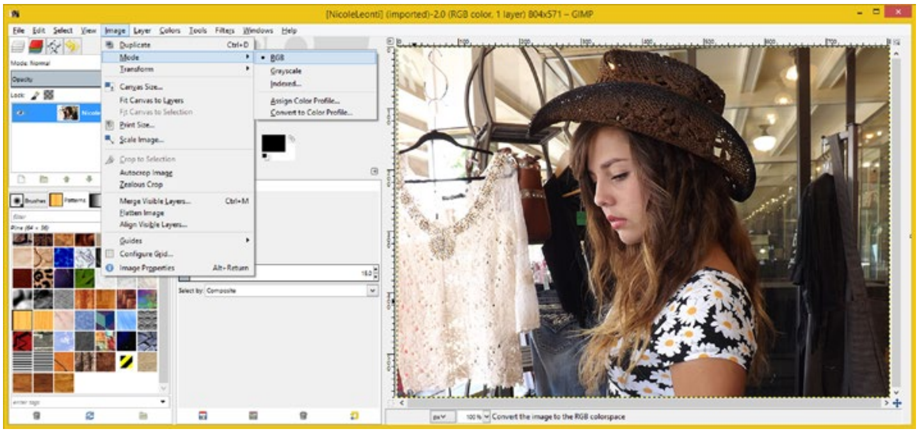


Figure 4-3. *The GIMP Image ➤ Mode menu only supports additive color*

These plug-ins can separate an RGB image into CMYK, offer color management using ICC profiles and LCMS, allow the soft-proofing of colors, and allow you to attach the ICC CMYK profile to your color-separated digital image files.

The Separate+ plug-in has the same features as Separate, but it also has the ability to convert RGB profiles to other RGB profiles and offers duotone support.

Additive Color: Display Using RGB Color Channels

LED, LCD, and OLED displays, on the other hand, use an **RGB additive color** model, in that they use **light** instead of ink. An advantage of using light, or additive color, is that more color is possible. If you have ever seen colors coming out of a prism or in a rainbow, you realize that color produced using light is nothing short of spectacular, and is something ink simply can't reproduce. This is why the book focuses primarily on RGB color.

Additive color is the opposite of subtractive color, and so, whereas combining red and green ink gives you a dark purple color, combining red and green light gives you a **yellow** color!

Pixel color is defined using three colors of light: red, green, and blue; hence "RGB." These colors are present in varying amounts in each pixel. Display screens utilize additive colors, by summing together the wavelength of light for each RGB color plane (or channel). This color mode is utilized in popular LED, LCD, and OLED displays used in tablets, e-readers, smart watches, smart phones, iTV sets, netbooks, PCs, and laptops.

In Figure 4-4, I have turned on **Use Color Channel Color** mode in the Photoshop Preferences dialog. Now, by selecting the **eye icon** on the left side of the color channel, you can see the green intensity data values using the Green channel's color.

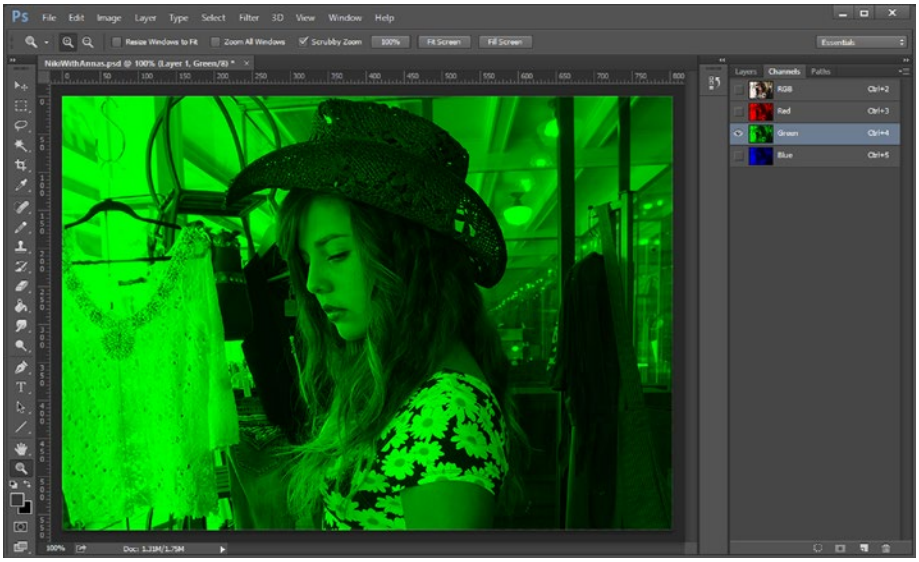


***Figure 4-4.** The Green color channel, selected in Photoshop CS6*

GIMP uses the same color channel palette, using an icon that mimics a stacked layer of colored vellum. As shown in Figure 4-5, I have turned on the Green and Red channels by using the same eye icons, to produce the **Green + Red = Yellow** additive color summary effect. You can selectively apply effects and specifications to desired color channels.

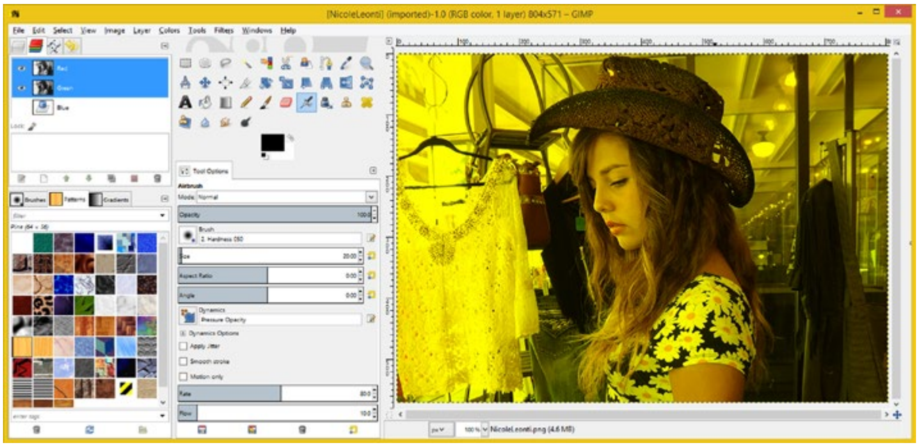


***Figure 4-5.** The Red and Green channels selected in GIMP*

This separation of color channels into their own **modal** palette makes image-editing software more powerful by allowing users to isolate color by individual channels.

The ability to work on, and apply effects to, individual color channels is an important and powerful feature of digital image editing software.

Additive color can be used to create tens of millions of different color values. We are going to take a look at the mathematics behind how this can happen in the next section.

The Mathematics of RGB Color: Multiplying Your Intensities

The number of red, green, and blue **shades,** or **levels of intensity,** of light you have available to mix together will determine the total amount of color that you can reproduce. In today's digital devices, pixels can produce **256** levels of light intensity for each red, green, and blue (RGB) color.

Colors will be generated for each image pixel, so every pixel in your image will have 256 levels of color intensity for each of these RGB color data values.

Each of these RGB channels, plates, or planes, would use **one byte**, or **eight bits**. Eight bits of data will hold up to 256 different values, so you have 256 levels of brightness for each red, green, and blue color.

The color-intensity (brightness) data inside each of the digital-image pixels in your digital image is represented with a **brightness level** for each color. This level can range between zero (brightness turned off) and 256 (brightness fully on).

To calculate the total amount of available color is easy; it is again simple **multiplication**. If you multiply 256 by 256 by 256, you get **16,777,216** colors, which represents the total unique color combinations you can get, with 256 levels per color to work with across three different color channels. You will often see this 16,777,216 color value represented for marketing purposes as 16.8M colors, rounding it up a bit.

Representing RGB Color Values: Using Hexadecimal Data Values

The 256 levels of brightness for each red, green, and blue color value in each pixel allow you to set eight bits of value to control the variation of color brightness for each of the color values. The color brightness can vary from a minimum of zero to a maximum of 255, which, if you count from 1, comes to 256 different color intensity data values.

The number of bits that are used to represent a digital image pixel color can be coded by using **Base16**, or **Hexadecimal**, notation. Base16 counts from **zero to F**, so that you will have sixteen values (bits) to represent color with. Having two of these hexadecimal values will give you **16 x 16 = 256** values, which is the number of colors that you will need to have for each of your RGB color channels. Hey! This all makes sense!

In Android and HTML5, JavaScript, and CSS3, this is done using the **Hash** or **Pound** sign (#), and in Java, this is done using the **0x prefix**. For instance, to represent the color **BLACK**, for which all values are off, or zero, the code would look like:

```
colorValue = #000000    // Android, HTML5, JS, CSS3 Color Value Hexadecimal
colorValue = 0x000000   // Java and JavaFX Color Value Hexadecimal
```

The color **WHITE,** on the other hand, has all pixels fully on, and as such would be represented using the following code:

```
colorValue = #FFFFFF    // Android, HTML5, JS, CSS3 Color Value Hexadecimal
colorValue = 0xFFFFFF   // Java and JavaFX Color Value Hexadecimal
```

For the color **RED,** you would only turn on the Red channel pixels, so your hexadecimal code would look like this:

```
colorValue = #FF0000    // Android, HTML5, JS, CSS3 Color Value Hexadecimal
colorValue = 0xFF0000   // Java and JavaFX Color Value Hexadecimal
```

For the color **GREEN**, you would turn on your Green channel pixels, so your hexadecimal notation would look like:

```
colorValue = #00FF00    // Android, HTML5, JS, CSS3 Color Value Hexadecimal
colorValue = 0x00FF00   // Java and JavaFX Color Value Hexadecimal
```

And, similarly, the color **BLUE** has the Blue channel pixels turned on, so your hexadecimal notation would look like:

```
colorValue = #0000FF    // Android, HTML5, JS, CSS3 Color Value Hexadecimal
colorValue = 0x0000FF   // Java and JavaFX Color Value Hexadecimal
```

In the next chapter, we will be taking a look at how these bit depths apply to different digital image file formats, such as an 8-bit GIF, or a 16-bit TIFF, or a 24-bit JPEG, or a 32-bit PNG, or even a 48-bit or 64-bit RAW image file format. Lots of learning left to go, but I promise, it will be fun!

Summary

In the fourth chapter, we took a look at the concept of **color theory**, which defines how the color is represented in each pixel of a digital image. You learned about the difference between **additive** (RGB) and **subtractive** (CMYK) color modes as well as the mathematics behind calculating the total available colors for a standard 24-bit RGB digital image. We went over how **color modes** are set in Photoshop and GIMP, how the RGB color mode is converted into industry-standard CMYK color representations in Photoshop CS6, how to add CMYK support to GIMP using the Separate+ plug-in, how color channels are represented, and how to select them for use in both Photoshop CS6 and GIMP 2.8.14.

In the next chapter, you will learn about digital-image **color depth** and how different image formats support different color depths.

CHAPTER 5

The Digitization of Digital Imaging: Color Depth

Now that you have an understanding of the primary two color theory models used in digital image editing and compositing, and the reason why we are going to be focusing on the **RGB mode** from here on out, it is time to take a more in-depth look at how colors are represented **digitally** for each pixel. The number of bits used to define color within a digital image and its file format are traditionally referred to as the digital image **color depth**. The most common color depth, used in the most common image format, JPEG, is **true color**, or **24-bit** color. There are also other image formats that use different bit-level color depths, such as the **GIF** format, which uses **8-bit** or **indexed color**; the **Targa** format, which uses **16-bit** or **high color**; and the **RAW** format and **HDRI** format, which can use up to 48-bit and even use 64-bit color.

We will build upon the knowledge from the previous chapter and learn about the **32-bit RGBA** mode, or if you prefer, the **ARGB** color mode (and 32-bit color depth). This mode is important for compositing, because it adds an **alpha** (**A**) **channel**, which I will cover in detail in Chapter 6, because it is just that important to the digital image compositing work process and pipeline (what I call the data processing path).

We will be looking at four primary color depths used in digital imaging today. The 8-bit or "indexed color" color depth is used for data footprint optimization, a 16-bit "high color" color depth is rarely used, the popular 24-bit "true color" is used in JPEG, and a compositing 32-bit PNG format has an alpha channel, which we will be covering in great detail in the next chapter as it gives us an ability to blend or fade imagery.

We will be looking at these four color depths during this chapter as well as the best color depth, or "bit depth," to use with your digital image compositing pipeline. It is key that you understand how color values are digitized for pixels.

Color Depth: Bit Levels that Define Color

In addition to the number of bits used to define color within a digital image and the file format, color depth uses those data bits to define the amount of color available to each of the pixels in a digital image. Common color depths used in the digital imaging industry include 8 bit, 16 bit, 24 bit, and 32 bit. I'll outline the common file formats for each bit depth (color depth) in this chapter and mention the new **High Dynamic Range Image** (HRDI) and **RAW** formats, which can hold 48 bits or 64 bits of color image data, here as well. I will also point out which widely used digital image formats are supported in Android, HTML5, Java and JavaFX, in case you are a multimedia producer, or a computer programmer, as well.

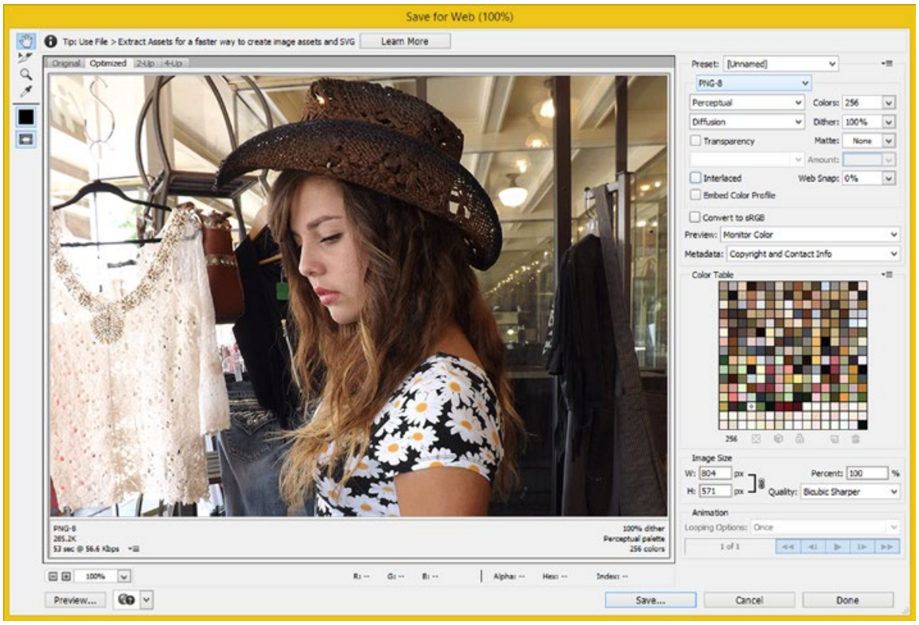
A medium color depth image features a 16-bit color depth and thus contains 65,536 colors (calculated as 256 times 256). It is supported by the Targa (TGA) and Tagged Image File Format (TIFF) digital image formats. If you want to use digital image formats other than the standard GIF, JPEG, and PNG in your Java programming, you would need to import the Java Image library.

True color imagery features the 24-bit color depth, and thus contains over 16 million colors. This is calculated as 256 times 256 times 256, which will equal 16,777,216 colors. File formats supporting 24-bit color depth include: JPEG (or JPG), PNG, BMP, XCF, PSD, TGA, TIFF, and RAW. Java supports three of these: JPEG, PNG24 (24 bit), and PNG32 (32 bit). Using 24-bit color depth will give you the highest quality level. This is why I'm recommending the use of PNG24 or PNG32 for Java, HTML5 and Android based apps and games.

Indexed Color: Using Palettes to Hold 256 Colors

The lowest color depth exists in 8-bit **indexed color** images. These images feature a maximum of 256 color values, which is why they are considered 8-bit images, and use an indexed "palette" of colors, which is why they are called "indexed color images." Popular image file formats for indexed color include GIF, PNG8, TIFF, BMP, and TGA.

The way that you convert your 24-bit, true color imagery data into an indexed color image format (GIF or PNG8) using Photoshop is to use a **File ➤ Save for Web** menu sequence. This opens the Save for Web dialog shown in Figure 5-1, which will allow you to set the file format (GIF or PNG8); number of colors; color conversion algorithm (perceptual, selective, adaptive, or restrictive); dithering algorithm (diffusion, pattern or noise); and a number of other advanced options.



Preset: [Unnamed]
PNG-8
Perceptual   Colors: 256
Diffusion   Dither: 100%
Transparency   Matte: None
Amount:
Interlaced   Web Snap: 0%
Embed Color Profile
Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
Color Table
256
Image Size
W: 804 px   Percent: 100 %
H: 571 px   Quality: Bicubic Sharper
Animation
Looping Options: Once
1 of 1

PNG-8
285.2K
53 sec @ 56.6 Kbps

100% dither
Perceptual palette
256 colors

100%   R: --   G: --   B: --   Alpha: --   Hex: --   Index: --

Figure 5-1. *You can use Save for Web in Photoshop to create indexed color images*

The number of colors that I selected for this image is 8 bit (256 colors), to show you what a good job the indexed color algorithm does. Notice that the image preview looks just like a true color image. You can select any number of colors, between 2 and 256, that is.

As an example of color depth, if you selected 2 colors that would be a 1-bit (PNG1) image, 4 colors would be a PNG2 (2-bit color depth) image, 16 colors would be a 4-bit PNG4 color depth image, 64 colors would be a 6-bit PNG6, and 128 colors would be a 7-bit PNG7 image.

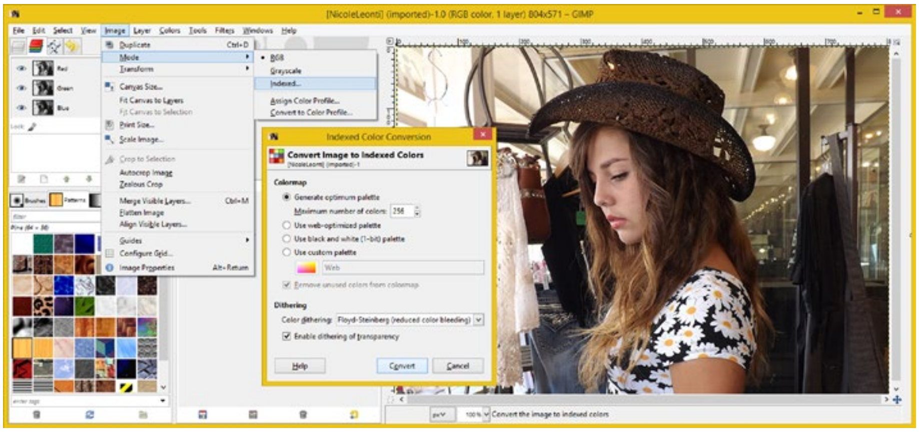
The **color selection algorithm** drop down, which is to the left of the number of colors drop-down selector, offers several different algorithms that **select colors** for your indexed color **palette**. I prefer **Perceptual**, however there are also **Selective**, **Adaptive**, and **Restrictive** algorithms. The best way to approach the use of these algorithms is visually, selecting one and then taking a look at your preview for the result. You can observe your file size in the bottom-left corner of the screen, as can be seen in Figure 5-1.

You can see the **dithering selection algorithm** drop-down menu underneath the color selection algorithm menu. I prefer **Diffusion** dithering, but there is also a **Pattern** setting, and a **Noise** (random) setting, as well. Dithering is the process of placing dot patterns between two different colors in a gradient to reduce an effect often seen in indexed color imagery, called "banding." There is also a drop-down menu to set the amount of dithering that the algorithm will apply; I usually leave it set at **100 percent**, and let the algorithm do its thing.

The **Interlaced** option usually adds to the data footprint (file size), and allows **progressive rendering**, for use on your web page. The **Matte** option allows **edge smoothing** against color values which you can supply if you are using an alpha channel in an indexed color image. We'll be covering this in Chapter 6.

You also have the choice of using the following functions: **Embed your Color Profile** (for this image, inside of the file) and **Convert to sRGB** (for the image color profile), but generally, you will want to keep these unselected, unless you have a good reason for using them present itself to you.

Next, let's take a look at how to convert true color image data to indexed color image data using GIMP 2.8.14. As you will see in Figure 5-2, the way to get this dialog in GIMP is to use the **Image ➤ Mode ➤ Indexed** menu sequence. This will bring up an **Indexed Color Conversion** dialog. This has fewer options than Photoshop's Save for Web dialog, but the important ones are there, so that you can specify color depth (bit level) and dithering.



Indexed Color Conversion

Convert Image to Indexed Colors
[NicoleLeonti] (imported)-1

Colormap
- (●) Generate optimum palette
  - Maximum number of colors: 256
- ( ) Use web-optimized palette
- ( ) Use black and white (1-bit) palette
- ( ) Use custom palette: Web
- [✓] Remove unused colors from colormap

Dithering
- Color dithering: Floyd-Steinberg (reduced color bleeding)
- [✓] Enable dithering of transparency

Help | Convert | Cancel

Figure 5-2. *GIMP's Image ➤ Mode ➤ Indexed creates indexed color*

In GIMP, I usually use one of the **Floyd-Steinberg** diffuse dithering algorithms; in this case, it is the one that reduces color bleeding, which keeps the color edges clean and tight. I also select the full 256 colors, which you'll usually want to do, unless you need to save a few kilobytes in data footprint, which isn't really worth the loss in image quality.

High Color: Using a 15-Bit and 16-Bit Color Depth

A medium color depth exists in rare 15- or 16-bit high-color images. The reason I say "rare" is because 15- and 16-bit file formats are not widely supported in the open source platforms (Android, Java, JavaFX, HTML5, CSS3, JavaScript) that everyone is using across all of their cool consumer electronics devices.

A 15-bit high-color format supports **32,768** color values, usually configured in the **555 RGB** format, which means **5 bits** of data are used for each of your 15-bit color RGB color channels.

A 16-bit high-color format supports **65,536** color values, usually configured in the **565 RGB** format, which means **5 bits** of data are used for each of the **Red** and **Blue** channels with **6 bits** of data being utilized in the **Green** color channel.

You might be wondering why this extra bit of resolution, representing an additional 32,768 levels of color intensity, is allocated to the Green color channel. The reason is that the green channel in your image holds the majority of the **contrast data** for your imagery! This is one of those key tidbits of information that makes this book so valuable, as the application of this knowledge to how you do things in GIMP or Photoshop will make a major difference in the quality of the resulting imagery.

Therefore, using a 565 RGB configuration for a high-color image will always produce a crisper, clearer image result. The digital imagery formats that support high color include: TIFF, TGA, BMP, and (Photoshop) PSD. Unfortunately, none of these formats are supported in Android or HTML5, or even in Java, JavaFX, or JavaScript, at least without using third-party image file format import libraries.

Next, let's take a look at the true-color (24-bit) image format, which we used to learn about color theory in Chapter 4.

True Color: Using a 24-Bit Color Image Backplate

As I have already pointed out, one of the most widely used file formats in digital imaging is the JPEG file format, which comes in one flavor: 24 bit. Other file formats that support 24 bits of color data include BMP, TIFF, TGA, PSD, and PNG24. Since PNG also supports 8 bit (PNG8) or 32 bit (PNG32) color, I call a 24-bit PNG format PNG24, to be precise.

As you learned in Chapter 4, true color (also known as "truecolor") imagery uses an **RGB 888** color configuration. The difference between the true color file formats comes in one primary differentiating factor: **lossy** compression.

Lossy compression means that an algorithm, which is also called a "codec" (**COde-DECode**), is throwing away some of the data in order to achieve a smaller data footprint. The opposite is **lossless** compression. It is useful to save your original file in a lossless data format, prior to applying any lossy compression, primarily this would be the **JPEG** algorithm.

Lossless compression, used by the PSD, BMP, TGA, and TIFF formats, doesn't throw away any original image data; it applies an algorithm that finds patterns that result in less data being used. This guarantees high resulting image quality.

"Digital vehicles" that deliver your digital images primarily use true color, 24-bit imagery. These include web sites, e-books, mobile applications, iTV programs, smart watches, digital signage, and similar digital image display devices and digital image social sharing forums.

In image compositing, however, only one layer (or plate) in the pipeline, which we'll get into in detail in Chapter 8, uses 24-bit data. I like to call it a **backplate**, or **background** plate (layer), because it's at the bottom (back) of the layer stack, and therefore doesn't need (or use) **transparency**.

All of the other layers in your compositing stack above the background plate need to support transparency and therefore require **32 bits** of **data**, which is also known as **ARGB** or **RGBA**.

This transparency is provided by a fourth channel, known as the **alpha channel**. In the following section, I am going to introduce you to this fourth channel, and we are also going to cover the alpha channel in more detail in Chapter 6, and every chapter after that, as transparency is an important digital image compositing concept.

True Color Plus Alpha: Using 32-Bit Digital Images

Besides 8-bit, 16-bit, and 24-bit digital images, there are 32-bit digital images. Formats that support 32-bit color data include PNG, TIFF, TGA, BMP, and PSD. I like to use PNG32 as it is supported in HTML5, Java, JavaFX, CSS3, JavaScript, and Android, whereas the other file formats are not integrated with the open source OSs and browsers like the PNG format is. These 32 bits of image data include 24 bits of RGB color data, plus 8 bits of "alpha" or **transparency value** data, held in what is commonly referred to as an **alpha channel**.

Since you now know that 8 bits holds 256 values, it will make sense to you that an alpha channel will hold 256 different **levels of transparency** data values for each pixel in a digital image.

This is an important concept for your digital image compositing because it means that layers that hold this 32-bit image data allow some portion (from 0 to 256, or all) of the pixel's color to bleed through to (or blend with) layers below it, allowing a composite (blended) image to be created.

Finally, let's take a look at how the four data channels translate into Java, JavaFX and JavaScript code, HTML5 and CSS3 mark-up, and Android Studio. That's right, we're talking about hexadecimal notation, for all of you programmers and multimedia producers that happen to be reading this foundational material.

In Android, HTML5, JavaScript, and CSS3, hexadecimal for color plus alpha takes the **ARGB** format, so the two alpha value positions go **first**. For example, to represent **TRANSPARENT**, which in code has all values off, or zero, this would look like this:

```
colorValue = #00000000   // Android, HTML5, JS, CSS3 Alpha+Color (ARGB)
colorValue = 0x00000000  // Java and JavaFX Alpha+Color Value Hexadecimal
```

The color **WHITE**, with **50 percent translucency**, on the other hand, uses all RGB pixels full on for WHITE plus 8 (which is 7 when you count from zero) in each of the alpha channel data value slots:

```
colorValue = #77FFFFFF   // Android, HTML5, JS, CSS3 Alpha+Color Value
colorValue = 0x77FFFFFF  // Java and JavaFX Alpha+Color Value Hexadecimal
```

The color **RED**, with **25 percent translucency**, would use a value of 3 for each alpha channel slot and turn on only the Red channel pixels using FF, or turn the Red fully on and leave Green and Blue fully off as follows:

```
colorValue = #33FF0000   // Android, HTML5, JS, CSS3 Alpha+Color (ARGB)
colorValue = 0x33FF0000  // Java and JavaFX Alpha+Color Value Hexadecimal
```

For the color **GREEN**, with **37.5 percent translucency, you** would use a value of 5 for each alpha channel and turn on only the Green channel pixels using FF, or turn the Green fully on and leave Red and Blue fully off:

```
colorValue = #5500FF00   // Android, HTML5, JS, CSS3 Alpha+Color (ARGB)
colorValue = 0x5500FF00  // Java and JavaFX Alpha+Color Value Hexadecimal
```

And for the color **BLUE**, with **75 percent translucency**, you would use a value of B for each alpha channel slot and turn on only the Blue channel pixels using FF, or turn the Blue fully on and leave Green and Red fully off:

```
colorValue = #BB0000FF   // Android, HTML5, JS, CSS3 Alpha+Color (ARGB)
colorValue = 0xBB0000FF  // Java and JavaFX Alpha+Color Value Hexadecimal
```

In the next chapter, we will dive into how to use alpha channel data in Photoshop and GIMP—it just gets more exciting!

Summary

In this fifth chapter, we took a look at the concept of color depth, which defines how many bits are used to represent the color in each pixel of your digital image. You learned about the different digital image file formats that support these different bit levels, like the 8-bit GIF, and the 16-bit TGA, and the 24-bit, JPEG and the 32-bit PNG. We looked at how true color mode is converted into indexed color palettes in both Photoshop and GIMP and reviewed concepts such as color selection algorithms, dithering algorithms, and progressive interlacing. You learned about high color 15- and 16-bit color imagery, revisited true color 24-bit imagery, and added an 8-bit alpha channel to create a 32-bit digital image format that is perfect for image compositing, especially if you are using HTML5 with CSS3, Android Studio, or Java and JavaFX. This is because PNG32 allows you to export your compositing layers from Photoshop or GIMP into NetBeans 8.1 (the Java and HTML5 IDE) or IntelliJ 14 (the Android IDEA).

In the next chapter, you'll learn more about the digital image **alpha channel** and how it can be used to facilitate image compositing pipelines by allowing more than one layer to exist.

CHAPTER 6

The Transparency of Digital Imaging: Alpha Channel

Now that you have an understanding of the basic color depths used in most digital image editing and compositing pipelines, we are going to be focusing on the 32-bit **RGBA mode** from here on out. The RGBA mode is tantamount for all of the compositing layers in an image composite other than the 24-bit color image back plate. We are also not going to focus on High Dynamic Range Imagery (HDRI) because it is data heavy and not practical for the types of usage that most of the book's readers will target, which include the Internet, e-books, iTV sets, smart phones, smart watches, tablets, netbooks, laptops and so forth.

In this chapter we are going to be focusing on the **alpha** component of the **ARGB** (I'll use these acronyms interchangeably) **image data mode**. Notice I'm a stickler here and am not calling this a "color mode," because only the **RGB** components are color. This other **A** component is **transparency**, which is not a color at all but a way to combine color across layers, planes, or plates in your digital image compositing "pipeline."

We will also be taking a look at how to access the **alpha channel** in Photoshop and GIMP as well as some valuable tools in both programs. These tools will allow you to work more easily with the transparency data that is held in the alpha channels.

Alpha Channels: Defining Transparency

Let's take a look at how alpha channels define digital image pixel transparency values, and how these channels can be used for compositing digital imagery. Alpha channels can provide transparency inside of your digital image compositing software, which I would term a "static" use, and can also be used via PNG32 image assets to composite digital imagery in real time using platforms such as Java, JavaFX, JavaScript, HTML5, CSS3, and Android. I would term the latter a "dynamic" use, as the code allows you to access your pixel transparency values by the millisecond, and animate this data, in any way that you like.

Digital image compositing involves the seamless blending together of more than one layer of digital imagery, and, as you might imagine, having control over the pixel transparency is an extremely important concept. Digital image compositing is often utilized in industries from graphic design to feature films, in game design, and more recently, for applications development.

Digital image compositing would be used when you want to create an image on your display that appears as though it were one single image (or even an animation) when it is actually the seamless collection of more than one composited image layers.

One of the principle reasons you would want to set up an image, or animation, composition, is to allow yourself to exert more control over the various elements in an image composite by having components on different **layers**. We'll be covering layers in much more detail, in Chapter 8.

To accomplish multilayer compositing, you always need to have an alpha channel transparency value, which you can utilize to precisely control the **blending** of the pixel's color with the color of those pixels in that same *x,y* image location, in other layers in the compositing pipeline (layer stack) below it.

Like the RGB color channels, the alpha channel will have **256 levels** of transparency ranging from 100 percent transparent (0) to 100 percent opaque (255). Each pixel will have different alpha transparency data, just as each pixel will have different RGB color data. Therefore the alpha channel is exactly like the color channels it accompanies, except that it defines blending, whereas the other three channels are defining red, green, and blue color value amounts, or levels of intensity.

Let's see how this looks in Photoshop CS6 using a simple example where we put a ring element over the model in the image that I have been using thus far for the book. As you can see in Figure 6-1, I've added the **RingElement.png**, PNG32 digital image to the top of the layer stack in Photoshop CS to show the alpha channel data (transparent versus opaque areas) for a decorative ring element, and what this alpha channel capability allows you to accomplish within your digital image compositing pipeline.

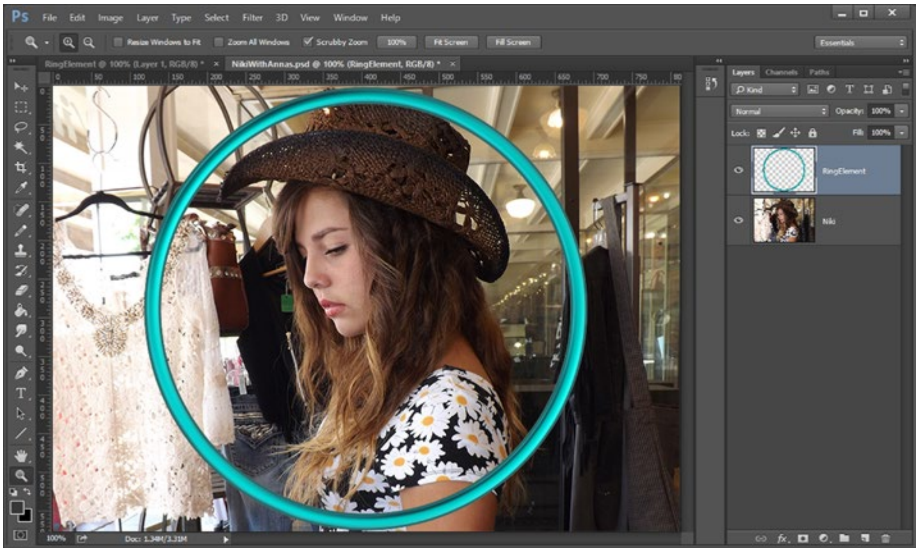


Figure 6-1. *A PNG32 RingElement added to the layer on top of Niki in Photoshop*

You can see that the alpha channel transparency value is shown for each pixel using a **checkerboard pattern**, seen in your layer preview icon in the **Layers palette** on the right-hand side of the screen.

The checkerboard pattern shows in any area of the layer that has a transparency value, so if a pixel (or area of pixel values) is partially opaque (50% translucent, for instance) it will show 50% of the checkerboard pattern. This is an example of a simple digital image composite, and, as you can see, the visual result is seamless.

You can see the checkerboard transparency representation in Figure 6-2, along with the **Channels palette**, selected on the right side of the screen.

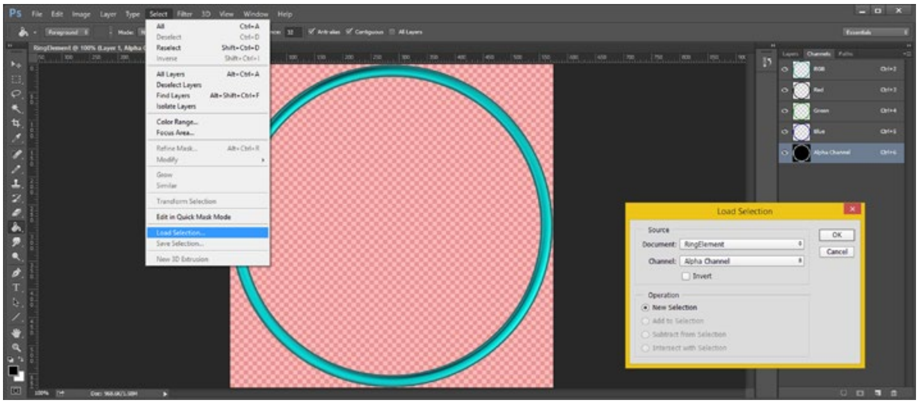

Load Selection
Source
Document: RingElement
Channel: Alpha Channel
[ ] Invert
Operation
(●) New Selection

Figure 6-2. The alpha channel, and the Load Selection dialog

I also selected the **Alpha Channel**, in the **Load Selection** dialog, which I accessed using the **Select ➤ Load Selection** menu sequence. This will produce a **red mask** over the image to better show you where the alpha channel data is (or alternately, where it is not). We'll be learning more about masking during Chapter 7, when we start actually working on pixels inside of the alpha channel to achieve advanced digital image compositing results.

I showed you how I selected my RingElement data, so that I could go on to use an **Edit ➤ Copy** menu sequence, to copy that data into the clipboard, and then paste it into the Niki image composite, shown in Figure 6-1, in the selected right-hand tab.

To place the ring image data in another layer on the top of the Niki layer, I selected the Niki layer and used the **Edit ➤ Paste** menu sequence. This pasted the selected pixel data into a layer automatically created above the Niki layer, filling any unused pixels with alpha transparency values, and automatically creating an alpha channel for the new layer. Finally, I double-clicked the layer label and supplied it with my own **RingElement** layer label, which can be seen selected in gray in Figure 6-1.

Next, let's take a look at how you would create the same compositing pipeline using GIMP. There is a really cool feature in GIMP 2.8.14 that makes bringing 32-bit imagery, such as the RingElement PNG32 file, into GIMP. This feature automatically creates a layer in the image compositing pipeline (layer stack) and places the imported imagery into this layer component.

Using the GIMP **File ➤ Open as Layer** menu sequence is far easier than the work process that I used in Photoshop CS, which you see in Figures 6-1 and 6-2. To use this command, select the **Niki.png** layer, shown selected in blue in Figure 6-3, and use a **File ➤ Open as Layer** sequence, shown on the left in Figure 6-3.

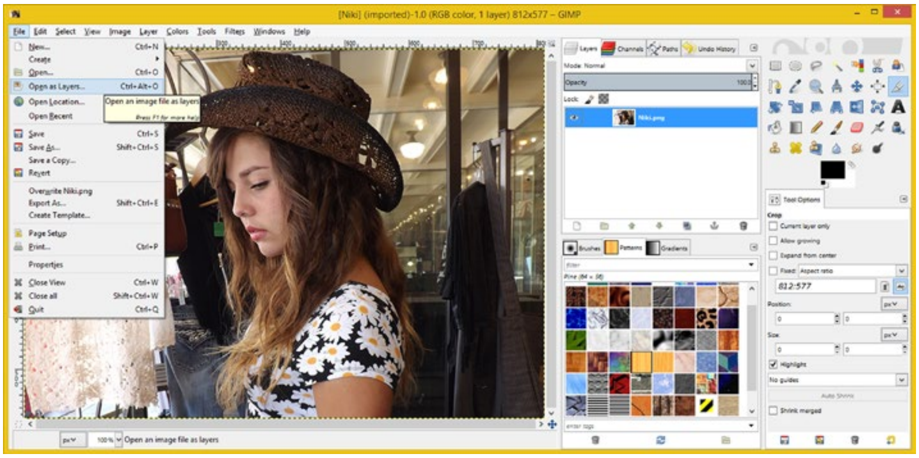


__Figure 6-3.__ You can use the Open as Layer menu option to insert a ring in GIMP

If you select a RingElement.png file from your **File Open** dialog, GIMP will add it for you, as a compositing layer, above your Niki.png layer, as shown in Figure 6-4. You will see that GIMP also uses a checkerboard pattern to represent transparency as you can see in the Layer palette on the right side of Figure 6-4. Notice that GIMP also shows you the pixel dimension of the RingElement.png layer, using a black-and-yellow dashed square, framing the hoop in the middle of the Niki image composite, on the RingElement layer.

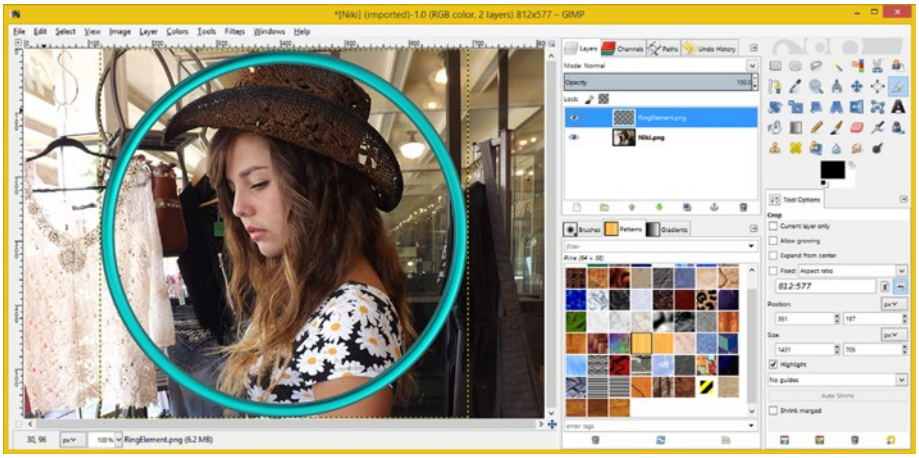

*[Niki] (imported)-1.0 (RGB color, 2 layers) 812x577 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
RingElement.png
Niki.png
RingElement.png (6.2 MB)

__Figure 6-4.__ The RingElement.png layer shows dimension and alpha

Alpha channels aren't only used for defining transparent areas in your compositing pipeline (layer stack), they can also be used for more creative purposes such as **storing selections**.

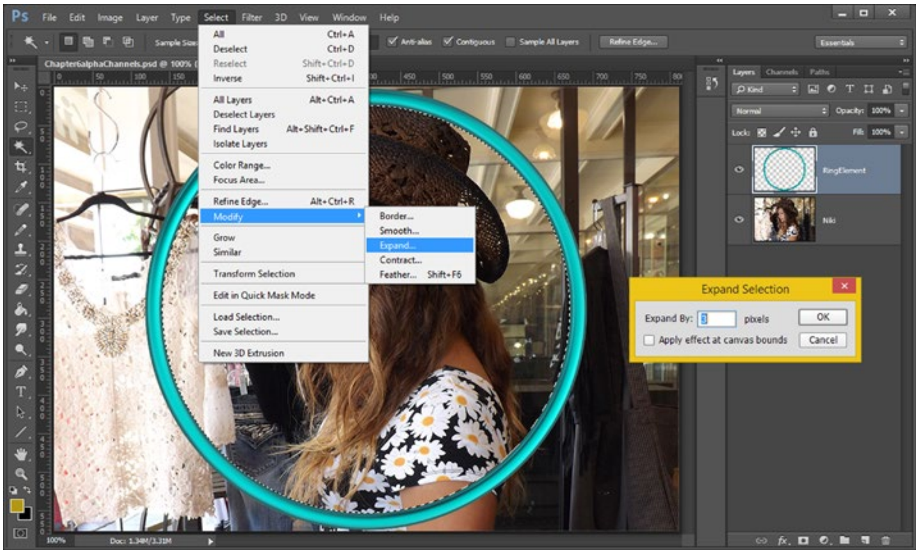
Selection: Using Alpha for Selection Sets

As you have already seen in Figure 6-2, alpha channels can also be used to store objects so that they can be easily selected for later use in your digital image compositing projects. We'll be looking at selection sets more in Chapter 7, when we look at **masking**, but I will show you some of the tools and techniques here, so that we can bridge into the related topics in these later chapters in the book more seamlessly.

Selecting Alpha Channel Data in Photoshop CS

Let's take a look at how this works by continuing with a Photoshop example we started in Figure 6-1. I have included the PSD file with this book; it is called **Chapter6alphaChannels.psd** and is in the files repository for this book on the apress.com website, if you want to open it now, and follow along with me.

The first thing you'll want to do is to click the **magic wand tool** at the top of the left-hand toolbar, as seen selected in Figure 6-5 (it looks like a sparkler).



Expand Selection
Expand By: 3 pixels
Apply effect at canvas bounds

Figure 6-5. Using Photoshop's Magic Wand tool to select the inside of a ring

Then, click inside the middle of the ring, and make sure your **RingElement** layer is the one that is selected in the right panel. Digital imaging software is "modal," as we will get into in greater detail during Chapter 10 covering **modal operations**.

As you can see in Figure 6-5, a center area of your ring will then become selected, with what is popularly termed in the industry as “marching ants.”

You can expand this area of selection, if you like, by a few pixels using the **Select ➤ Modify ➤ Expand** menu sequence and the **Expand Selection** dialog, which are also seen in Figure 6-5. I decided to expand my circular selection by 3 pixels in width.

I did this to make sure that my selection went underneath the edge of the ring, so there was no visible edge transition right at the edge of the ring, where the tool will put your selection edge. This is a popular digital image compositing trick that compositing professionals often utilize.

Then, click the **Niki** layer, and use **Select ➤ Inverse**, to select the image pixels outside the ring as seen in Figure 6-6.

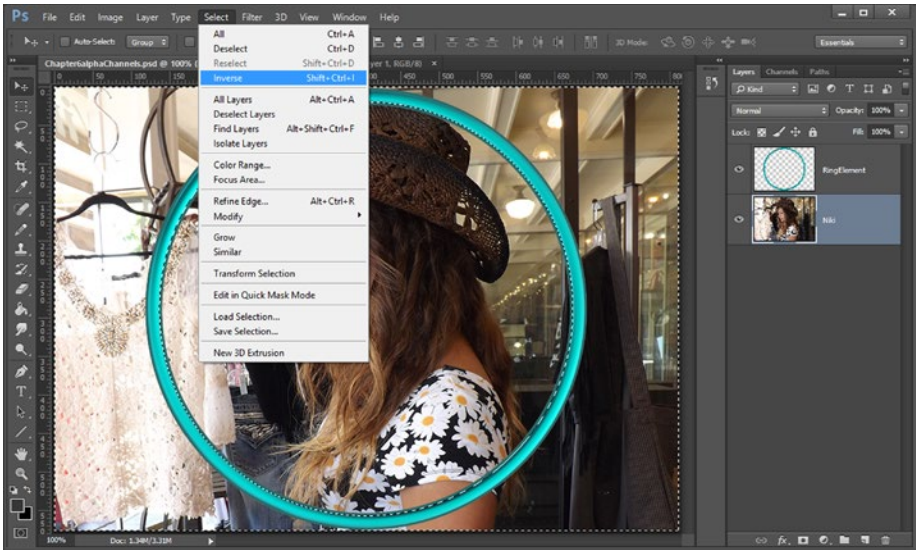


Figure 6-6. *The selection inverted and the Niki layer selected*

You can hit the **Delete** key, to **delete pixels**, as seen in Figure 6-7.

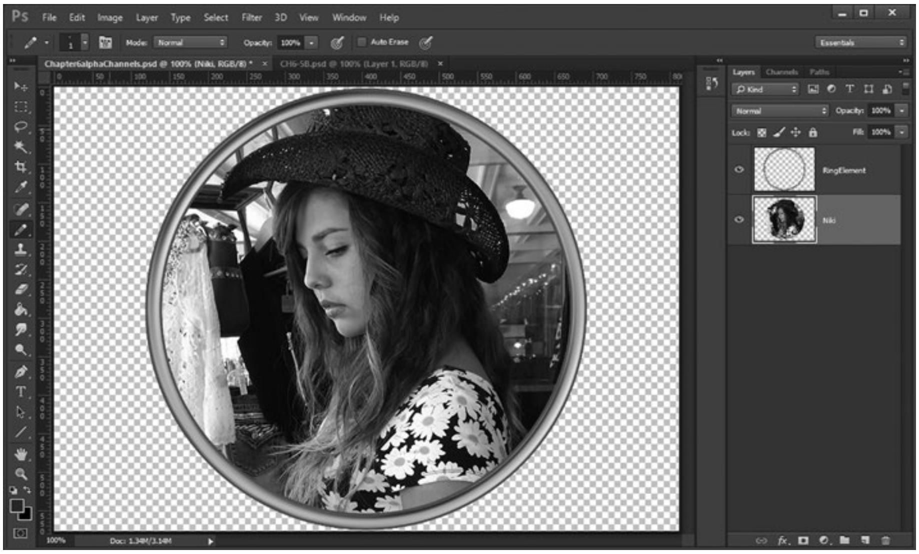


Figure 6-7. By using the Delete key, you can delete the selected pixels

Since Photoshop CS is **modal**, the key "move" is to select the pixels while the **RingElement** layer is selected, and before you press the Delete key, **change the selected layer** to the **Niki** layer, so that those are the pixels that will be removed. This will leave only the round Niki decorative element, ready to use on a web site, an app, or social media.

Now let's take a look at the work process for doing the same thing using GIMP, being that ultimately these two digital imaging software packages are quite different, as far as your work process goes. This particular process is fairly similar for the two software packages, others are not similar at all.

Selecting Alpha Channel Data in GIMP

In GIMP, the contiguous pixel (color or transparency) selection tool is called the **Fuzzy Select tool**, rather than the Magic Wand tool in Photoshop. The icon is quite similar, and is shown selected in blue in the top-right corner of Figure 6-8. Make sure the **RingElement.png** layer is selected and click inside of the ring to create your selection. Next use the **Select ➤ Grow** menu sequence to access a **Grow Selection** dialog. If you grow your selection by **3 pixels**, you will make sure your selection, and resulting image data, goes underneath the ring completely.

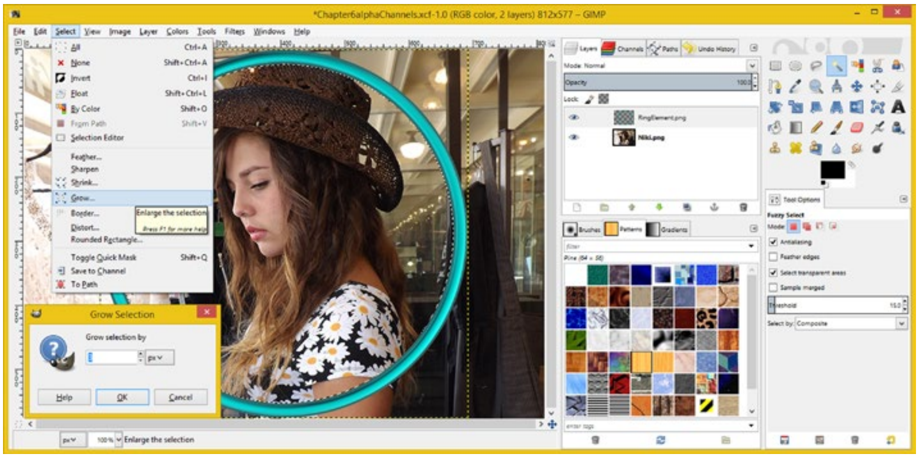


Figure 6-8. *To make sure the image goes underneath the ring completely, start by clicking inside of the ring and then use Select ➤ Grow and enlarge by 3 pixels*

Essentially I'm expanding a selection to make sure its edge is "hidden" behind the RingElement. You can see the selection edge is visible on the inside of the ring element in Figure 6-8, whereas, in Figure 6-9, it is underneath the ring element, even though your selection marching ants look like they're actually on top of the ring element (they're not).

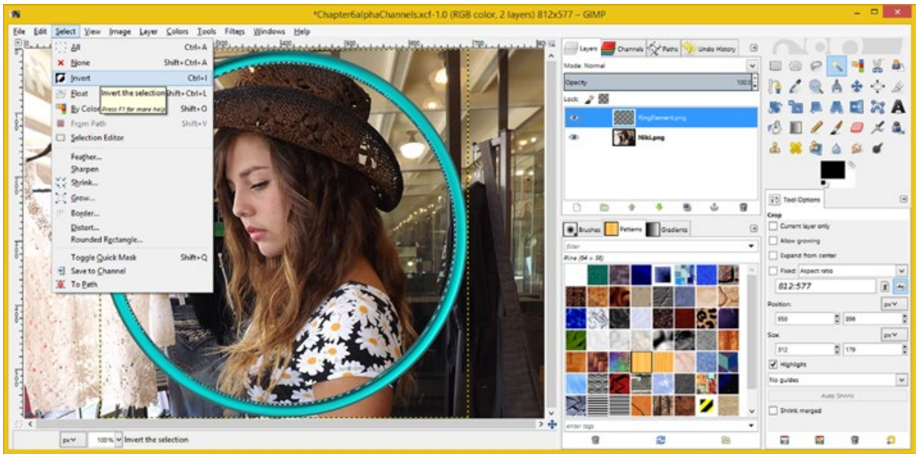


Figure 6-9. *To invert the part of the image that is selected, click inside the ring and then use the Select ➤ Invert menu*

I will be using this technique a lot, over the course of this book, so that you will become familiar with expanding your selection so that it is not right on the edge of your composite layer, which might cause slight edge imperfections in an image.

Next, use a **Select ➤ Invert** menu sequence to invert the selection. Doing this will select the part of the image that is underneath and outside of the ring instead of that which is inside of it—leave what is inside of the ring intact. The menu sequence can be seen in Figure 6-9, along with the inner ring selection once it has been expanded 3 pixels outward by using the Grow Selection dialog. Now, all you have to do is to select the Niki.png layer and click **Delete** to remove the background image pixels which are located outside of the ring element!

Figure 6-10 shows the GIMP screen after I have selected the Niki.png layer, and hit the **Delete** key. As you will see, GIMP uses a white background color behind the layer, one of the many ways that the program differs from Photoshop CS6.

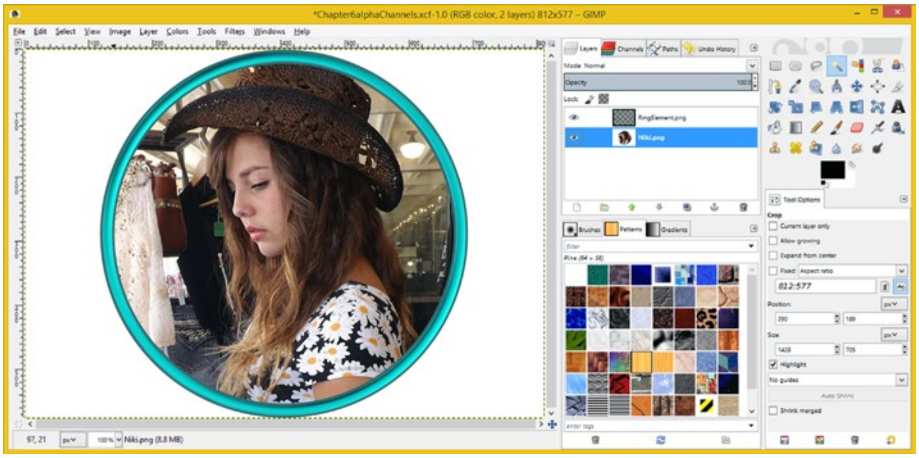


Figure 6-10. *You can select the Niki.png layer and hit the Delete key to remove the background image*

If you want to get rid of the white background and have the transparency, you will need to right-click on the Niki.png layer, with your original 3-pixel-expanded selection activated, and then select the **Add Layer Mask** dialog and **Selection** option.

This will turn the selection into the **layer mask**, making the layer itself transparent in its own background plate rather than using the white color. However, that's a topic for Chapter 7, when we get into a more advanced look at masking techniques.

We will be using layer masks more and more as this book progresses, so you will become familiar with this important and powerful concept. I just wanted to expose you to it here early on, so that you could start getting familiar with it.

Fortunately, that chapter is next, so this turns out to be another seamless transition between chapter topics, which I am trying to apply consistently in this book so that it flows nicely.

Summary

In this sixth chapter, we took a closer look at the concept of the digital image **alpha channel**. Image alpha values define how a pixel represents translucency within each of the compositing layers within the digital image compositing pipeline. You learned about how Photoshop and GIMP each represent alpha data and about the **Layers** and **Channels** palettes used to display this data. You also learned how to composite a PNG32 image object overlay with alpha values over a PNG24 image object that does not have any alpha values, because it is only a backplate, also known as a background image).

We also went over how to select areas of transparent pixels in Photoshop using the Magic Wand tool and in GIMP using the Fuzzy Select tool. In addition, we looked at how to **expand** and **invert** a selection and **delete** selected pixels to create even more pixel transparency in in our digital image compositing pipeline.

In the next chapter, you will learn more about digital image **masking** and how to use pixel selection tools to “pull” a digital image mask and save it for later use by using an alpha channel. As you can see, there is a good reason for the chapter order in this book!

CHAPTER 7

The Isolation of Digital Imaging: Masking Tools

Now that you have an understanding of alpha channel data and how it can be used in a digital image compositing pipeline, as well as how to use it to "store" digital image objects, the next logical step is to learn how to use **masking tools** to extract subject matter from your digital images. This is called "pulling a mask" in the digital image compositing industry.

Photoshop and GIMP have a number of masking tools that can be used to achieve this **pixel extraction** end result. We will be looking at some of them in this chapter along with some of the approaches and work processes that can be used to **pull a mask** out of your 24-bit digital image assets, and thereby turn them into 32-bit image assets. The assets that result include alpha channels. The alpha channel serves to isolate any subject material in images that you want to composite in a layer stack.

Masking: Isolating and Extracting Pixels

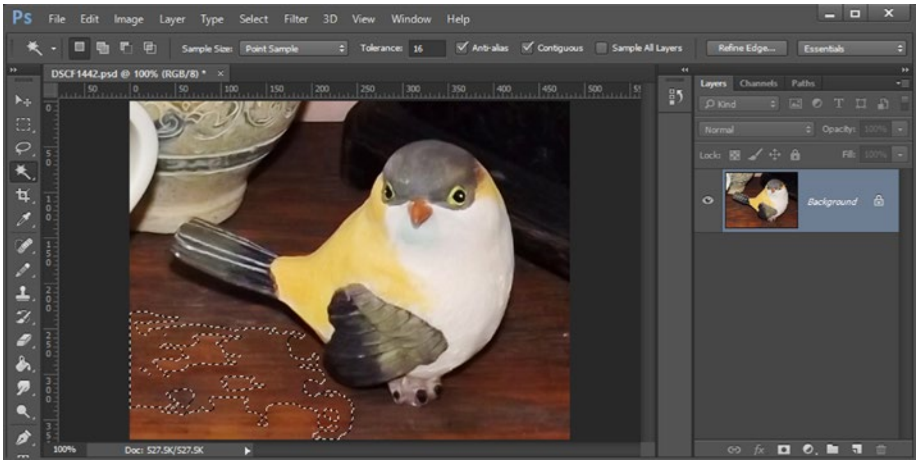
Masking is the process of **extracting** an **irregularly shaped** area of pixels from an image, usually for the purpose of digital image compositing. Generally, the desired outcome is the combination of these pixels with another image to achieve some artistic end result, or a special effect. This is done with **layers**, which we're going to be covering in detail in Chapter 8, as well as more advanced features, such as **Porter-Duff** blending and transfer modes. All of these functions use these complex, multi-modal operation, digital image compositing software packages, such as the ones that we're using for this book, both of which I recommend that you acquire, and master.

I'm not going to go into a whole lot of terms and theory and the like regarding masking because it's more about the work process and all the different tools the software packages offer to select pixels, in a significant number of different ways.

If you do not want to learn how to mask images by hand, you can purchase a **green screen** or **blue screen**, and photograph your subjects (objects) in front of it and then use one of the popular **plug-ins** that auto-generates your mask for you.

Photoshop Magic Wand: Tolerance and Sampling

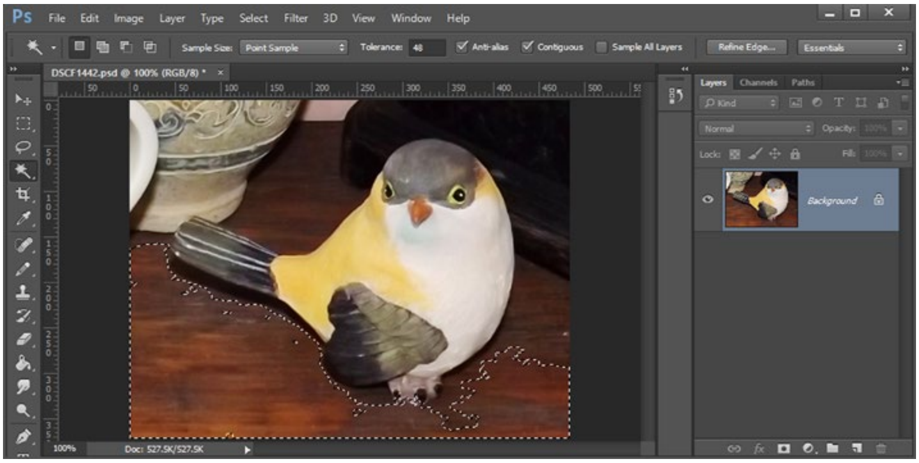
Let's take a closer look at how the **Magic Wand** masking tool in Photoshop works by extracting a bird figure from a PNG24 image. In Figure 7-1, I have set a **Tolerance** of **16** and clicked in the wood area of the image to show what the tolerance threshold is for this particular image. Tolerance is how many similar colors in the **point sample** (pixel) that is clicked on will be included in the selection (set) of pixels. Let's increase the Tolerance.


Ps
File Edit Image Layer Type Select Filter 3D View Window Help
Refine Edge...
Essentials
DSCF1442.psd @ 100% (RGB/8) *
Layers Channels Paths
Background
100%
Doc: 527.5K/527.5K

Sample Size: Point Sample Tolerance: 16 ✓ Anti-alias ✓ Contiguous Sample All Layers

Figure 7-1. Result of clicking on the wood tabletop after setting a Tolerance of 16

Figure 7-2 shows the visual result of using a three-times-greater Tolerance setting of 48. As you can see, you will use this Tolerance setting extensively, to get the tool to give you the selection result that you desire.


Ps
File Edit Image Layer Type Select Filter 3D View Window Help
Refine Edge...
Essentials
DSCF1442.psd @ 100% (RGB/8) *
Layers Channels Paths
Background
100%
Doc: 527.5K/527.5K

Sample Size: Point Sample Tolerance: 48 ✓ Anti-alias ✓ Contiguous Sample All Layers

Figure 7-2. Result of clicking on the wood tabletop after setting a Tolerance of 48

The work process I use is to set the Tolerance and point sample sample size, click a color I want to eliminate, and then use the **Select ➤ Deselect** (**Select ➤ None** in GIMP) menu sequence to deselect the selection if it is not close to what I'm trying to achieve, and set a different Tolerance, and sample size, if necessary. Then I take a similar pixel color sample and repeat the process until I get close to the selection result I need.

Now you know what different Tolerance settings produce. You can change the value at any time during your masking work process (do this between clicks to select pixel areas). Let's move on to the **Sample Size**, which we will change from a single point sample, done in pixels, to a 3-by-3 average pixel block, as is shown in Figure 7-3.

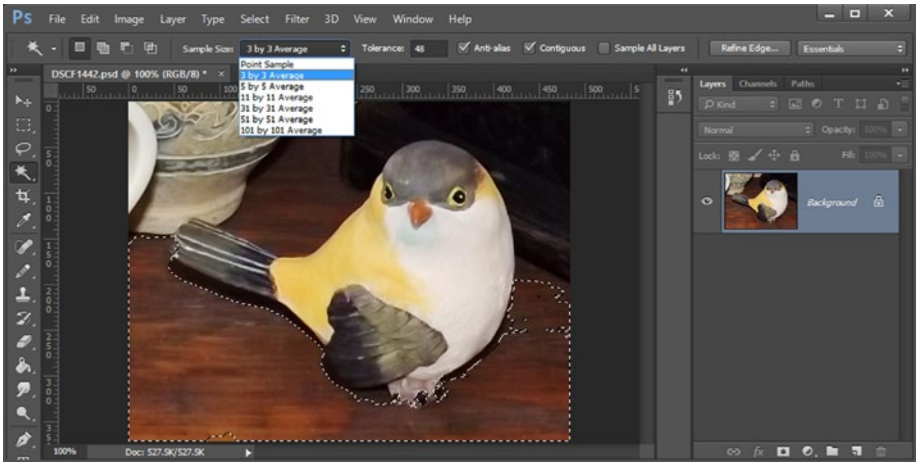

Sample Size: 3 by 3 Average
Tolerance: 48
✓ Anti-alias
✓ Contiguous
Sample All Layers
DSCF1442.psd @ 100% (RGB/8) *
Doc: 527.5K/527.5K

Figure 7-3. *Result of selecting 3 by 3 Average sample size and clicking on the wood tabletop on the screen*

Now when I click in the lower-left-hand side of the wood tabletop, I get more pixels in the selection set just as if I had increased the Tolerance to a higher value. Tolerance is how much the selection can vary from the selected color, and Sample Size allows you to give the algorithm an average from the color culled from the (matrix of) adjacent pixels. Using the two settings will allow you to fine-tune any selection, so be sure to play around with these to get a feel for how they affect the selection set that is obtained.

The other options (checkboxes) allow you to **anti-alias** a selection edge (I always use this function), and afford you a **contiguous** selection area. Anti-aliasing averages pixel colors along an edge to provide a smoother edge result. You can also sample color from other layers, if you need to, but normally, you'll be working on just one image plate at a time.

The reason I select **Contiguous** will become evident after you read the next section of this chapter, but, in brief, what this option does is to only expand the selection that you have just clicked to contiguous color pixels.

If you deselect Contiguous (Antialias and Contiguous are selected by default), you'll get selection "islands" throughout your image, wherever that sampled color and its tolerance range occur. The reason Contiguous gives you more control is that it allows you to extend the current selection. I will tell you how to do this extension in the next section, during my discussion of adding to a selection set by using (holding down) the **Shift** key modifier. This is the most popular Magic Wand work process.

Adding to a Selection Set: Using the Shift Modifier

If you have an area of uniform color (or uniform color gradient) around the subject you're trying to mask, you can theoretically tweak the sample size and tolerance until you get the numeric inputs that generate the selection set that will extract your subject matter perfectly. However, most images, such as the one we are using here, are "busier" than that, and will require more than one pixel of color sampling (click).

A solution to this is to sample different colors and add to the selection, by holding down your Shift modifier key, when you are selecting any pixels after the first pixel you select.

For this particular image, there were tan, brown, crème, pink, and black areas next to the bird that we are trying to extract, so we will need to use this technique. I used a low Tolerance—in this case, **12**—as is shown in Figure 7-4.

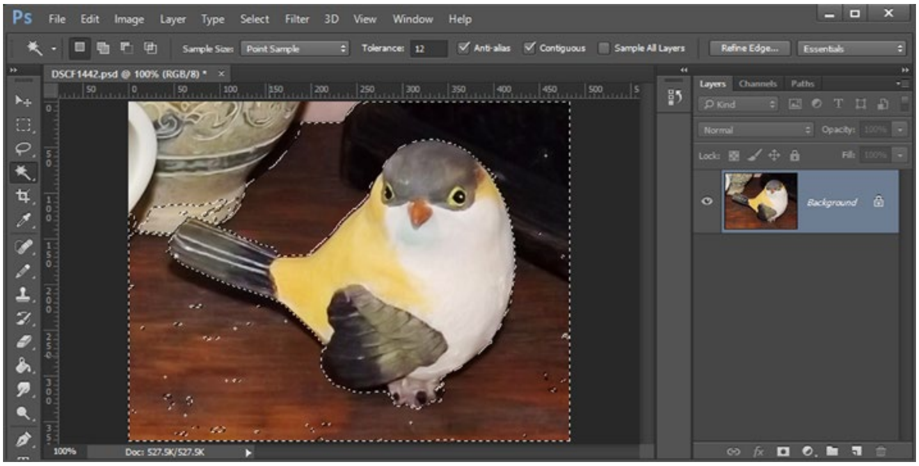

Sample Size: Point Sample   Tolerance: 12   ✓ Anti-alias   ✓ Contiguous   Sample All Layers

Figure 7-4. *The RingElement.png layer shows dimension and alpha*

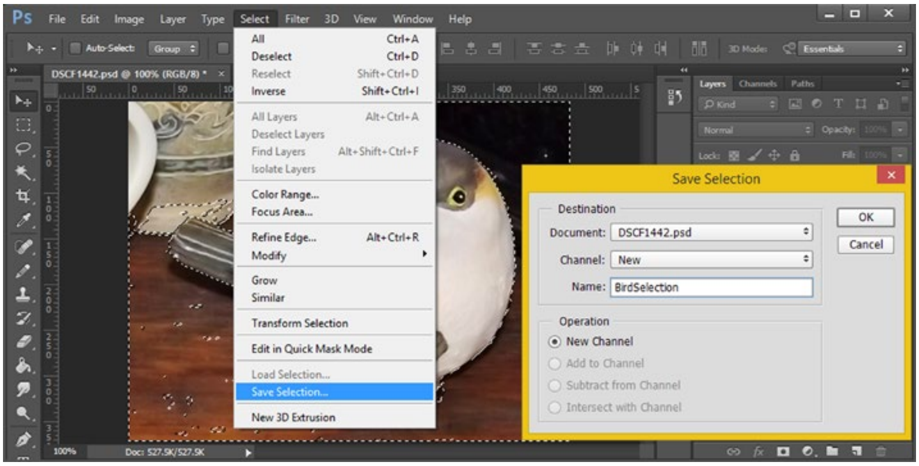
To achieve this selection set (of pixels), start a click sequence in the lower-left-hand corner of the wood tabletop, and after the first click, hold down the **Shift** key on your keyboard and click in the other unselected areas around your bird. This will add to a selection set, and your selection area will expand and refine until you finally obtain the result seen in Figure 7-4.

Once you have most of the area around the bird selected, as shown in Figure 7-4, you can then learn how to use the **alpha channel** to refine your selection, which is what we are going to cover in the next section of the chapter. It is interesting that many digital artisans never think to pull their main mask quickly using the selection tools, save the result as an alpha, and finish the task simply by using a brush, and then later on, reload the result as a selection set, using that alpha channel!

Selection Editing: Using an Alpha to Edit Selection

As you have already seen in Chapter 6 (in Figure 6-2), an alpha channel can be used to store masked objects and easily select them later on when you want to use them. Interestingly, you can also **edit** these black and white alpha channel pixels, by using any and all of the digital image editing and compositing tools that your digital image editing software offers.

Let's use this fact to our advantage in the masking work process, and use the **Save Selection** dialog shown in Figure 7-5, to turn what you have done with the Magic Wand tool so far into alpha channel data. You can probably refine this mask faster by using image editing tools than if you try to shift-click inside of progressively smaller and smaller pixel selection set areas!



Save Selection

Destination

Document: DSCF1442.psd

Channel: New

Name: BirdSelection

Operation

[✓ New Channel] Add to Channel Subtract from Channel Intersect with Channel

__Figure 7-5.__ You can save your alpha channel as something like BirdSelection in the Save Selection dialog

As shown in the figure, Photoshop's menu sequence for this process is **Select ➤ Save Selection**. You can name your alpha channel something like "BirdSelection" and then click the **OK** button to create the new alpha channel, which will contain the selection data (grayscale) that the Magic Wand tool has created for you thus far in your masking-work process.

To view this alpha channel, select your **Channels palette** tab, as is shown on the right side of Figure 7-6, and click the **BirdSelection** alpha channel **visibility icon** (it is an eye), and turn off the color channel visibility icons, using the main **RBG** channel, which can be seen at the top of your Channels palette.

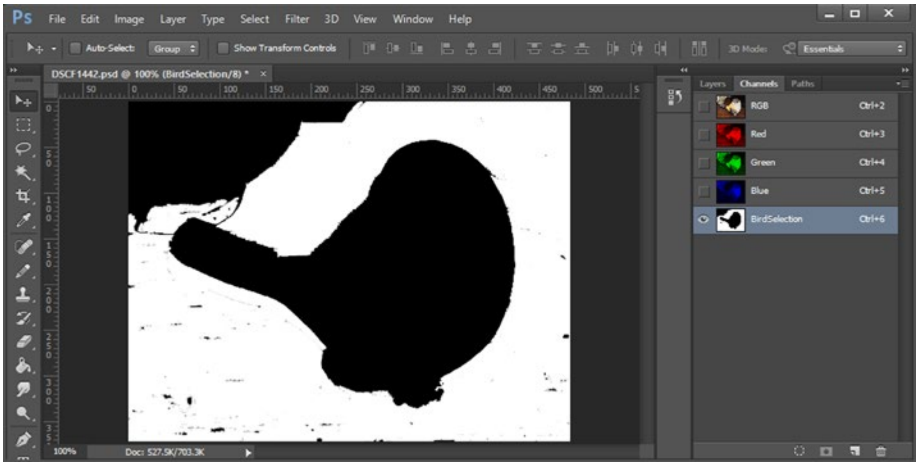
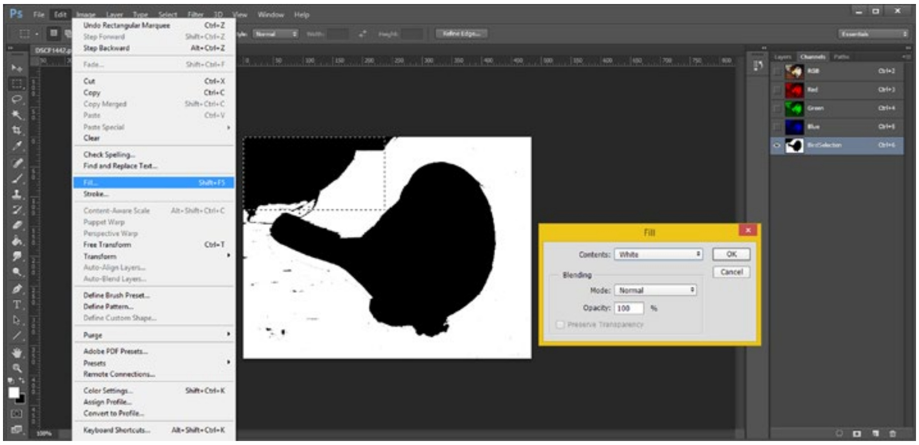


Figure 7-6. *You can select BirdSelection in the Channels palette to edit it*

This RGB channel at the top of the Channels palette is a convenience (summary) channel which when selected, will operate on the red, green and blue channels at the same time, which I'd call a summary operation, as it applies any tool, or algorithm, to the three color channels in exactly the same fashion.

In this case, that happens to be a visibility selection, or visibility deselection, more precisely, of your pixel color data, for the currently selected image composite layer.

You can use any editing tool to clean up the black areas outside of the black bird outline, including an eraser or brush tool (set to use white). The fastest way to remove large, black areas is to outline a rectangle over the area that you want to remove, which will select that area, as is shown in Figure 7-7, and then use the **Edit ➤ Fill** menu sequence, after you have set the Fill **Contents** to a color value of **White**.



Contents: White
Blending
Mode: Normal
Opacity: 100 %
Preserve Transparency

Figure 7-7. *To remove large black areas, select the content and go to Edit ➤ Fill ➤ White*

You can remove all of the black pixels on the outside of your black bird outline, using the Fill, Eraser or Brush tools, so that you create a clean delineation, between black and white pixels, as can be seen in Figure 7-8.

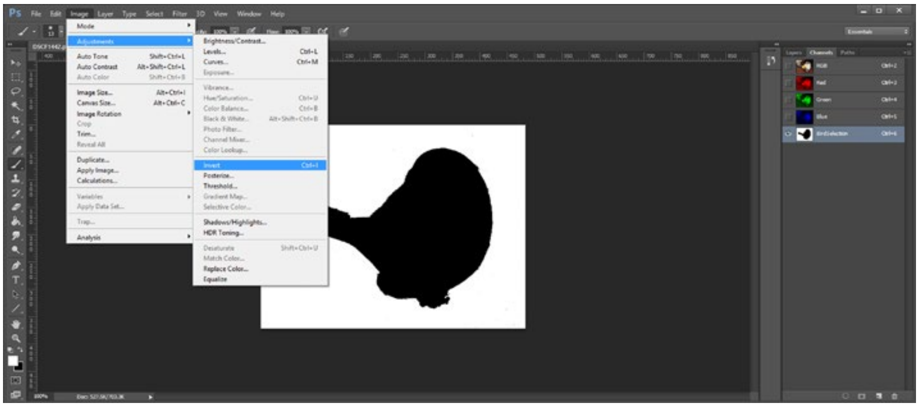


Figure 7-8. *After the outside of mask is clean, use Image ➤ Invert*

Once your general masking areas are cleaned up, you will **invert** the selection. This will then select the bird—currently, everything **except** for the bird itself is what has been selected as that is the easy way to do masking: in reverse.

Sometimes in digital imaging, the best way to approach a work process is in an exact opposite way of what you might have thought was the best approach! Masking is a perfect example of this, as you select what you don't want, and then flip (invert) the selection to grab what you do want!

The Invert function can be located by using the **Image ➤ Adjustments ➤ Invert** menu sequence, as you can see highlighted in blue at the top left portion of Figure 7-8.

Once you invert your selection, the **fine-tuning** stage of mask editing begins. This stage is when you will begin to work on the edges of your mask.

The mask's edge is its most important area, as the edge quality is what will affect the perceived quality of your mask. This is important when you use your masked object in a layer of one of your digital image composites, to combine the pixels of the mask with the rest of the digital image composite's layers.

Mask Fine-Tuning: Alpha Edge Editing Techniques

The next step in the mask creation work process is to perfect the edge between the white (opaque) and the black (transparent) areas of your mask. This is the **anti-aliasing** of the mask, and if you selected the **Anti-alias** option in the Magic Wand tool, there may already be some of this in place in the algorithmic mask. If you didn't select this option, there will only be black and white pixels along the edges of your mask. Let's use the zoom (magnifying glass) tool, and see if there are any **gray** pixels. These gray pixels will provide a mask with "real-time" color blending (using anti-aliasing) between the subject matter (bird), and any background image that you may wish to composite the object on top of later on, since these define **transparency**.

As you can see in Figure 7-9, the mask has been inverted and there are indeed gray pixels along the edges providing this antialiasing function. To create this image, I zoomed into the slightly curved area on a “clean” edge along the bottom portion of the bird’s tail. As you can see in the tab, I zoomed in 700% so that you could see the edge pixels and the anti-aliasing.

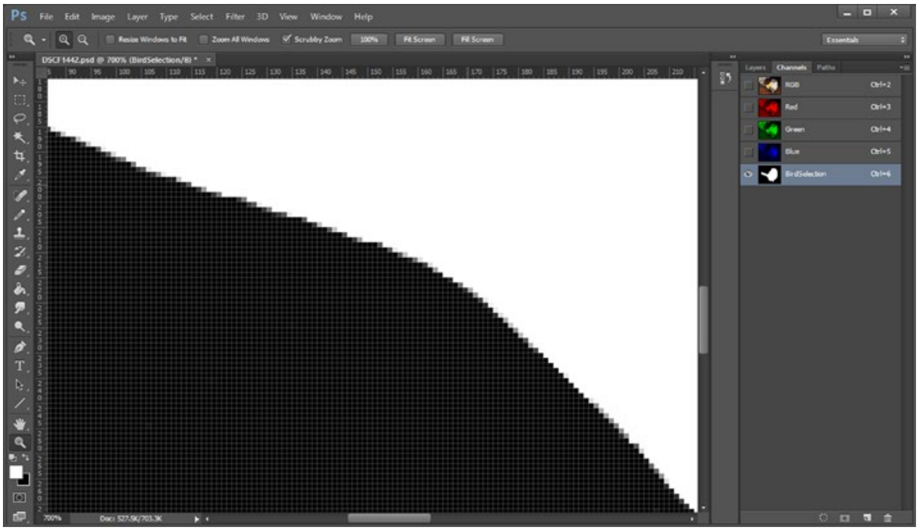
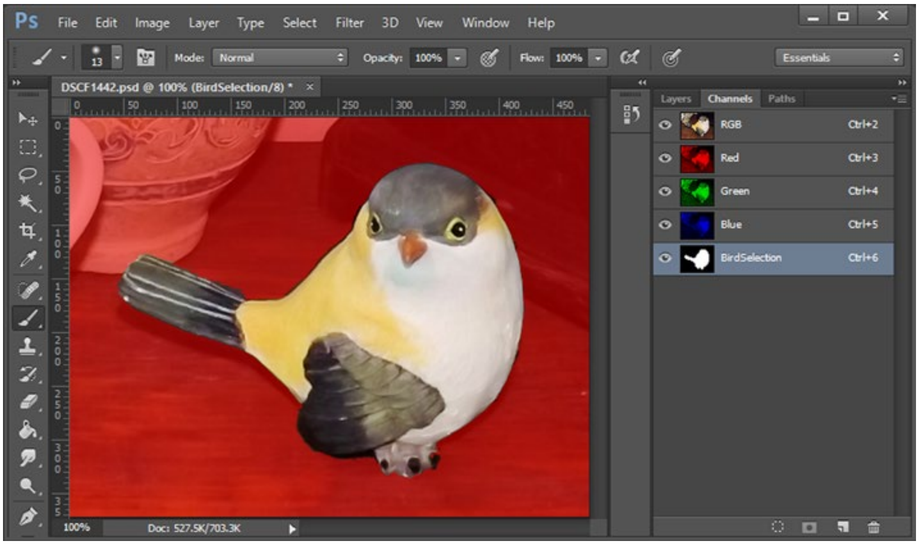


Figure 7-9. You can zoom into the edge of the mask to see antialiasing

Now, let’s zoom back out, by using the **View ➤ 100%** menu sequence, and turn on the **Mask Overlay** feature in Photoshop CS. This is done by clicking the **Eye** icon, next to the RGB channel.

This will tint a red value over the transparent areas in the mask, as is seen in Figure 7-10. The tint will allow you to edit the mask while also seeing the image that you are masking, so that the image that you are masking can be used as a guide.



Mode: Normal Opacity: 100% Flow: 100%

DSCF1442.psd @ 100% (BirdSelection/8) *

RGB	Ctrl+2
Red	Ctrl+3
Green	Ctrl+4
Blue	Ctrl+5
BirdSelection	Ctrl+6

100% Doc: 527.5K/703.3K

Figure 7-10. *Select the RGB channel's Eye icon to turn on the mask*

As you can see in Figure 7-11, I used a Brush tool, with a 13-pixel fuzzy edge, and zoomed in 400 percent to fix the top edge and back of the bird's tail, which, as you can see, needed some work, in terms of fine-tuning the edge conformance of this mask to the bird object.

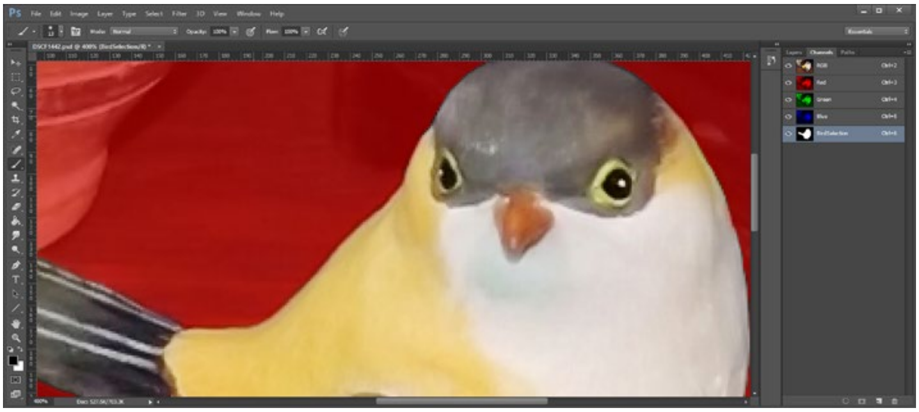


Figure 7-11. *To edit the edge of the bird's tail, use the 13-pixel fuzzy brush and zoom in*

You can turn this red masking color aid on and off using the RGB channel Eye icon. You will do this as you work, so that you can see the actual mask itself.

You can also work in black-and-white mode, without this mask overlay, to make sure the mask is all black and all white, with gray anti-aliasing pixels along the edges between the two colors.

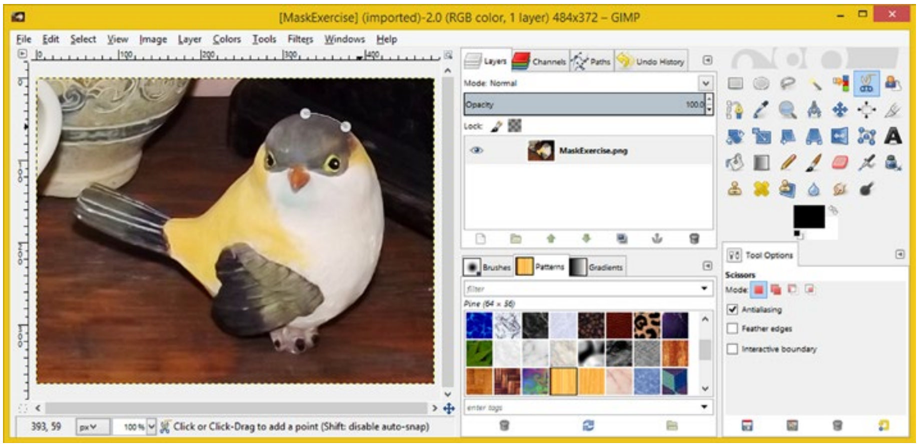
This is typically how masking artists work; they use the red mask overlay guide to get an image of how the mask is progressing, and then turn it off to work with the high contrast black and white representation as it is easier to see what you are doing in areas of the mask other than the edges.

You should spend as much time perfecting the mask as you need to get the result you want. Keep in mind, like everything else, it takes a lot of practice to become a professional compositor! It's important to note that although we used a 484-by-372 resolution sample image to look at this masking process, using a **higher resolution** image for your masking process will almost always yield a higher quality image compositing result. The individual pixel size, relative to the entire image, is a smaller ratio (percent), so edges and anti-aliasing is finer.

The equivalent of the Magic Wand tool in GIMP is the **Fuzzy Select** tool. Since it works in an extremely similar way in both programs, I have decided to instead show you how to pull masks in GIMP by using another innovative algorithm that takes a different approach. This algorithm can be implemented by using the **Scissors Select** tool.

GIMP Scissors Tool: Intelligent Masking Algorithm

To access the Scissors Select tool, open the **MaskExercise.png** PNG24 digital image file in **GIMP** using the **File ➤ Open** menu sequence. Then, select the **Scissors** tool icon, shown on the top right side of the screen in Figure 7-12, and click the edge of the object you want to extract (cut out with these digital scissors) to start the outline you are going to be laying down. When you click a second point on the object, the Scissors Select tool will apply its **edge detection algorithm**. This can be seen at the top of the bird's head in Figure 7-12.


[MaskExercise] (imported)-2.0 (RGB color, 1 layer) 484x372 – GIMP

393, 59 px 100 % Click or Click-Drag to add a point (Shift: disable auto-snap)

Tool Options
Scissors
Mode:
[✓] Antialiasing
[ ] Feather edges
[ ] Interactive boundary

Figure 7-12. *To extract an object from an image, open the image, select the Scissor tool, and click on its edge. Clicking it a second time enables edge detection*

If you select the **Interactive Boundary** option, as shown, checked off in the lower-left-hand side of the screen in Figure 7-13, an edge detection algorithm will **recalculate in real time** as you adjust the points on your selection outline to refine the mask nodes.

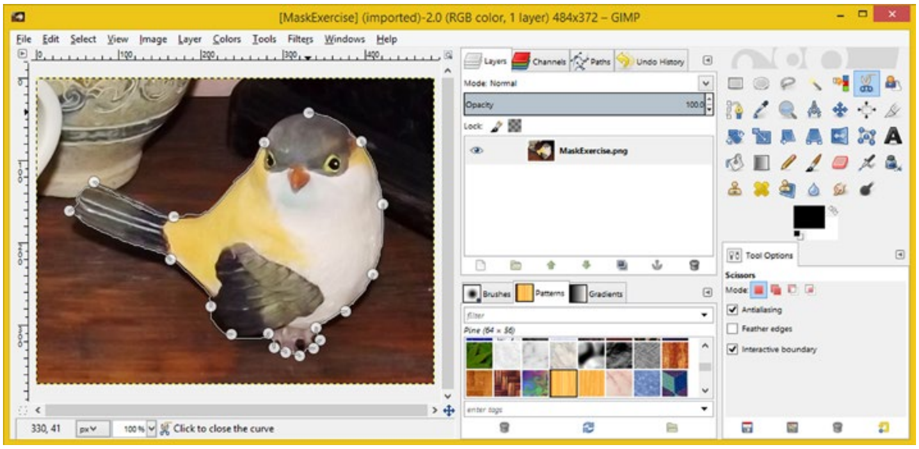

[MaskExercise] (imported)-2.0 (RGB color, 1 layer) 484x372 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Mode: Normal
Opacity 100.0
Lock:
MaskExercise.png
Brushes Patterns Gradients
Pine (64 × 56)
Tool Options
Scissors
Mode:
Antialiasing
Feather edges
Interactive boundary
330, 41 px 100 %
Click to close the curve

Figure 7-13. Click along the edge of your object to create an outline

The Scissors tool is like an intelligent spline that automatically looks behind itself and conforms its curvature as it travels between points (I call them nodes for the tool) and sets the tension handles (the curvature in and out of each node). You can even refine node positions to adjust the curves!

Adjusting the Scissors Tool: Intelligent Spline Nodes

I have found that I get better edge detection algorithm results when I put nodes at color changes in the object, as shown in Figure 7-14, in the gray and yellow areas of the bird. You can move node positions by dragging them, at any time during the process of laying down the curve, including after it is closed. The curve is closed by clicking on the first node that you placed, which, in this case, is at the top of the bird's head. As you can see in the figure, the line between nodes straightens as you drag the node to reposition it, with the new curve fit algorithm being calculated when you drop or release a node in a new position.

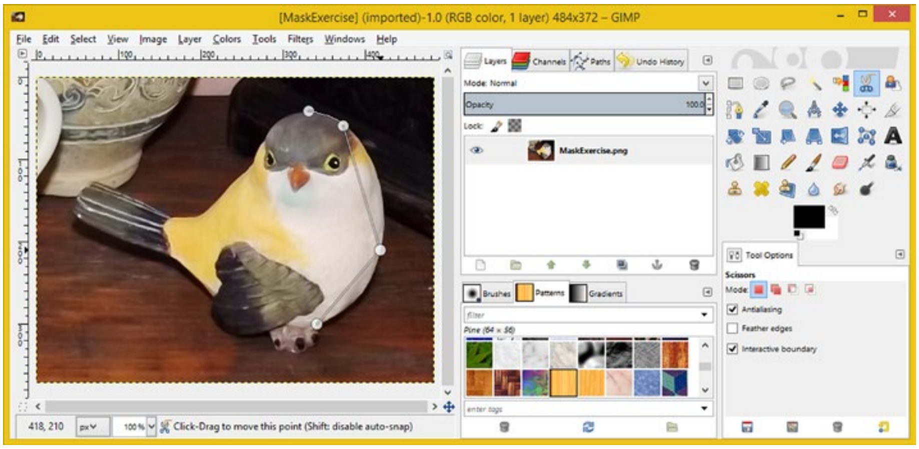

[MaskExercise] (imported)-1.0 (RGB color, 1 layer) 484x372 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Mode: Normal
Opacity 100.0
Lock:
MaskExercise.png
Brushes Patterns Gradients
Pine (64 × 56)
Tool Options
Scissors
Mode:
Antialiasing
Feather edges
Interactive boundary
418, 210 px 100 % Click-Drag to move this point (Shift: disable auto-snap)

Figure 7-14. *You can move the nodes any time to get a better result*

If you reposition your right-hand node in the middle of the bird's chest, you get 100 percent-superior results from the intelligent-curve-placement algorithm as compared with any other edge placement (try it for yourself, using the MaskExercise.png file that's included with the book). The results are shown in Figure 7-15, with perfect edge detection.

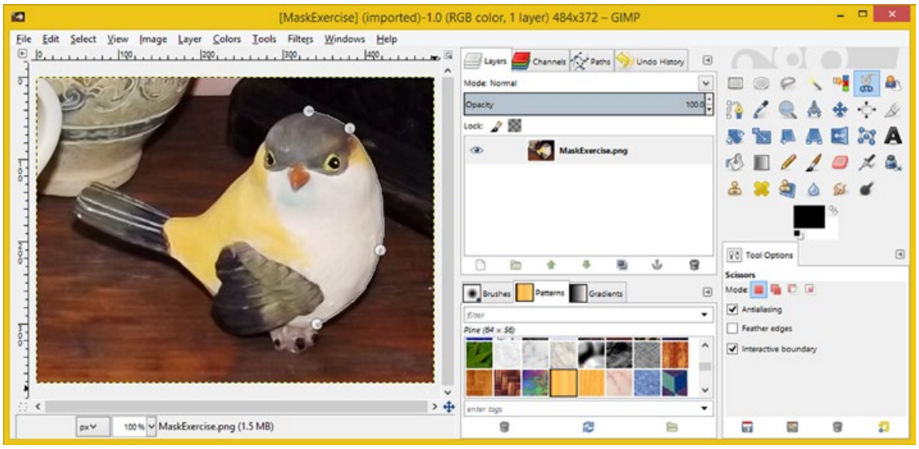
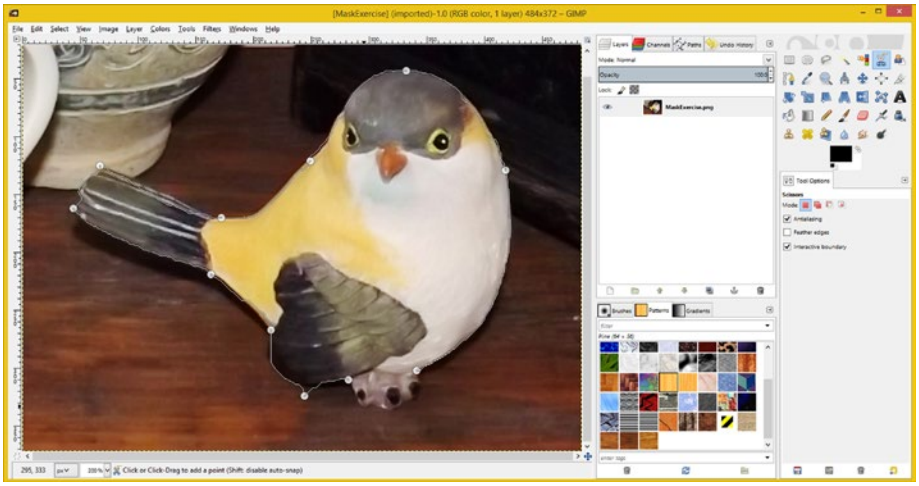

[MaskExercise] (imported)-1.0 (RGB color, 1 layer) 484x372 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Mode: Normal
Opacity 100.0
Lock:
MaskExercise.png
Brushes Patterns Gradients
Pine (64 × 56)
Tool Options
Scissors
Mode:
Antialiasing
Feather edges
Interactive boundary
px 100 % MaskExercise.png (1.5 MB)

Figure 7-15. *This node placement has a better edge-match result*

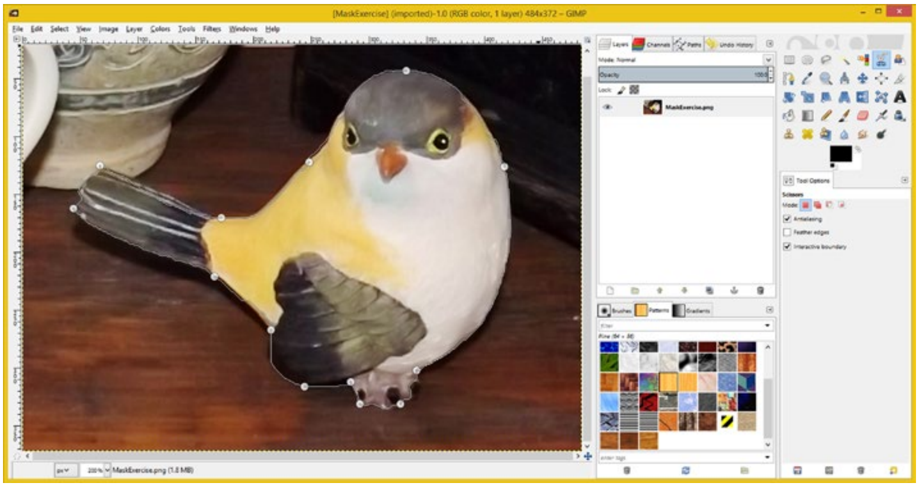
In Figure 7-16, I have zoomed in 200 percent, so I can see the talons more clearly for the node placement at the bottom of the bird image. As you can see, there are a few nodes that I had to slightly reposition to get a better edge fit from the Scissor tool algorithm, specifically the one at the tip of the wing (compare Figure 7-16 with Figure 7-17).


[MaskExercise] (imported)-1.0 (RGB color, 1 layer) 484x372 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Mode: Normal
Opacity 100.0
Lock:
MaskExercise.png
Brushes Patterns Gradients
Tool Options
Scissors
Mode:

295, 333 px 200 % Click or Click-Drag to add a point (Shift: disable auto-snap)

- [✓] Antialiasing
- [ ] Feather edges
- [✓] Interactive boundary

***Figure 7-16.** Position easier nodes first, save details for the end*


[MaskExercise] (imported)-1.0 (RGB color, 1 layer) 484x372 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Mode: Normal
Opacity 100.0
Lock:
MaskExercise.png
Brushes Patterns Gradients
Tool Options
Scissors
Mode:

px 200 % MaskExercise.png (1.8 MB)

- [✓] Antialiasing
- [ ] Feather edges
- [✓] Interactive boundary

***Figure 7-17.** To close the spline, click on the first node you laid down*

Also, notice that in Figures 7-14 through 7-16, I have started the outline next to the bird's talons, as this is the detailed area for your mask, and so I wanted to work on this part last. So my first point is to the right of the talons, and I position nodes counter-clockwise, ending to the left of the talons, so the last nodes that I lay down are the talon detail.

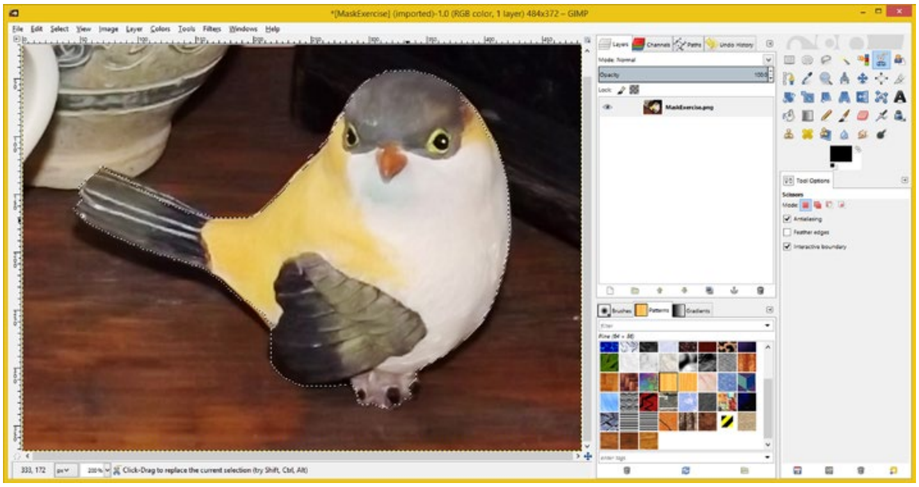
The reason for this is that I can zoom in to the talons area last, once the majority of the mask area has been outlined using the Scissors Select tool.

It's important to note that if you click another GIMP tool, the one you are using will **abort**, and so if you want to zoom, you will have to use the **View ➤ Zoom ➤ Zoom In** menuing sequence (or **View ➤ Zoom ➤ Zoom Out**) to keep this Scissors tool active. It's useful to know both the menu sequences, as well as the tool icons, for both Photoshop and GIMP for this reason.

As you can see in Figure 7-17, the Scissors algorithm is doing a fairly good job on the talon area, except for the claws, which I will have to deal with manually, once I convert this to a selection, and save it as an alpha (for editing).

To close the selection curve completely, position your pointer over the first node you placed, and when it changes to the **close curve pointer**, click and the curve will be closed. It is important to note that you can still adjust (move) the nodes when the selection curve has been closed. To turn the selection curve into a selection area, click the inside of the curve.

As you can see in Figure 7-18, I've turned the curve into a selection, and can now use the same work process that I did in Photoshop to save the selection set and edit it in a channel. The menu sequence to do this in GIMP is **Select ➤ Save to Channel**. Then you can toggle the Quick Mask (red color guide) mode with **Select ➤ Toggle Quick Mask**.


*[MaskExercise] (imported)-1.0 (RGB color, 1 layer) 484x372 – GIMP

Click-Drag to replace the current selection (try Shift, Ctrl, Alt)

Figure 7-18. *Click inside of the closed spline to create your final selection*

As you can see in Figure 7-19, saving to a channel will allow you to reopen the channel and edit further using the **Brush tool** (selected) with black and white colors, which can be **toggled** using that **curved arrow** between the color swatches, under your tool icons.

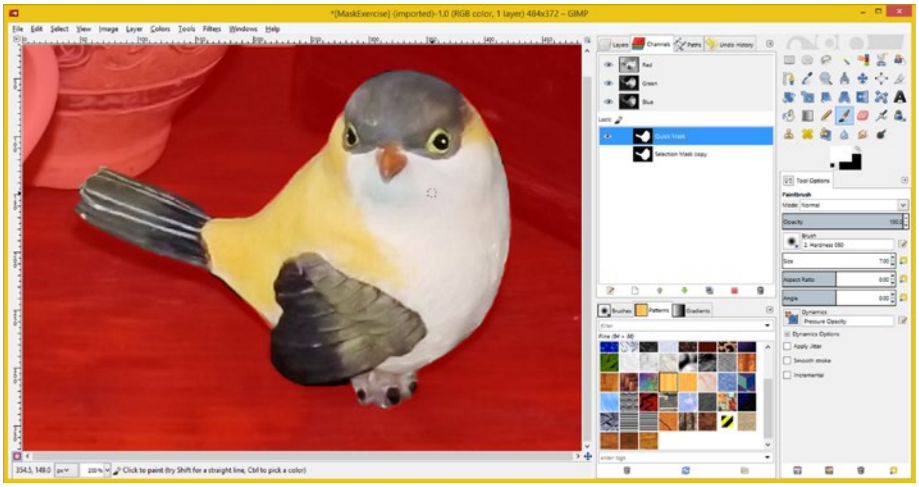

*[MaskExercise] (imported)-1.0 (RGB color, 1 layer) 484x372 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Red
Green
Blue
Quick Mask
Selection Mask copy
Tool Options
Paintbrush
Mode Normal
Opacity
Brush
Size
Aspect Ratio
Angle
Dynamics
Pressure Opacity
Apply Jitter
Smooth stroke
Incremental
Brushes Patterns Gradients
Click to paint (try Shift for a straight line, Ctrl to pick a color)

Figure 7-19. *Using Quick Mask mode in GIMP to refine your mask*

As you can see in Figure 7-19, I used the Brush tool to make sure the right toe was included inside of the mask area, and cleaned up the other areas a little bit.

This Scissors Select tool really gets you to the point where 99% of the masking work is done by your node positioning and the tool algorithm, which is pretty amazing if you think about it, especially since GIMP is free for commercial use. Of course, you can spend as much time as you want on edge clean-up and perfect the mask, since you have the XCF file now. Practice makes perfect.

In the next chapter, we'll be taking a look at using the **Layers palette**.

Summary

In this seventh chapter, we reviewed the most common ways to **pull a mask** using Photoshop (**Magic Wand** tool) and GIMP (**Scissors Select** tool). You learned about how to use the Photoshop Magic Wand and its **Sample Size** and **Tolerance** settings, and how to save the selection to an alpha channel and edit it further to perfect your end result. In GIMP, you learned about the Scissors Select tool, which takes a different approach of using a powerful algorithm to create a spline that curve fits to any of the object's edges, which you direct it to by laying down **nodes** on the perimeter of the object. We also went over saving this result to an alpha channel and further editing the mask, just like you did in Photoshop CS6.

In Chapter 8, you will learn more about digital image compositing Layers and how to work with their functions, which are accessed using the Layer palette in both Photoshop CS6 and GIMP 2.8.14.

CHAPTER 8

The Organization of Digital Imaging: Using Layers

Now that you have learned how to mask objects for use in your digital image compositing pipelines, the next logical step is to learn how to leverage the **Layers palette** to organize your digital image compositing projects. Layers allow you to isolate masked objects as well as to apply imaging special effects to individual layers and your digital image backplate—that is, the foundational background image upon which your digital image composite is based.

Originally, in early digital imaging software revisions, layers were just used to isolate masked objects, in the same way that vellum on an overhead projector allows you to add to your presentations. The layer was solely designed to hold image compositing components. At a certain point in the development cycle, developers started adding other cool features to layers, making them an order of magnitude more powerful.

I have included this chapter covering the Layers palette since at this point, using layers will enable you to do a whole lot more than just organizing your digital image compositing! Indeed, one could write an entire book just on using the Layers palette, since it is now packed with so many features, as you will see during the remainder of this book.

Photoshop and GIMP both support a plethora of cool layer organization tools, which can be used to organize very complex digital image compositing pipelines and create special effects constructs, as you will see over the next five chapters.

Because Photoshop is especially adept when it comes to using layers, allowing you to implement all manner of imaging matriculations and special effects, we will spend a bit more time looking this program's layer features, and then take a gander at the impressive features that GIMP 2.8.14 supports.

Hopefully, GIMP 3.0 will add some of the powerful layer features that Photoshop CS6 currently offers, specifically the adjustment layer concept that gives you the ability to perform a digital image pixel adjustment, in the form of an adjustment layer, complete with non-destructive settings that can later be changed by editing the adjustment layer itself!

You'll be learning all about Photoshop adjustment layers during the course of this chapter. Let's get started—we have a lot to cover! Layers and layers of relevant imaging information in fact, and no pun intended!

Layer Differentiation: Photoshop vs. GIMP

One of the areas where Photoshop and GIMP differ the most as far as features are concerned is in the area of Layers. If you are really serious about digital image editing or compositing, chances are you will pay the $10 per month to get Photoshop CS, or have one of your clients pay the $120 annual stipend as part of your project contract. GIMP 2.8 has enough layer features to perform professional digital image compositing, but Photoshop CS has some special effects application features that make tweaking results or changing something at the client's request significantly easier. GIMP 3.0 will be adding significant power and features in 2016 after these are tested out in version 2.9.

Photoshop Layers: Organization and Effects

As you will see if you drop down Photoshop's **Layer** menu, shown in Figure 8-1, there are a plethora of layer organization, masking, and digital imaging algorithms (called "adjustments"). I will be focusing on the **New** and **New Adjustment Layer** submenus in this chapter, as there is not enough time or pages to cover everything that Photoshop offers via its Layers palette. It's another one of those topics that could make up its own book, as there are fill layers, video layers, Smart objects, layer styles, and even a feature that does your web slicing for you!

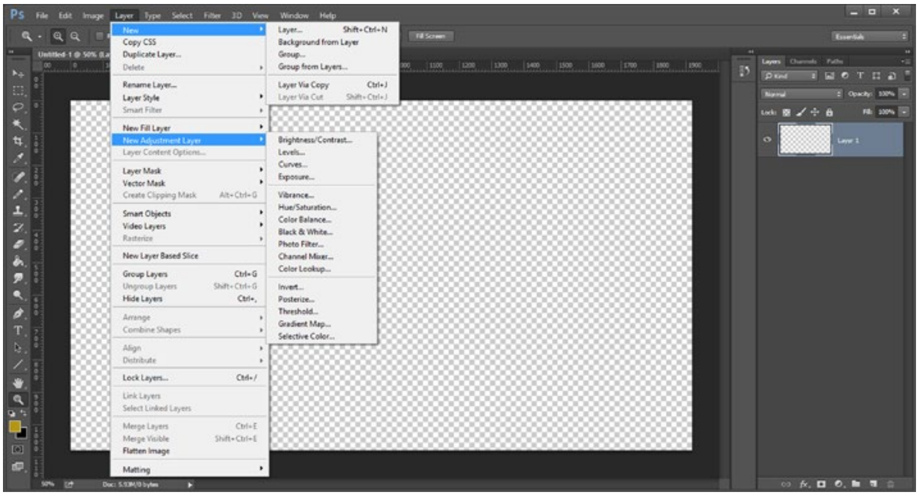


Figure 8-1. *Photoshop Layer menu offers many advanced features*

We will be looking at those layer features that are most often used in your digital image compositing work process over the course of this chapter. These include a new layer creation, duplicating layers, grouping layers, naming layers, merging layers, adjustment layers, and deleting layers. I'll show you how to access a number of these commands in multiple places in the software, both in this chapter as well as in subsequent chapters during the rest of the book.

To get the empty **Layer 1** seen in Figure 8-1, I opened up Photoshop and used the **File ➤ New** menu sequence. This is how to create your empty digital image compositing pipeline.

I will show you the **Layer** menu in GIMP next, so that we cover both software packages, which I am trying to do in each chapter over the course of this book.

GIMP Layers: Organization and Masking

To access the digital image compositing layers in GIMP, open the GIMP program and use the **File ➤ New** menu sequence to create a new empty digital image compositing document. To see how layer features compare with Photoshop, drop down GIMP's **Layer** menu and **Mask** submenu, as shown in Figure 8-2. As you will see, GIMP has far fewer layer features than Photoshop CS6, but the core features needed for image compositing are all there. These are some of the layer features we will be covering in the chapter, as well as in subsequent chapters as we build on your knowledge of the software and learn new features.

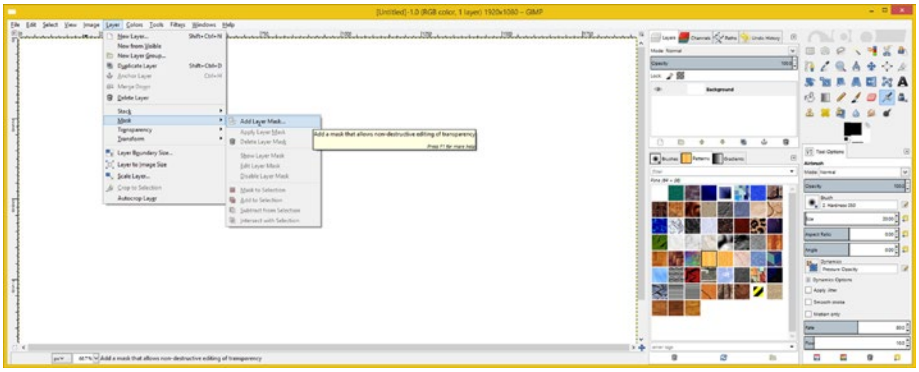


Figure 8-2. *GIMP Layer menu and its Mask sub-menu*

Just like in Photoshop, these GIMP layer features can be accessed by using the **Layer** menu, or the **context-sensitive** menu system, which can be accessed by right-clicking on the layer itself, as a short-cut to make your work flow proceed faster.

As you can see in Figure 8-2, GIMP allows you to create a **New Layer** or **New Layer Group** as well as to duplicate layers, merge layers down (together), anchor layers, and delete layers. We'll be looking at compositing work processes that use these functions during the rest of the book, so you will see how they are used, even if we don't specifically cover them during this particular chapter.

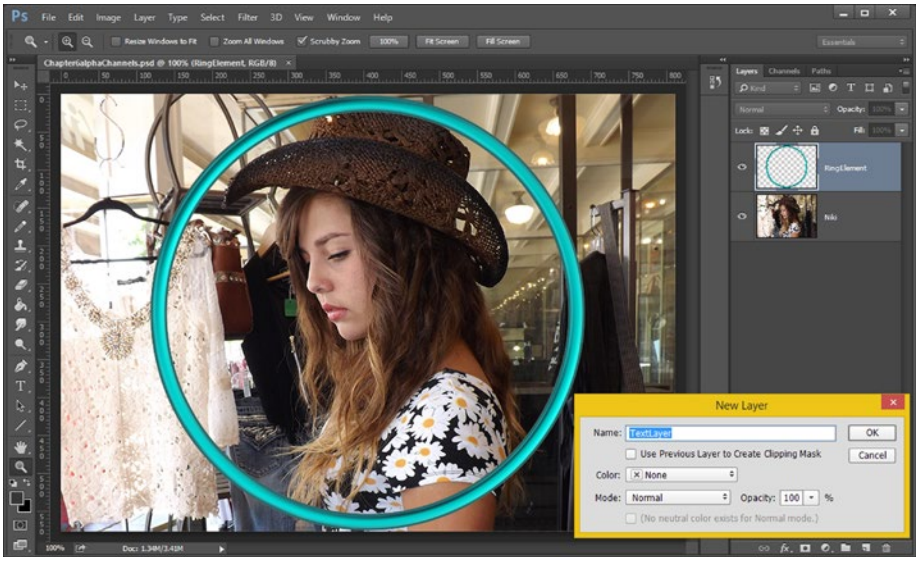
You can also **Scale** your layer in GIMP, or change the **Layer Boundary Size**, to make the layer larger than the digital image it contains. You can also make the layer smaller than the image if the layer boundary is larger than its image, by using a **Layer to Image Size** option. There is also the **New from Visible** option, which merges layers marked as visible (where the eye icon is displayed) into a single (new) layer located at the top of the layer compositing stack so that it is at the top of your compositing rendering preview.

As you can see in Figure 8-2, GIMP layers each use white as a background color. GIMP has a **Layer ➤ Mask** submenu allowing you to **Add Layer Mask** to get the transparency that you get as a default in Photoshop. There are also other layer mask functions (seen in gray) that become enabled once a GIMP layer has a mask associated with it. They include **Apply** (render to pixel), **Show**, **Delete**, **Edit**, and **Disable** (turn off temporarily) **Layer Mask**.

As you can see, GIMP includes all the foundational layer support that you need to create and organize complicated image composites. Since the program's composites work in much the same way as they do in Photoshop, let's take a look at layer compositing in Photoshop first and look at GIMP's layers later.

Using Layers in Photoshop: Compositing

Let's continue building the composite we started in Chapter 6. Begin by opening the `Chapter6alphaChannels.psd` file and then use the **Layer ➤ New Layer** menu sequence, shown in Figure 8-1, to access the **New Layer** dialog shown in Figure 8-3. Name the layer **TextLayer**, select **Color: None** (transparency) and **Mode: Normal**, and finally, **Opacity: 100%**. You will be learning about modes in Chapter 9. I wanted to show you how to add **Text objects** to your composite as a part of this exercise, as it has many practical applications.



New Layer

Name: TextLayer

☐ Use Previous Layer to Create Clipping Mask

Color: None

Mode: Normal   Opacity: 100 %

☐ (No neutral color exists for Normal mode.)

__Figure 8-3.__ Use Layer ➤ New Layer menu sequence and then Name: TextLayer

We are now ready to take a look at the Photoshop CS **Text tool**, which is accessed using the **capital** *T* in the lower part of the icon toolbar on the left side of the screen. You have already used the Magic Wand tool for mask creation; now you are going to explore some of the other tools.

■ **Tip** If you click any of these icons on the Photoshop toolbar at the left side of the screen, and **hold** your mouse button down, a pop-up will appear, that shows other closely related tools that you can select.

Adding a Text Object: Using the Vertical Text Tool

In order to add a text object, click and hold the letter *T* icon from the left toolbar, and select **Vertical Type Tool** from the drop-down menu that will appear, as shown in Figure 8-4. Make sure your TextLayer is still selected, and then click the top right in your composite and type "**COMPOSITING**" using a **gold** color and **54-point Arial Black** font. As you can see at the top of the figure, you can set the font, style, point size, anti-aliasing, justification, and color in the **Text Tool Options toolbar** underneath your top menus.

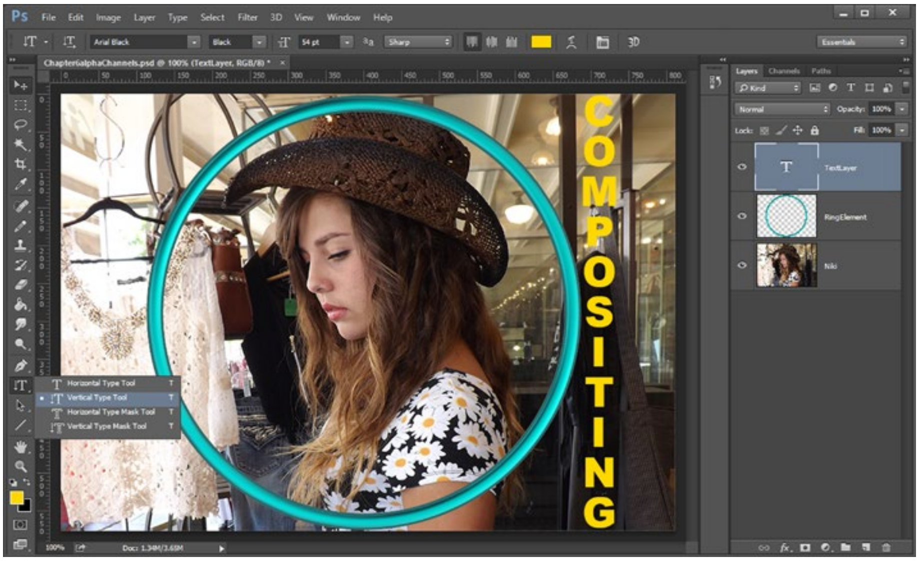


Figure 8-4. *In order to add a text object, click the letter T icon and select the Vertical Type Tool, and then type, "COMPOSITING"*

To fine-tune the position of the Text object once you're done, click the **Move** tool (the top icon on the toolbar) and use your **arrow keys** to move it by single pixels, or drag the object with your mouse.

Special Effects Layers: Adding Effects to the Text

Next, let's add some **drop shadowing** to your Text object to make it "pop" and add a bit of professionalism to your image composite. To do so, right-click the **TextLayer** and select the **Blending Options** in the drop-down men that appears on the right-hand side of the screen, as is shown in Figure 8-5.

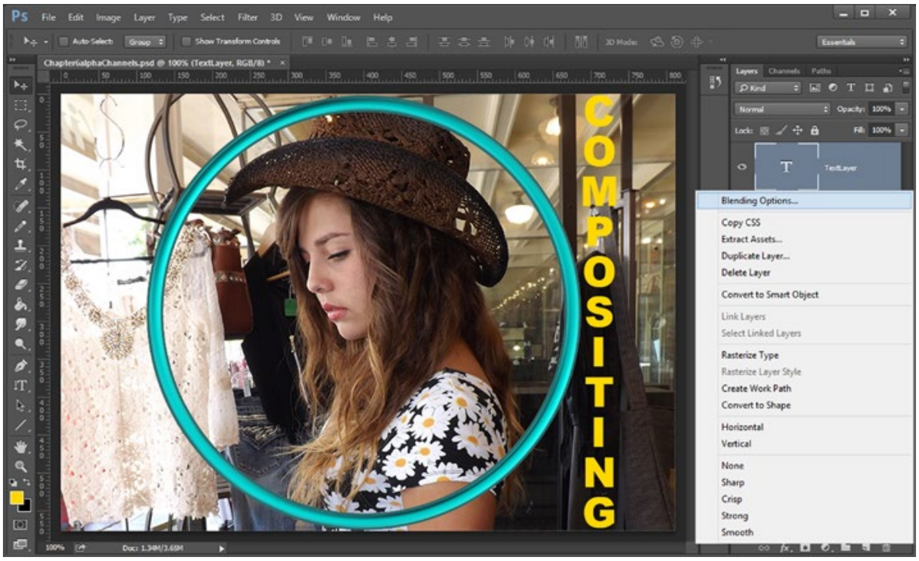
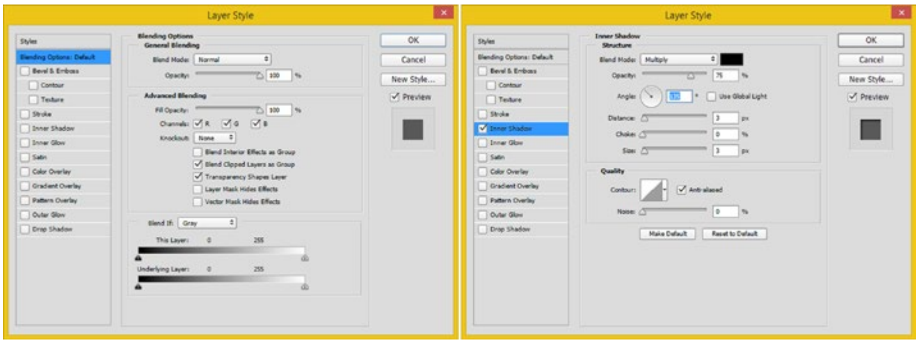


***Figure 8-5.** To add drop shadowing, right-click the TextLayer and select Blending Options from the drop-down menu*

I like to use an **Inner Shadow**, as selected on the left side of the screen of Figure 8-6, to achieve shading on the text, giving it the effect that of being cut into the image.



Blending Options
General Blending
Blend Mode: Normal
Opacity: 100 %
Advanced Blending
Fill Opacity: 100 %
Channels: ✓ R ✓ G ✓ B
Knockout: None
Blend Interior Effects as Group
✓ Blend Clipped Layers as Group
✓ Transparency Shapes Layer
Layer Mask Hides Effects
Vector Mask Hides Effects
Blend If: Gray
This Layer: 0 255
Underlying Layer: 0 255

Inner Shadow
Structure
Blend Mode: Multiply
Opacity: 75 %
Angle: [illegible] ° Use Global Light
Distance: 3 px
Choke: 0 %
Size: 3 px
Quality
Contour: ✓ Anti-aliased
Noise: 0 %

***Figure 8-6.** To shade your text, select the Inner Shadow option and set Angle and Size*

To do this, I set a **135**-degree **angle**, **3 pixels** for both **Size** and **Distance**, an **Opacity** of **75%**, the **Multiply Blend Mode**, zero pixels of Choke, and an **Anti-aliased Contour**.

If you make sure the Preview option on the right side of the dialog is selected, you can play around with these settings and their effects will be rendered, in real-time, on your Text object, so that you will be able to see your result and refine it in this settings dialog until it is exactly what you wanted.

Once you click **OK**, you will see your resulting Inner Shadow effect, as shown in Figure 8-7, and notice that Photoshop is using a new sublayer to encapsulate the settings. This is the use of layers that I was referring to earlier when I discussed using a layer paradigm to contain special-effects settings and visibility. As you can see, you can edit these settings at any time (try it!) or preview the effect using the layer visibility (eye) icons.

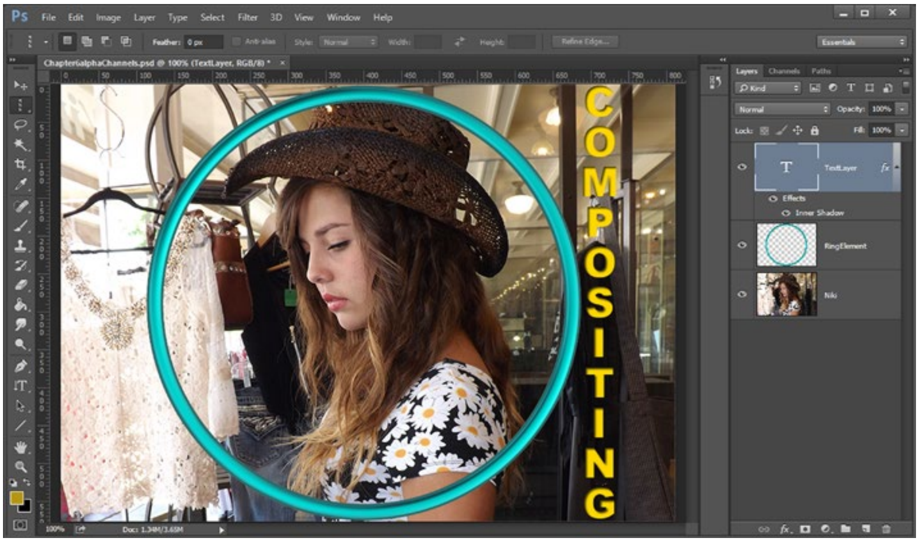


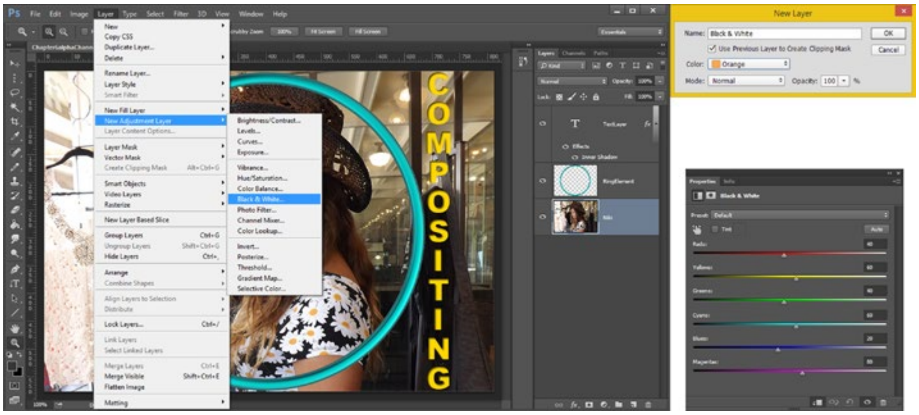
Figure 8-7. *To turn the Inner Shadow effect on and off, toggle the eye icons*

If you click the eye icons next to the Inner Shadow effects, you will see that you can turn just this effect on and off. This enhances the **modal** nature of the digital image compositing software and allows everything to be composed as separate items, or what I like to call "moves," within the layer stack, or "compositing pipeline."

Adjustment Layer: Adjusting Image Characteristics

Let's now take a look at a unique type of layer in Photoshop called the **Adjustment layer**. An "adjustment" in Photoshop is a change that a photographer would make to the digital imagery. This might include shifting the color, or increasing the contrast, or turning a color image into a black-and-white photograph, which we will be doing next by using one of the adjustment layers. These layers, in essence, allow for what can be termed as "non-destructive" editing, as the adjustment is held separately, inside of the layer itself, and is never permanently applied to your digital image data.

To create a black-and-white adjustment layer, select the Niki layer and using the **Layer ➤ New Adjustment Layer ➤ Black & White** menu sequence to open a **New Layer** dialog, as seen in Figure 8-8.



New Layer

Name: Black & White

☑ Use Previous Layer to Create Clipping Mask

Color: Orange

Mode: Normal Opacity: 100 %

__Figure 8-8.__ You can add a Black & White adjustment layer to Niki layer

Then, select the **Use Previous Layer to Create a Clipping Mask** option in the **New Adjustment Layer** Dialog, which will then attach this adjustment layer to the Niki layer.

I also selected an **Orange** color, to show you how you can color code your layers, as another way to organize what you're doing. Throughout the book I will color adjustment layers with the Orange color to make them stand out to the readers.

Once you click **OK**, and create an adjustment layer, Niki will be turned into a **black-and-white** image, as can be seen in Figure 8-9. You can click on the eye icon to toggle this effect (image desaturation adjustment) and turn it on and off at will.

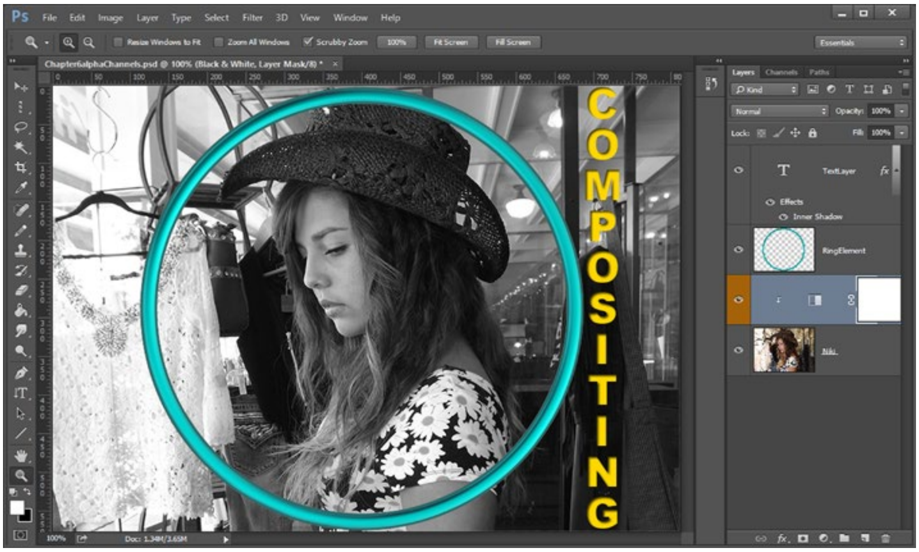


Figure 8-9. *An adjustment layer displays the image adjustment data that you have input*

Let's proceed to take a look at some of these same compositing features in GIMP, to see how that software matches up as far as creating a composite is concerned. Like the Photoshop image composite, we will include an image, object, vertical text, and a special effect in the GIMP image.

Using Layers in GIMP: Basic Compositing

Let's use a right-click, context-sensitive menu work process to add the new layer in GIMP. Right-click the aqua RingElement layer, as seen in Figure 8-10, and select **New Layer** to access a New Layer dialog. Name the layer **TextLayer** and set a Layer Fill Type of **Transparency**. This dialog will set a size for the layer that is equal to the document size settings. This is the equivalent to the first step you took in Photoshop to create a new layer.

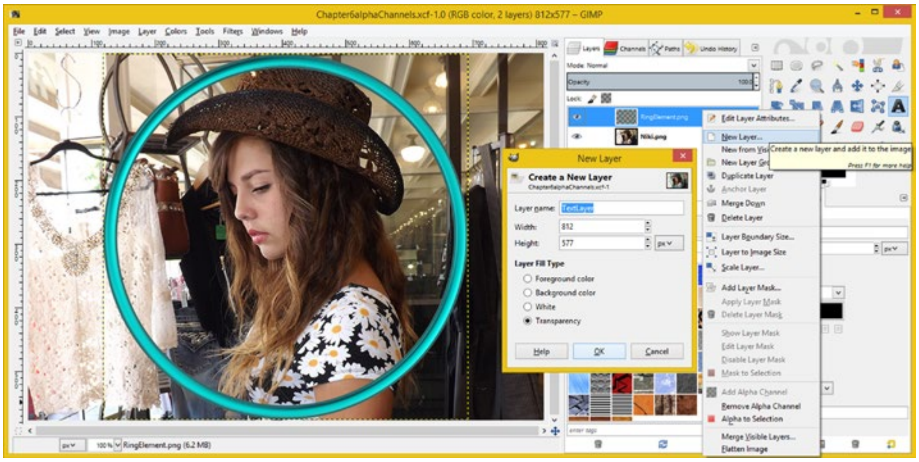


Figure 8-10. *To add a new layer in GIMP, right-click the RingElement layer and select New Layer*

The next step you performed in Photoshop was to add text by clicking the *T* icon (GIMP uses an *A*), selecting the Vertical Type tool, and then entering the word "COMPOSITING." Since GIMP doesn't have a Vertical Text tool, to get a similar result, you have to hit the **Enter** key after you type in each letter. As you can see in Figure 8-11, you set your gold color using a similar **color picker** dialog. You can set the font, style, antialiasing, and line spacing under the Tool Options on the lower-right side of the screen.

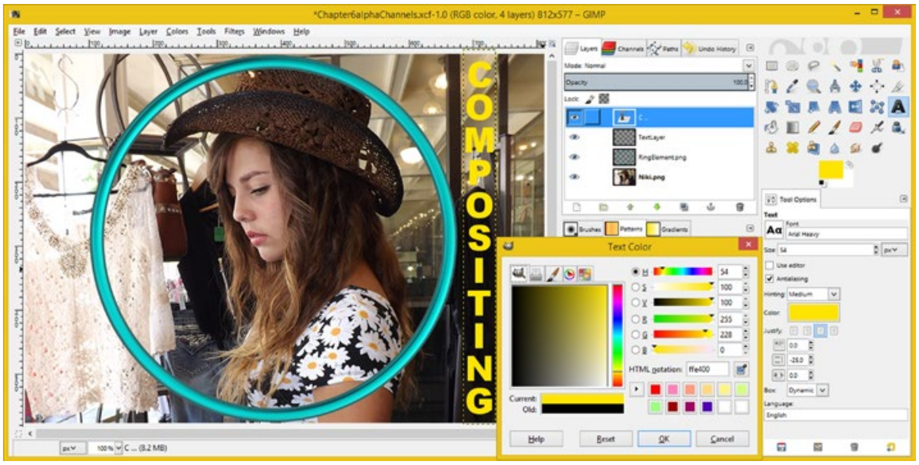


Figure 8-11. *To enter text in GIMP, click the A icon, select Text tool options, set the color to gold, and enter the text*

The next thing you did in Photoshop was to add the Inner Shadow effect so that it looked like the text was cut out of your image. GIMP does not have **Layer Styles** like Photoshop CS6 does, so I will show you how to create shadows during Chapter 9 when I cover pixel Blending modes.

As you can see in Figure 8-11, GIMP creates the layer for you when you create text, so you will not need the TextLayer! I only wanted to show you how to create a new layer in the program, so that you would know how to do it. One of the convenient things about this, as you see in the figure, is that your text layer sizes (width and height) are set for you automatically by GIMP's Text tool.

GIMP doesn't have adjustment layer functionality either, so I will show you a different (non-destructive) work process, using features that will simulate this end result.

To delete layers in GIMP, right-click the TextLayer and select **Delete Layer** from the drop-down menu that appears, as seen in Figure 8-12.

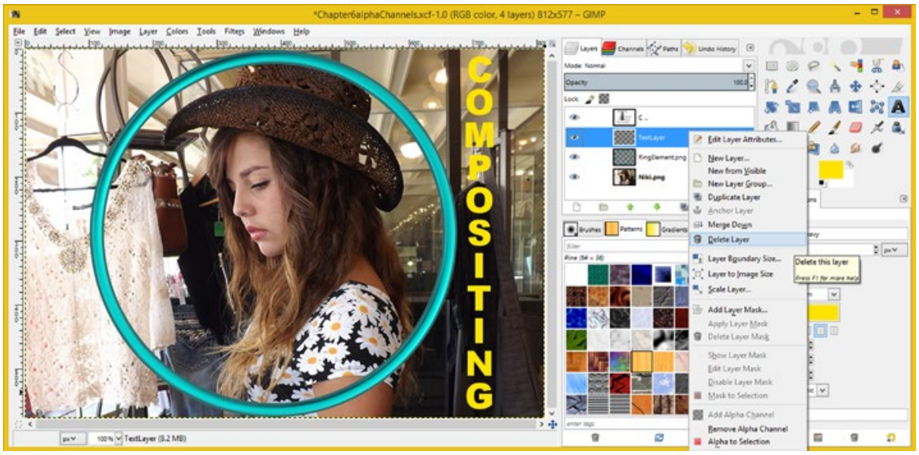


Figure 8-12. Right-click on TextLayer, and select Delete Layer

To implement **non-destructive editing** of your Niki image, right-click the **Niki.png** layer and select the **Duplicate Layer** menu option, as shown in Figure 8-13. Name the new layer "**Niki Black & White**." This work process keeps your original color data intact in the Niki.png layer, allowing you to turn this black-and-white Niki layer into a layer that contains this desaturated image data in its own separate layer.

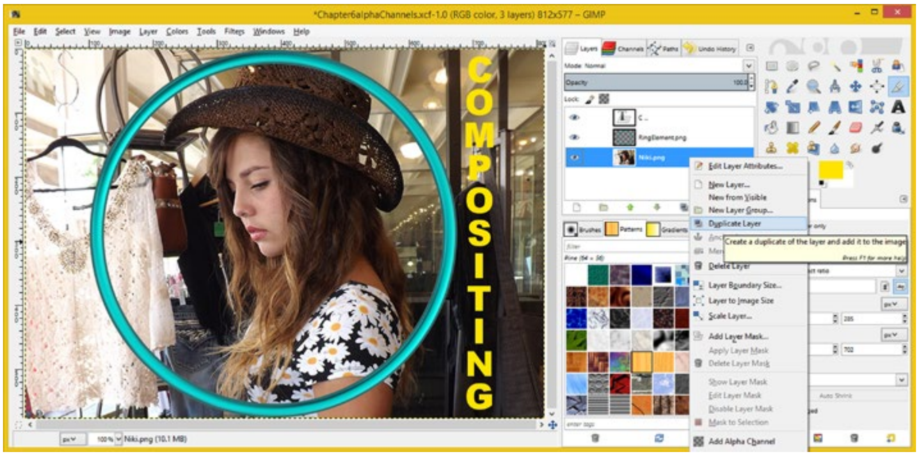


Figure 8-13. *Right-click the Niki.png layer and select Duplicate Layer*

Rename the duplicated layer, by double-clicking the new layer's name to enable it for editing. The result can be seen in Figure 8-14.

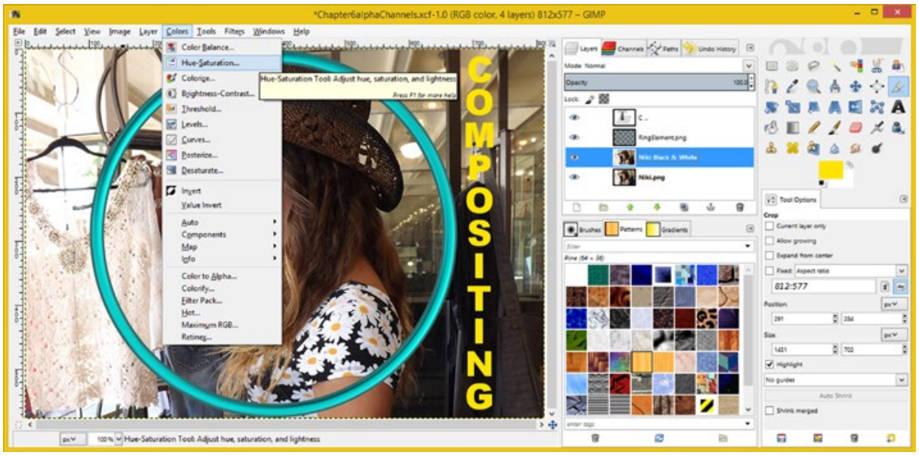
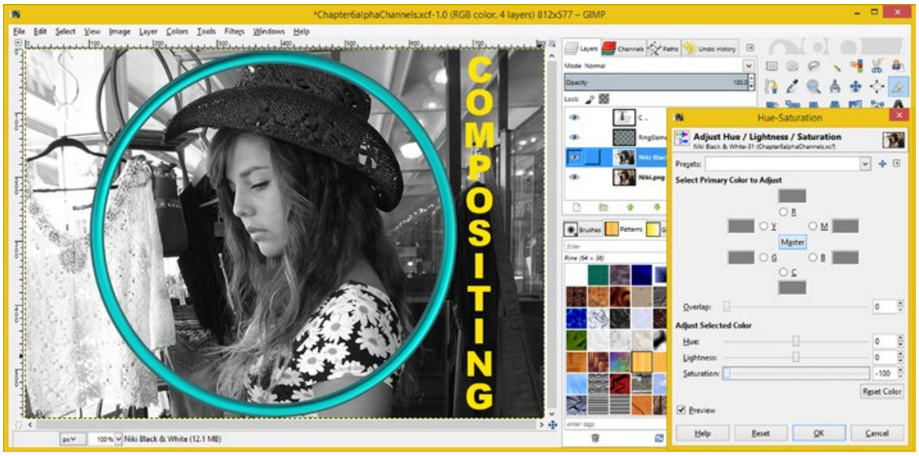


Figure 8-14. *Use Colors ➤ Hue-Saturation to desaturate Niki Black & White layer*

Then use the **Colors ➤ Hue-Saturation** menu sequence, and open the **Hue-Saturation** dialog, so that you can remove any color saturation, as shown in Figure 8-15.


*Chapter6alphaChannels.xcf-1.0 (RGB color, 4 layers) 812x577 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
COMPOSITING
Layers Channels Paths Undo History
Hue-Saturation
Niki Black & White (12.1 MB)

Adjust Hue / Lightness / Saturation
Niki Black & White-31 (Chapter6alphaChannels.xcf)
Presets:
Select Primary Color to Adjust
R Y M Master G B C
Overlap: 0
Adjust Selected Color
Hue: 0
Lightness: 0
Saturation: -100
Reset Color
[✓] Preview
Help Reset OK Cancel

Figure 8-15. *Set the Saturation slider to -100 to desaturate*

The Hue-Saturation dialog allows you to adjust the **Hue** (the color temperature), **Lightness** (the level of white or black in the image), and **Saturation** (the amount of color versus black or white), for complete control over your image layer's color.

If you make sure that the **Preview** option at the bottom of the dialog is selected, you can see the results of your slider settings in real time in the selected layer as you move each slider, so that you can fine-tune, or "tweak" your result until it is exactly as you intended it to be.

As you can see in Figure 8-16, you have used GIMP layers to achieve essentially the exact-same end results as you got in Photoshop, except for drop shadowing inside of your text layer.

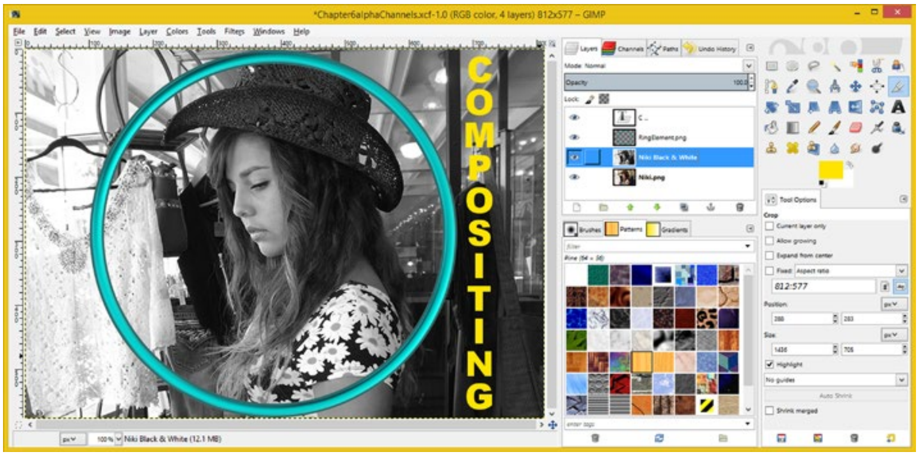

*Chapter6alphaChannels.xcf-1.0 (RGB color, 4 layers) 812x577 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
COMPOSITING
Layers Channels Paths Undo History
RingElement.png
Niki Black & White
Niki.png
Brushes Patterns Gradients
Tool Options
Crop
Niki Black & White (12.1 MB)

Figure 8-16. *You now have the same desaturated result in GIMP*

You will be learning how to create these shadows using GIMP in the next chapter covering **blending modes**. These modes are necessary for use in the creation of photo-realistic drop shadowing effects, so, here again, the order of the topics in the book chapters is a perfect fit.

Summary

In this eighth chapter, we reviewed how to use **layers** to organize your digital image composites using Photoshop and GIMP. We looked at the basic Layer menus in Photoshop and GIMP to see what layer features those packages support, and observed that advanced layer functionality is one of the areas that differentiates Photoshop CS6 from GIMP 2.8.14.

You learned about how to create a new layer in Photoshop and how to add a **Vertical Text** object to the layer. You learned how to create and apply **Layer Styles** in Photoshop as well as how to create and apply **Adjustment Layers**, using Photoshop.

We then went over how to create the very same composite in GIMP, using a different work process. You created vertical text, using a horizontal text tool, and learned how to apply non-destructive editing, by utilizing standard layers, rather than by using adjustment layers, since GIMP does not support those in version 2.8.

In Chapter 9, you will learn more about how **Blending Modes** work with layers and layer functionality in digital image compositing, as the two go hand in hand. Blending modes can also be accessed using your Layer palette in both Photoshop CS6 and GIMP 2.8.

CHAPTER 9

The Algorithms of Digital Imaging: Blending Modes

Now that you have learned how to use layers in a digital image compositing pipeline, in this chapter, we'll dive in deeper and take a look at the blending mode functionality contained within each of these layers. These **blending modes** are present in both GIMP and Photoshop, and they allow **algorithmic functions** for combining pixels across different layers. These blending algorithms were originally the work of **Thomas Porter** and **Tom Duff**, and in the Android API, blending modes are, in fact, a part of a **PorterDuff** class, named after these two men.

In addition to these pixel blending algorithms now being made available in the Android 5 API, Porter-Duff modes are also available in the JavaFX API, which means that they are also now included in Java 7, Java 8, and Java 9. They have also recently been added to HTML5, using CSS3 and JavaScript, and so they are really worth learning about, especially if you're interested in producing interactive new media content using all these popular open source software application development platforms.

If you want to learn how to use blending modes with Java code, in addition to inside digital image compositing software, I have several titles that cover this, including *Beginning Java 8 Games Development* (2014), *Pro Android Graphics* (2013), and *Pro Java Games Development* (2016), all of which are published by Apress.

Pixel Blending: Porter Duff Modes Theory

I am only calling these pixel transfer and pixel blending modes "Porter Duff" here because I'm a programmer, and so that you'll know what they are and where they came from. Although they're called this in Android (programming), they're not in Photoshop and GIMP, so the term is really just something for you to throw out in industry conversations to make yourself look super knowledgeable. These modes are usually called **blending modes**, or **transfer modes**, or sometimes **layer modes**, or **compositing modes**.

Porter Duff modes are applied in digital imaging software using layers that are set to one of the modes—the default being "normal." As you might imagine, the default pixel combination (blending) mode uses no algorithm at all, so keeping it on the normal setting means that no pixel blending algorithm is applied.

It is important to note that alpha channel data will still be applied, though, as it defines **translucency**, and is always factored into your compositing pipeline, whether your blending mode is set to normal or some other pixel blending algorithm.

Alpha channels define straight (normal) opacity and are not a complex algorithmic way of blending pixel colors in order to change their actual color values. Alpha allows a per-pixel fade between your image plate (layer) and what's underneath it, so it uses simple mathematics, at least compared to the complex algorithms that are used in these blending and transfer modes.

This means that you can use your alpha to control where a blending mode is applied on a pixel-by-pixel basis as well as have 256 levels (8 bits) of control over how much of this blending is applied. So your alpha channel and **opacity slider**, which controls all pixel fading at the same time (seen at the top right side of the screen in Figure 9-1), can be used to fine-tune the application of your blending mode.

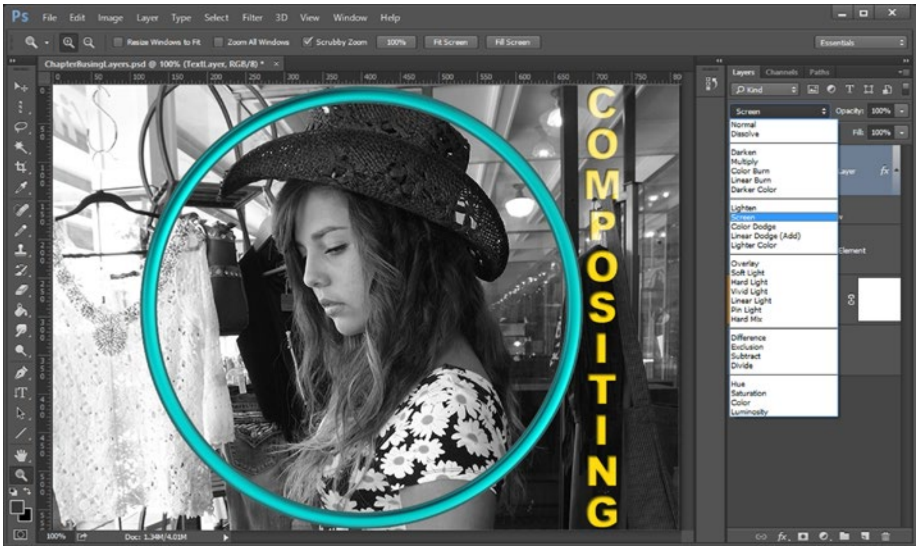


Figure 9-1. *If you select the mode drop-down menu in Photoshop, you can view all of the available modes*

As you will see, these layers are getting to be more and more complex, as you have to keep a growing number of things in mind, such as the alpha channel, blending mode, layer opacity, layer masking, layer styles (special effects sub-layer), layer adjustments, and the color channels that make up your pixel data, to name a few.

This is why I use the term "modal" to describe how one works in digital image compositing software and the Photoshop tool being used. What happens in your composite preview is directly related to the setting for a number of the operational modes, such as the tool you are using, that tool's options, the layer that is currently selected, the blending algorithm that is selected, and so forth. We'll be looking at the modality of digital imaging software in the next chapter, as being aware of all of these modes is the key to your success as a compositor.

Photoshop Blending: Blending Modes Supported

Let's open the Photoshop file from Chapter 8, if it's not open on your desktop already. Once you do so, click the Blending Mode menu (the down-arrow drops the menu down) at the top of the Layers palette, as shown in Figure 9-1. As you will see, Photoshop has five groups of blending modes. The **Normal** (no mode applied) and **Dissolve** (random pixel color transfer) modes are at the top, as they don't use any pixel blending algorithm. The next section describes modes that result in a **darkening** effect**;** the section after that outlines modes that result in a **lightening** effect**;** the section that follows lists those modes that mix pixels to provide **special effects;** and the section after that outlines the modes that provide basic **mathematical** Boolean algorithms. A final section describes modes which will create special effects by transferring pixel **color attributes**.

As you can see in Figure 9-1, I selected the **Screen** mode, so that you can view this effect, which turns your colored text into a screen, so that some of the details of the layer behind it can be seen as part of your colored text object. These modes can add some very subtle and professional details to your image compositing pipeline as well as some crazy special effects, as you'll see during the chapter as we look at some of these algorithmic blending and transfer modes.

Photoshop has **26** blending mode algorithms. Let's take a look at GIMP next to see how many of these blending modes the program supports for your digital image compositing.

GIMP Blending: Blending Modes Supported

To get started, open the `Chapter8usingLayers.xcf` file using GIMP, which can be seen in Figure 9-2. As you will notice, GIMP has fewer blending mode features than Photoshop, but again the core blending and transfer modes that are needed for image compositing are all there. They're grouped in a different way, with lighten, darken, mix, Boolean, then color-based modes, which can be used to create image effects. Just like you access them in Photoshop CS, these blending modes can be accessed at the top of the Layer menu in GIMP. I've dropped down the Mode menu in GIMP in the Figure 9-2, so that you can see the twenty modes (six short of Photoshop) that GIMP currently supports. I would guess that the other six algorithms are patented and will not show up in GIMP 3.0 until after those patents expire.

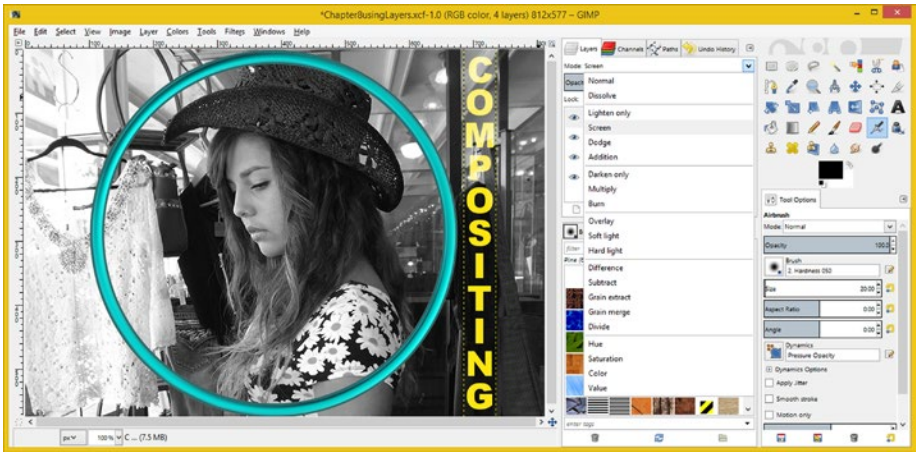


Figure 9-2. *If you select the mode drop-down in GIMP, you can view all of the available modes*

Besides **Normal** (no blending mode algorithm is applied), the most commonly used modes include the **Multiply mode**, which we will be using in the next section to apply shadows, as well as the **Screen**, **Overlay**, **Soft Light**, and math Boolean transfer modes (**Addition**, **Subtract**, **Divide**, and **Difference**). The best way to see what these do is to experiment with using them! You could do this using the colored Text object layer, and (or) use them in conjunction with your RingElement object layer.

Try different modes for each (or both) of the layers to see how they interact with both the Black & White Niki layer and the truecolor Niki layer. To get the truecolor layer in Photoshop, turn off the adjustment layer visibility (eye icon). For GIMP, both the Niki layers can be toggled on and off, using this **layer visibility** (eye) icon as well.

As you can see, GIMP includes all the foundational modes that you'll need to create digital image composite effects. I promised to show you how to create drop shadow effects in GIMP, so let's get that out of the way next, using the Multiply mode.

Using Modes in GIMP: Multiplying a Shadow

Let's continue building the composite you started in Chapter 6 and use the **Chapter8usingLayers** file in GIMP, shown in Figure 9-2, which we just opened. Start by clicking the **COMPOSITING Text layer** to select it, and then use the **Layer ➤ Duplicate Layer** menu sequence to duplicate it. You could also simply right-click the Text layer and select **Duplicate Layer** if you want to use a faster context-sensitive menu shortcut. Select the duplicate layer, name the layer **Shadow**, click the **Text tool**, and use the color picker to set the color to **black**.

If your text color picker is not showing the gold color, be sure that you have clicked the text object itself in your composite image view on the left. After you click on the actual text object in your image compositing pixel display area, GIMP will then allow you to click the color swatch dialog, and drag the color picker crosshair, from gold over to black.

As you can see in Figure 9-3, I've named the layer, set its color to black, and finally right-clicked it and selected **Discard Text Information**. This menu option **rasterizes** the black text so that it becomes **pixel data**, which you will turn into a drop shadow over the rest of the work process.

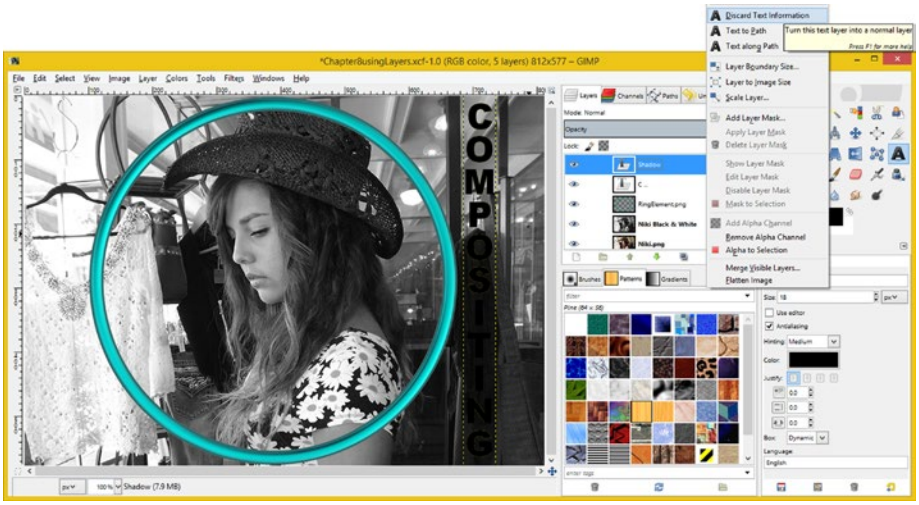


Figure 9-3. To turn the black text into pixel data, first right-click the selected Shadow layer and then click Discard Text Information

Next, click the **Move** tool, shown as selected in the upper-right-hand corner of Figure 9-4, and then choose **Move the Active Layer** in the Tool Options area located underneath the tools palette.

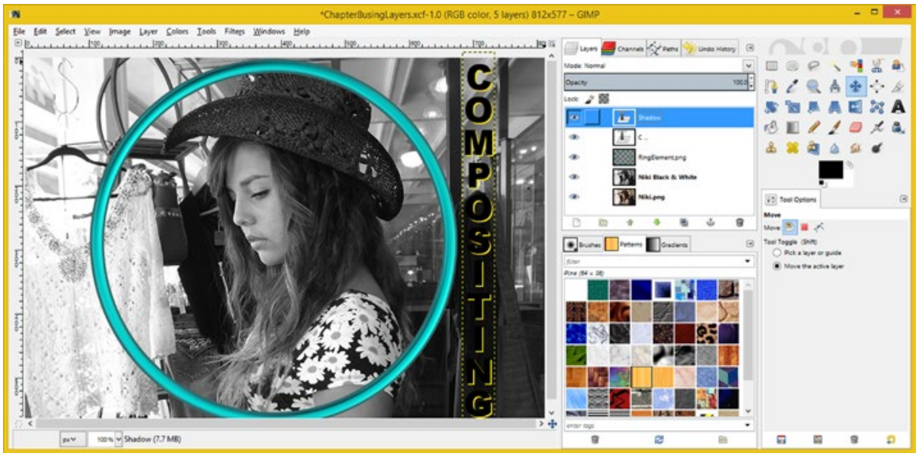


Figure 9-4. You can use the Move tool with arrow keys to offset the black text by 2 pixels

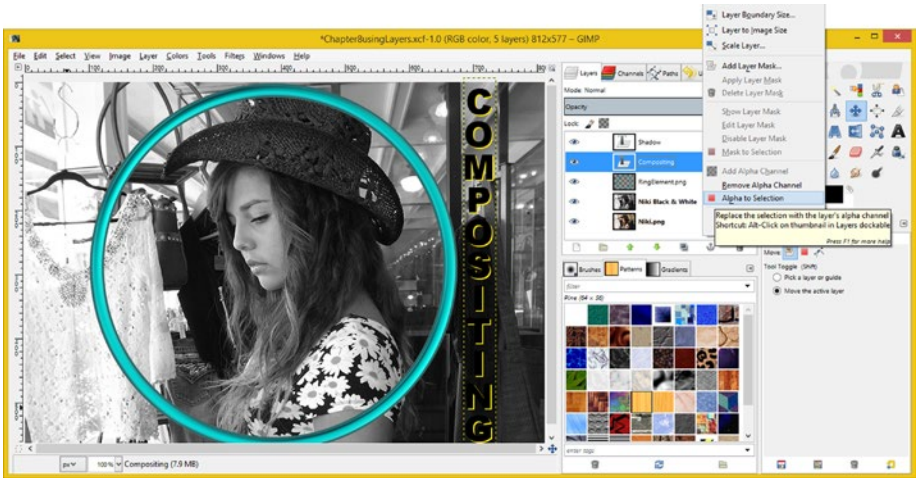
Click the **Shadow** layer to select it, and then the black text in the image preview area, to show the Move tool what it is that you wish it to move. Be sure to just click, not click and then drag, as this will move the black text too much.

You can then use the arrow keys to move the text object, or any selected object in GIMP, by one pixel at a time. This is a useful feature, so be sure to practice using it.

Now, hit the left arrow key twice and the up arrow key twice. This will offset your black text from your gold text, as you can see in Figure 9-4.

You are starting to see how this work process could yield a plethora of drop shadow possibilities—that is, once you apply a blur effect to make it look more like a drop shadow, of course! First, we have a **masking** step to go through, however, before we apply the **blur filter**, to make the shadow look more photo-realistic.

Now, right-click on the Compositing layer, to select it (I've renamed the layer in sentence case letters so that it fits in the layer), and then choose the **Alpha to Selection** option as shown in Figure 9-5 (and be sure to read the pop-up info for hints, tips and shortcuts).



Replace the selection with the layer's alpha channel
Shortcut: Alt-Click on thumbnail in Layers dockable

Figure 9-5. *Selecting the Compositing layer and using the Alpha to Selection option*

■ **Tip** If you mouse over any of the GIMP user interface components, such as menu items and tool icons, you'll get yellow pop-up information regarding what that GIMP function does, as well as additional tips, tricks and shortcuts. This is a great way to learn the software.

Next, select your **Shadow** layer, and use the **Edit ➤ Clear** menu sequence, to remove the majority of the pixels, using the selection that you culled from your Compositing text layer, as shown in Figure 9-6. This is an example of a "modal operational move," as I call it, where you select a layer and the tool that is executed after selection becomes of tantamount importance.

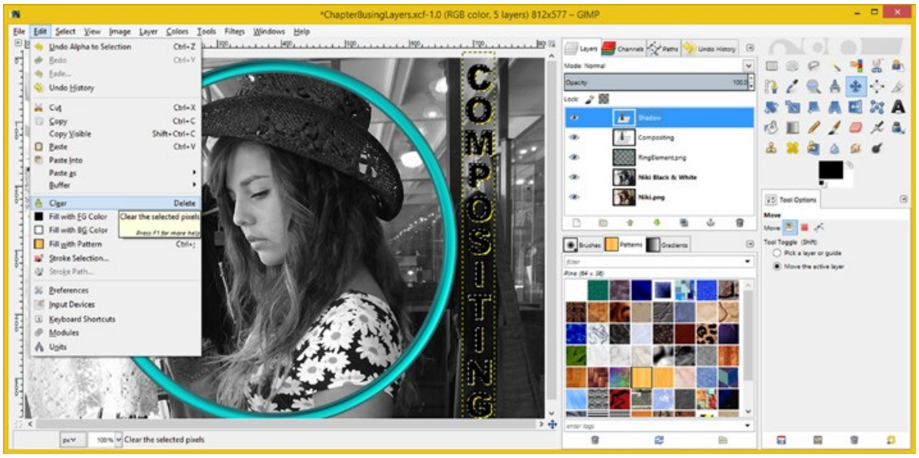


Figure 9-6. *To remove the majority of the pixels, select the Shadow layer and use the Edit ➤ Clear sequence*

Then, remove your black text selection (marching ants), using a **Select ➤ None** menu sequence, and select your Shadow layer, so you can apply your blur.

Use the **Filter ➤ Blur ➤ Gaussian Blur** menu sequence, and open the **Gaussian Blur** dialog, as shown in Figure 9-7. Apply **2 pixels** of blur, in both the *X* and *Y* dimensions. Make sure that the chain icon is locking the aspect ratio for this dialog.

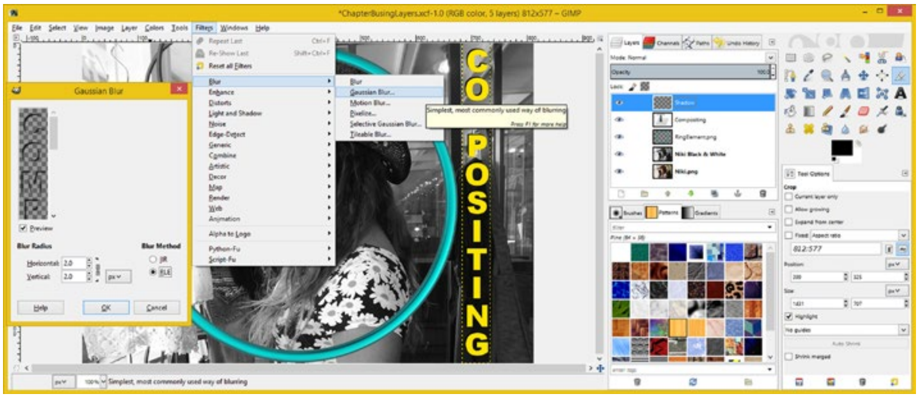


Figure 9-7. *To blur the shadow edge, use the Filter ➤ Blur ➤ Gaussian Blur menu sequence*

Click the **OK** button, to apply your Gaussian blur setting to the Shadow layer, and the shadow will immediately look more realistic.

Then select the **Move** tool, click on the **Shadow** part of the preview image to tell GIMP that is what you want to move, and use the arrow keys to position the shadow treatment.

Position the drop shadow so that you achieve the same effect that you got in Photoshop with the Layer Style Inner Shadow tool.

You may have to use the arrow keys to move the shadow a couple of pixels in the X and Y dimensions so that it looks like the shadow is on the inside of the text object, making it look like it is being cut into the image.

To make the drop shadow color blend (multiply) with the gold text color, set the Mode drop down menu to **Multiply**, as is shown in Figure 9-8 at the top right of the screenshot, circled in red. This adds even more realism, especially in the areas of the shadow which have been blurred.

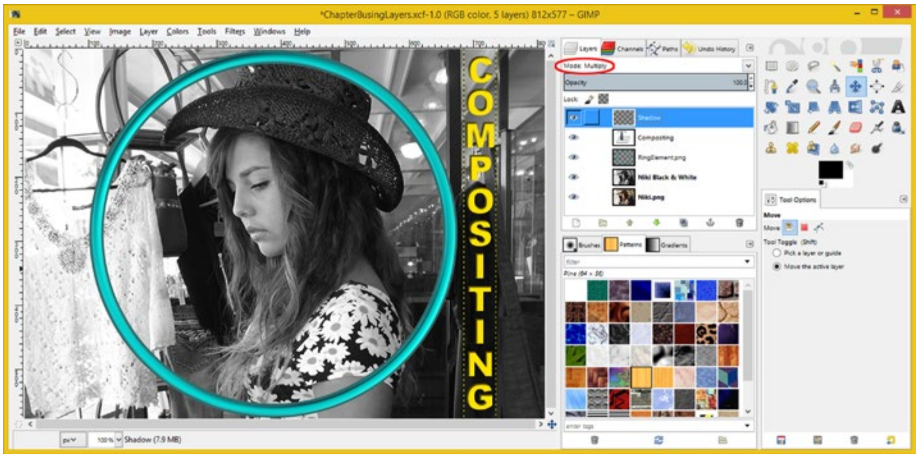


Figure 9-8. *You can fine-tune the shadow using the Move tool and a Multiply mode*

As you can see, you can also accomplish the majority of what you can in Photoshop using GIMP. It took you four layers to create this effect in Photoshop, and five in GIMP, but the end result can be tweaked to be identical across both programs.

I will use Photoshop to look at the remaining modes, which are used the most frequently for compositing, to make sure that I give equal billing to both software packages in each chapter in the book.

Photoshop Blending Modes: Cool Effects

During the rest of this chapter, let's take a look at some of the blending modes in Photoshop, which can provide you with some really powerful image compositing "looks" and special effects. Because some of them are very similar to one another, I'm not going to cover all 26 modes here, but I will go over some of the modes you'll find yourself using frequently. These include modes that can simulate light shining on your imagery, overlay one image on top of another, and create similar pixel-based 2D special effects. We have already covered the **Screen** mode and **Multiply** modes in GIMP 2.8.14, so I am going to focus on the other modes in Photoshop CS6, as these two work the same way in Photoshop as they do in GIMP. Let's get started looking at all the cool effects that blending modes can create, using the image composite we have created thus far.

Soft Light: Shining a Light on Your Digital Imagery

The Photoshop Soft Light mode is useful for simulating a light shining on an image. Let's create this effect in our Chapter 8 image composite, which is seen in Figure 9-9. Start by turning off the **Text Layer Effects** by clicking the visibility eye icon next to the **Effects** layer in the Layers Palette on the right side of the screen. Then, select the **Text** tool, and click on the gold color swatch at the top of the screen, and use the color picker to change the text color to **white**. We will do this so that we can simulate a light shining on the image. Next, click the **Move** icon at the top of the icon toolbar, which is on the left side of the screen, and move the vertical text to the right side of the image composite. Finally, select the **Soft Light** mode from the drop-down menu in the Layers Palette, and you will get an effect similar to a text shaped flashlight shining on your digital image composite. Pretty Cool Stuff!

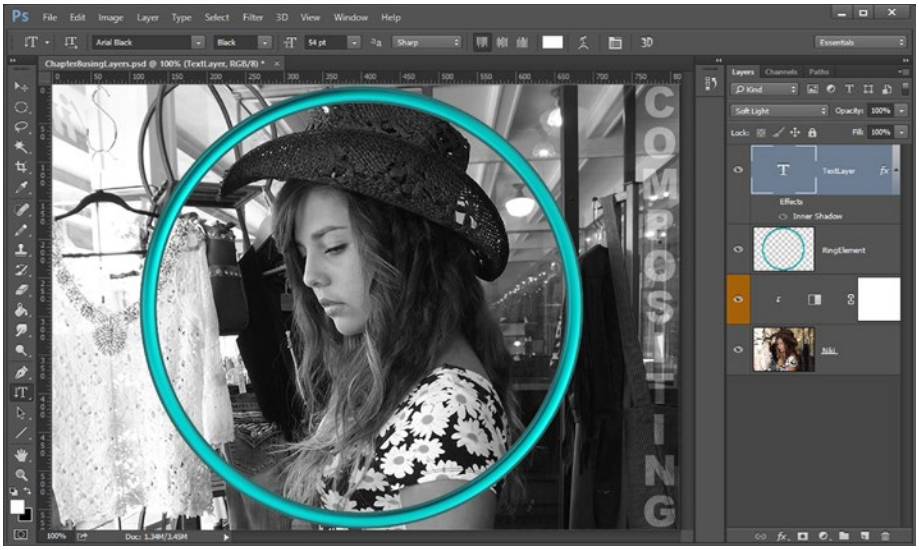


__Figure 9-9.__ Select the TextLayer, and change mode to Soft Light

We have been looking at the Soft Light mode; now let's see if a **Hard Light** mode provides us with a significantly different special effect. It is important to note here that both Soft Light and Hard Light modes are also available in GIMP 2.8.14, so you can create these effects using that software package as well, if you like, possibly to get a little bit more practice at digital image compositing.

Hard Light: Adding a Glass Overlay Effect to Text

For the Hard Light test, I reset the text to an orange-gold color using the Text tool and turned the Inner Shadow Layer Style back on using the visibility eye icon, as I want to show you how to combine drop-shadowing effects and blending modes for a maximum visual impact. I also moved the vertical text back to its original location between the wood beams in the image and selected the **Hard Light** mode from the drop-down menu at the top of the layer. A professional end result is seen in Figure 9-10.

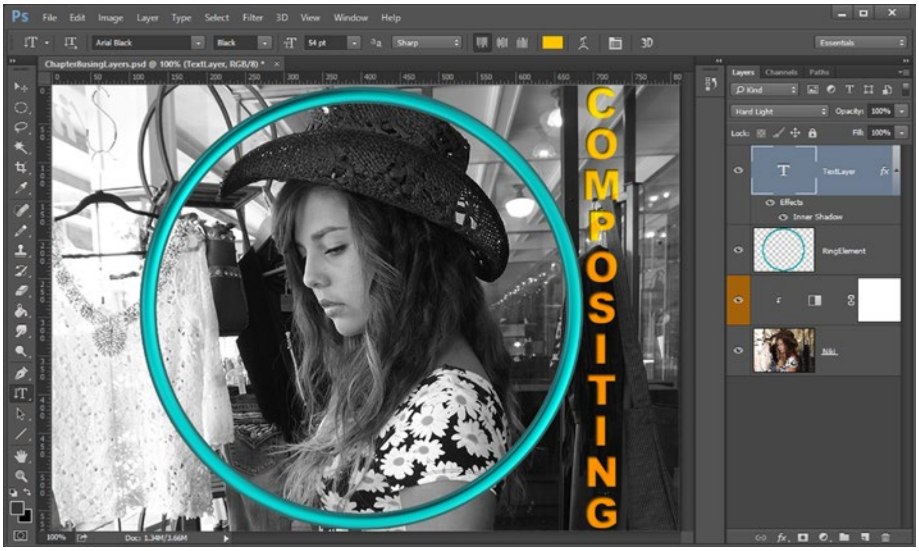


Figure 9-10. *Left-click the TextLayer and set a Hard Light mode*

Linear Burn: Turning a 3D Metal Hoop into Glass

Let's set modes for both the text and ring-object layers next, in order to increase the amount of pixel effects data that Photoshop is processing. We will have blending modes calculated on two layers and Inner Shadow effects on the text layer as well as Black & White image adjustments on the Niki layer.

First, select your **RingElement Layer**, and replace the Normal blending mode with the **Linear Burn** mode, which will turn your aqua chrome hoop into an aqua glass hoop, as shown in Figure 9-11.

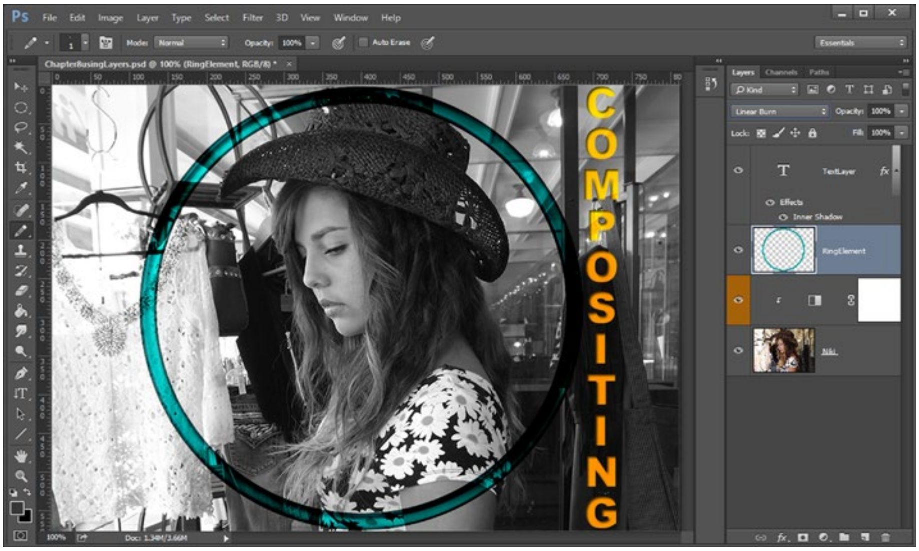


Figure 9-11. *To turn your aqua chrome hoop into an aqua glass hoop, select the RingElement layer and set the Linear Burn Mode in the drop-down menu*

The Linear Burn blending mode is similar to the Color Burn blending mode. Where the Color Burn mode increases contrast, the Linear Burn mode decreases brightness, darkening the base color and reflecting the blend color, which makes the metal look like glass. The Linear Burn mode is similar to the Multiply blend mode, but the Linear Burn mode produces a much more realistic result with this particular special effect implementation (metal to glass). In fact, try selecting a Multiply mode now for comparison purposes.

It's important to note that using **White** as a blend color with a Linear Burn blending mode will produce no image changes.

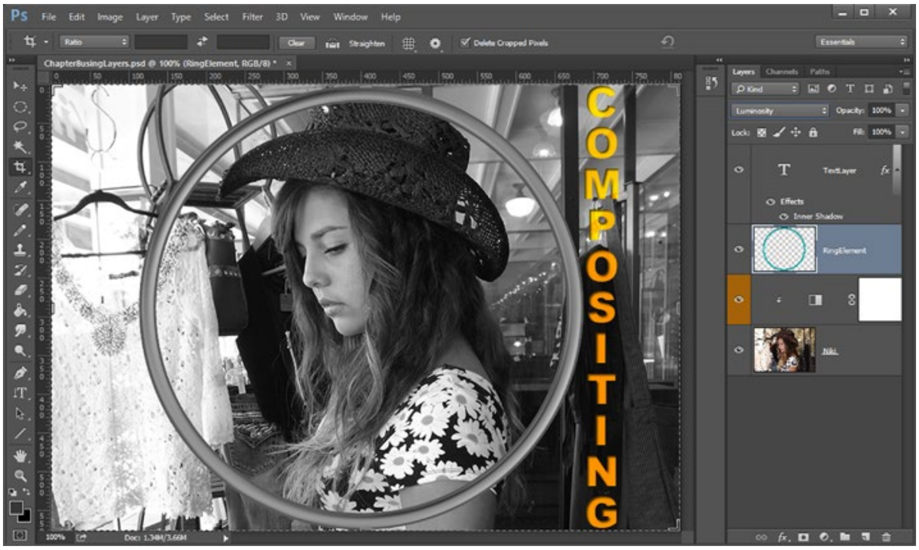
Digital photographers can utilize the Linear Burn blend mode to make both tonal and color adjustments to their digital photography. This mode, which was introduced in Photoshop 7, is particularly useful when you want enhanced effects in those darker areas of your digital photography. It is sometimes known as **Subtract** in other imaging software packages.

Next, let's use a **Luminosity** mode to turn your blue ring element into a chrome hoop element by removing the saturation. As we have seen in Chapter 8, removing color Saturation leaves only Luminosity (black and white, or grayscale) values.

The **Hue** affects the color temperature, or the color, of the image; the **Saturation** affects how much color (versus how much grayscale) it contains; and the **Luminosity** affects how bright (white values) or dark (black values) the image uses.

Luminosity: Turning Your Metal Hoop into Chrome

Here again, start by selecting your RingElement Layer, as shown in Figure 9-12. Then replace the Linear Burn blending mode with the **Luminosity** blending mode. This will transform your aqua-tinted chrome ring element into a darkened-silver chrome ring element. When using blending modes, this is as simple as setting the Luminosity blending mode on the layer that contains your 3D ring element.



File Edit Image Layer Type Select Filter 3D View Window Help
Chapter8usingLayers.psd @ 100% (RingElement, RGB/8) *
COMPOSITING
Layers Channels Paths
Luminosity
Opacity: 100%
Fill: 100%
TextLayer
Effects
Inner Shadow
RingElement
Niki

Figure 9-12. Select a RingElement layer and set Luminosity mode

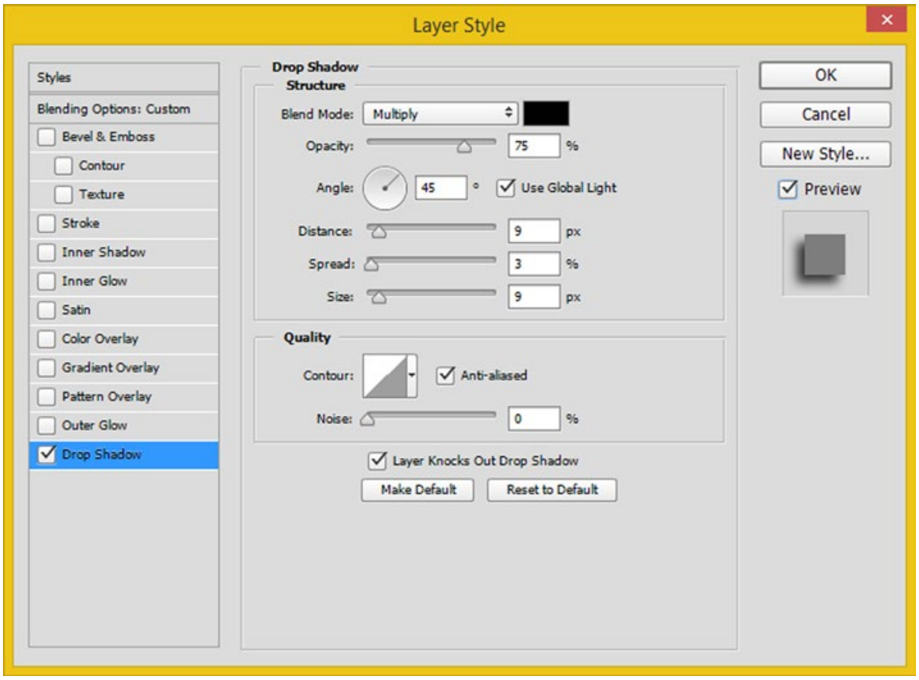
Luminosity is one of your four color-oriented, which could be termed "tonal," blending modes, in a group located at the bottom of Photoshop's blending mode drop-down menu, as you saw in Figure 9-1. These four color blending modes operate on those Hue-Saturation dialog values that you have seen before, affecting only the Hue, Saturation or Lightness components of the pixels between the two images.

In simple terms, when you combine a layer with another layer set to use the Luminosity blending mode, the resulting pixel color keeps the original (underneath) layer's hue and saturation values, but the luminosity value comes only from the blending layer set to Luminosity. Next, let's add the **Drop Shadow** Layer Style to your Ring Element Layer and use the **Difference** blending mode to create a cool special effect, so that you are using all the capabilities of these four layers that we have implemented in Photoshop CS6.

Difference: Using a Layer to Invert Pixel Colors

Now, let's add some more complexity to our existing layers by adding a Layer Style to the RingElement layer, so that we can drop shadow it over the Niki image layer and have it look like it is floating above the image.

To do so, right-click your **RingElement** layer after you select it and choose the **Blending Options** menu item, to access the **Layer Style** dialog, as shown in Figure 9-13.



Drop Shadow ✓

Blend Mode: Multiply
Opacity: 75 %
Angle: 45 ° ✓ Use Global Light
Distance: 9 px
Spread: 3 %
Size: 9 px
Contour: ✓ Anti-aliased
Noise: 0 %
✓ Layer Knocks Out Drop Shadow
✓ Preview

Figure 9-13. *Set the Drop Shadow Layer Style settings*

Then, select a **Multiply** blend mode, an Opacity of **75** percent, and an Angle of **45** degrees. Select **Use Global Light**, and then set a Distance of **9** pixels, a Spread of **3** pixels, and a Size of **9** pixels.

Check off the Anti-aliased and Preview checkboxes, as is shown in the figure.

Once you click the **OK** button, you will have a realistic drop shadow underneath your ring element that you did not have to create.

You saw the work process that it would take to create this drop shadow effect in GIMP. In Photoshop, the Layer Style dialog does this for you, which is one of the program's advanced features that GIMP does not yet have. Next, select the **Difference** Mode under the Layers tab, as shown in Figure 9-14.

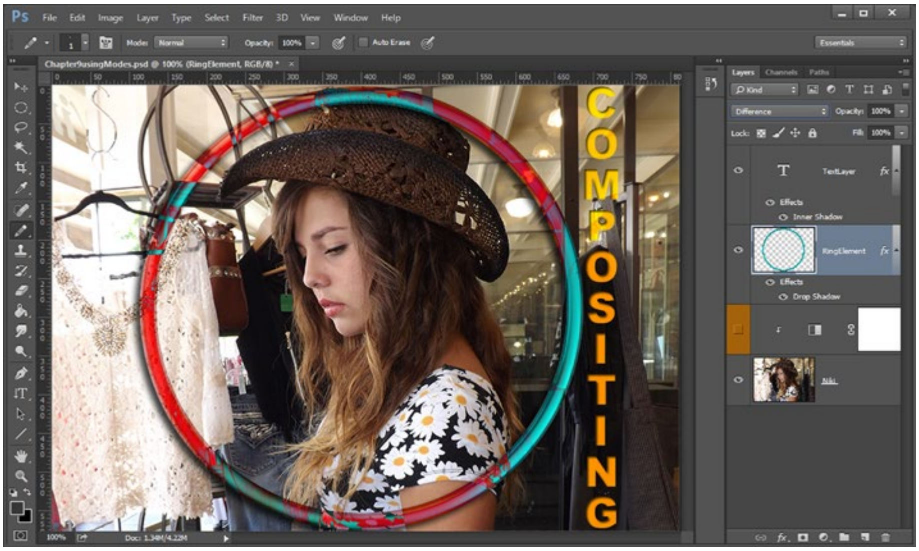


Figure 9-14. *Select a RingElement layer and set Difference mode*

Now, turn your visibility (eye) icon off in your orange adjustment layer, which will bring back the original color data from the Niki layer so that you can see the effect of the mode better. Areas of the ring will turn red or blue, based on image data in the layer underneath the RingElement layer.

Make sure to spend some time experimenting with all the blending modes in Photoshop and GIMP so that you can get a feel for what they'll do for your digital image compositing pipeline in combination with alpha channels, opacity, layers, and layer styles. This is one area where experience with the modes helps you to know what will work and what will not for a given image.

Summary

In this ninth chapter, we explored using **blending modes** to apply powerful, pixel-based algorithms to your digital image composites using Photoshop and GIMP. We looked at the basic Mode menus in Photoshop and GIMP to see what blending mode algorithms those packages supported, and observed that the advanced functionality of blending modes is one of the areas that differentiates Photoshop CS6 from GIMP 2.8.14, with Photoshop giving you six additional blending mode algorithms.

You learned about how to create the inner drop shadow in GIMP using the multiply mode and a work process that involved duplicating text, selecting alpha data, masking, and applying a blur filter.

We looked at some of the other popular blending modes in Photoshop, which we did not look at in GIMP, and applied these to your Photoshop file for the chapter. We also added blending modes and layer styles to more of your layers to increase the complexity of the project even further and demonstrate what blending modes can do in conjunction with alpha channels and layer styles.

In Chapter 10, you will learn more about digital image compositing software's **Modal Operation** and how these modal operations work with digital image compositing layers and layer functionality, as the two go hand in hand.

CHAPTER 10

The Modality of Digital Imaging: Modal Operation

Now that you've learned some of the basic components of digital image compositing, such as concepts and terms, alpha channels, selection sets (masking), layers, blending modes, and the like, let's spend a chapter taking a closer look at the nature of the **modal operation** of advanced digital imaging software. This modal operation is also present in other high-end multimedia production software packages, such as 3D, digital video editing and special effects, and audio packages. This modal functionality is also part of programming languages, such as C, C#, C++, Python, Java, JavaFX, and JavaScript.

Due to the modality of these advanced software packages, there are a number of factors—in this case, **operational modes**— whose settings you need to keep track of in your mind at all times during your operation (use) of the program.

This is because the settings for these modes—for example, what tool you are using; your tool option settings; which layer is selected; what keyboard modifier (e.g., Shift, Control, Alternate) is being held down; which blending mode is selected; which icon (e.g., visibility, lock, etc.) is on or off; and so on—will determine the end result that you get with the software.

Modal Operation: Part of the Pipeline

Just like there is a compositing pipeline that processes your image layer, mode, and channel settings as it progresses up the layer stack from bottom to top, there is a more complex operational-settings pipeline that the digital image compositing software is processing in real time as you use the software. To be able to produce the result you are asking the software to create, it has to pay attention to the tool you have selected, the layer you are working on, and the options you have selected, as well as what you are doing with your mouse and keyboard.

Processing Pipeline: Tool, Options, and Settings

Since this is a book about the fundamentals of digital image compositing, and not its advanced functions, I am not going to get into the level of what tool, option, layer, alpha channel, selection set, blending mode, key modifier, and pixel location (real-time mouse location, movement, drag, etc.) is processed in what order inside of the software code base. From a user perspective, that does not really matter anyway, because before you use the tool (move the mouse, click, drag, or depress keyboard keys), you set all of the modes that define what is going to happen before you start using your selected tool. If you set these operational modes incorrectly, you will get unintended results. Sometimes, these can even be really cool unintended results, which you might decide to keep!

Since these expert software packages have many different tools and options, most of which can affect one another, the use of complex, new media software can quickly become an advanced project, as you have to keep all of these settings, and relationships, straight in your mind. This is similar to computer programming where you have to keep track of variables and how you are processing them in a similar fashion. The more complex your image composite (or software application) gets, the more components you have to keep straight in your mind, in real time, as you work on it.

Practice Makes Perfect: Keeping It All in Your Mind

Using digital image compositing software might look easy as you look over another operator's shoulder, because you cannot see what is going on inside of their mind. However, in your endeavor to master these types of software packages, you will find that it is a challenge to keep all of these operation modes straight in your mind, in real time, as you attempt to get the software package to function in exactly the way you are intending it to, in order to accomplish the project objective—that is, to achieve your desired visual end result.

What this means is that you'll need to consciously start to practice keeping track of all of these different operational modes, and how they affect one another, as part of your learning process, if you want to truly master any digital image compositing software package.

As you have seen already during the book, all of these digital image editing and compositing software package offerings process their operational modes pipeline differently!

This is why in this book I have chosen to show the different work processes for both of the most popular image compositing software programs—the open source program, GIMP, and paid software program, Photoshop—which ultimately achieve the same results. In this way, the book covers a much larger user base and also drives home a modal operation point, since these two packages use different work processes.

Modal Operation: A Photoshop Example

Let's continue to make your current compositing project more advanced, and this time look specifically at how a modal operation approach is being undertaken as we go along. We will learn about some new features and tools within Photoshop and about how to turn the pixels inside of the RingElement into color, while keeping the pixels outside of the RingElement in black and white.

Continuing to implement more advanced features will allow you to continue to see how using the modal operation is important in getting the desired results in your digital image compositing software package; in this case, Photoshop CS6.

Photoshop Magic Wand Tool: Modal Selection Sets

Open up your Chapter 9 Photoshop file `Chapter9usingModes.psd`, if it's not open already. Then, click the **RingElement** layer to set the layer you will be operating on. Next, click the Magic Wand tool, shown as selected on the top left toolbar in Figure 10-1. Now, **click inside the RingElement** in your image composite to select that area. Use the **Select ➤ Modify ➤ Expand** menu sequence to open the **Expand Selection** dialog, and then expand the selection by **9 pixels**, so that the selection is halfway underneath the RingElement.

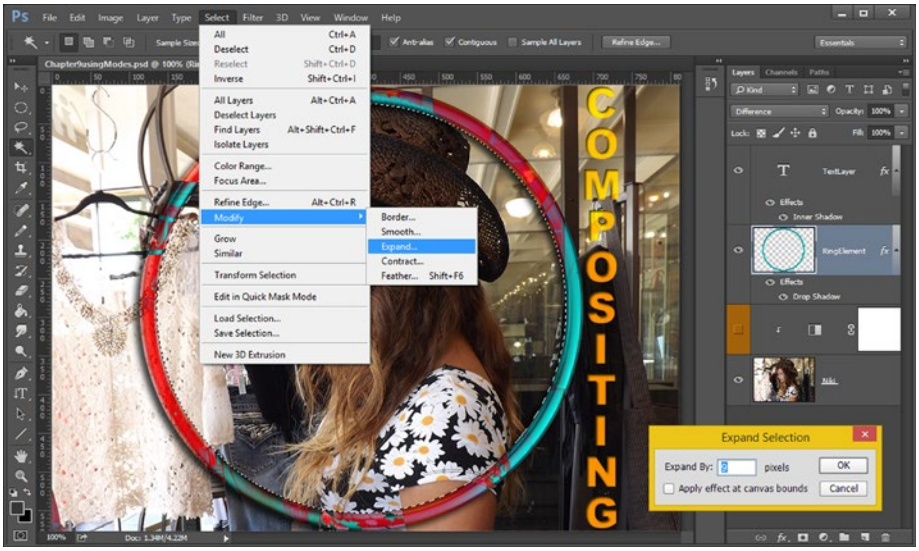


__Figure 10-1.__ To begin your modal operation, select the inside of the ring and expand it by 9 pixels

If you wanted to save the selection as an alpha channel, you would use **Select ➤ Save Selection**, and name your selection **RingSelection**.

Now that you have a selection area lying under the RingElement, since the Niki layer is your back plate, and at a lower compositing order than your RingElement layer is in the layer stack, it is time to learn how to apply this selection as a **Layer Mask**, to, or for, your Black & White Adjustment layer.

Photoshop Layer Mask: Delete Default Layer Mask

Start by selecting the **orange adjustment layer**, as shown in Figure 10-2, and **right-click** on the white **Layer Mask Preview** icon at the right side of the adjustment layer. Then, use the **Delete Layer Mask** menu option to remove the white layer mask. This layer mask selects the entire image for black-and-white adjustment processing, since every pixel in the mask is fully on (white or #FFFFFF value, which means that there is a full adjustment application for each pixel in the entire image). What we're doing here is deleting the default layer created by Photoshop, so that in the next section we can instruct Photoshop to make a new layer mask using the selection set that we created in the previous section using the Magic Wand in conjunction with the RingElement layer.

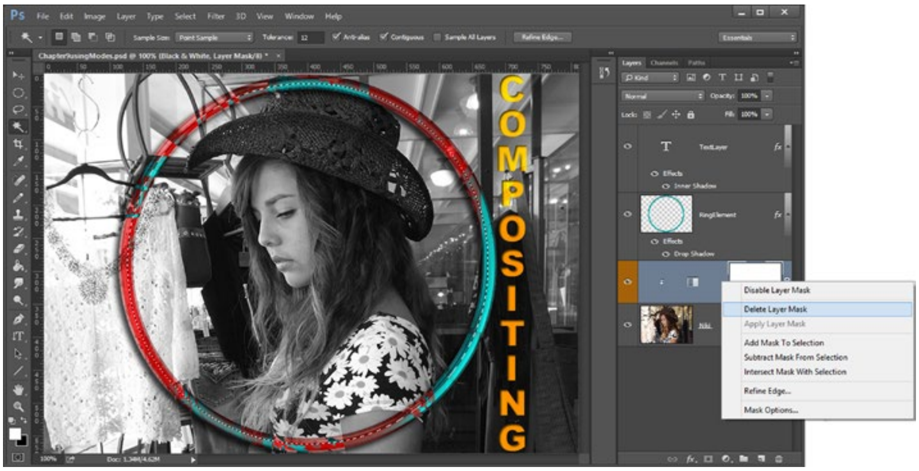


Figure 10-2. *To remove the white layer mask set up by Photoshop, right-click the adjustment layer mask and delete it*

There's another work process that you can use to achieve what we're about to do in the next section—replace the default (all pixels selected) layer mask using your custom selection set inside the RingElement. This work process would involve selecting an inside area of your RingElement, as you did in the previous section, **before** you apply the **Layer ➤ New Adjustment Layer** work process. If you do things in this order, your selection will then appear as the layer mask that is shown in the adjustment layer automatically.

Custom Layer Mask: Add Selection as Layer Mask

The orange adjustment layer should still be selected at this point, as seen in Figure 10-3, with an additional square around a Black & White Adjustment icon showing that the layer mask is empty. To add the selection set to the adjustment layer as its mask, you would **hold down the Alt key** on the keyboard and **click the Add Layer Mask icon** at the bottom of your Layer palette. This will make the inside of the RingElement show Niki in color and show the outside of the RingElement in black and white. If you wanted to get the reverse effect—that is to reverse the mask that is generated—simply do not hold down the Alt key, and then click the Add Layer Mask icon.

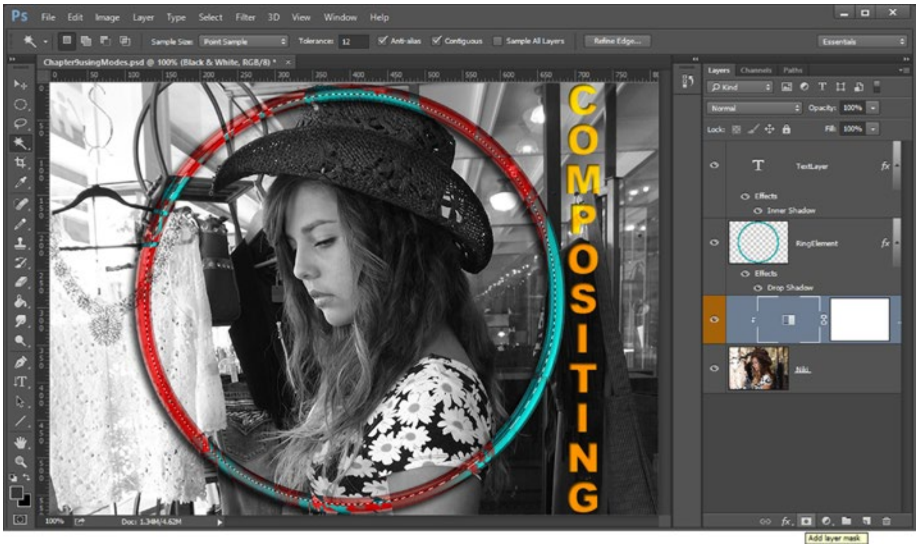


Figure 10-3. *To add the selection set to the adjustment layer, hold down the Alt key and click the Add Layer Mask icon*

Figure 10-3 also shows the **pop-up tool tip** at the bottom right corner of the screen, telling you which icon is your Add Layer Mask icon. The other icons can be used to access Layer Style (fx) and things such as creating a new layer, deleting a layer, locking a layer, and other core layer functions.

Mouse over these icons yourself if you want to memorize which of them perform which layer functions, something you should do if you want to become a professional digital image compositor.

As you can see in Figure 10-4, your adjustment layer now shows your selection as its Layer Mask, and, the inside of your ring element is now using colored pixels whereas the outside is using black-and-white (grayscale, or, desaturated) pixels.

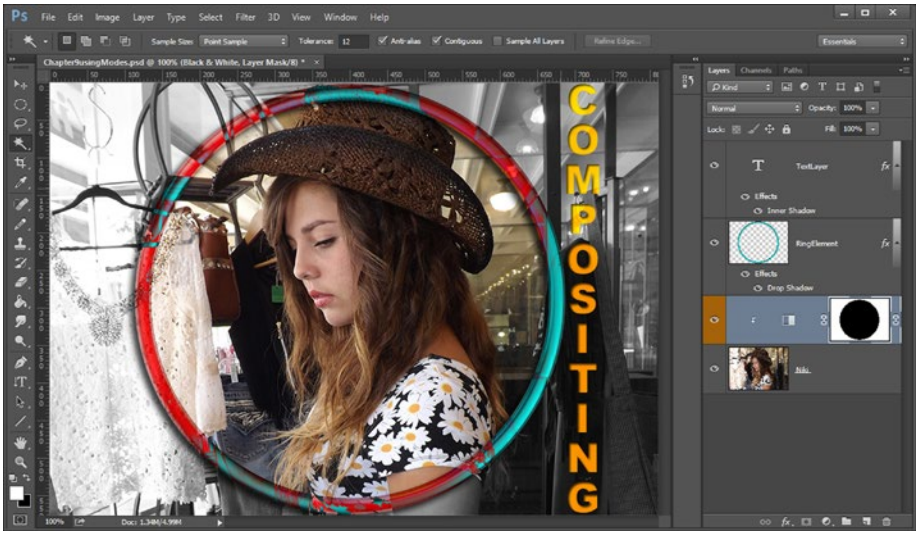


Figure 10-4. *Convert your selection to an adjustment layer mask*

Notice that we've done all the compositing work with only four layers, and yet, we have applied half a dozen or more special effects.

Now, let's take a look at how this work process should be done using modal operations native to GIMP 2.8.14.

Modal Operation: The GIMP Example

Open the `Chapter9usingModes.xcf` file and select the **RingElement** layer and **Fuzzy Select tool**. Click inside the ring to create the selection you created in Photoshop, as seen in Figure 10-5.

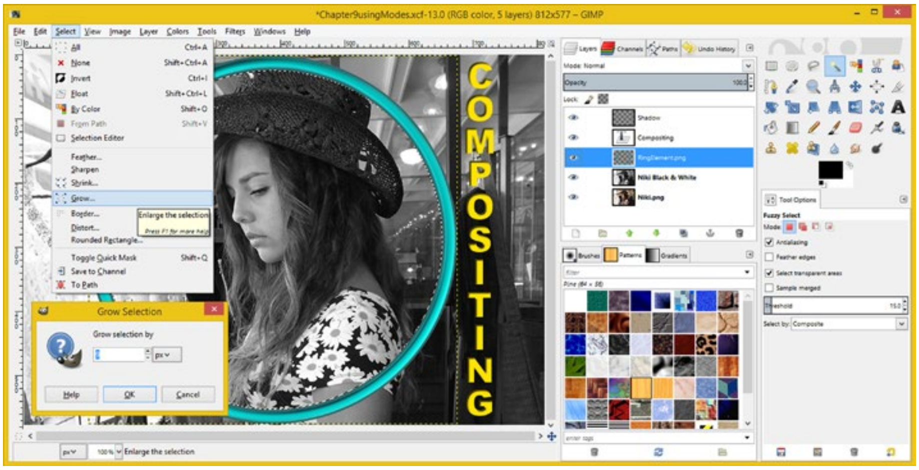


Figure 10-5. Start the GIMP modal operation by selecting the RingElement Layer using the Fuzzy Select tool

Then, use GIMP's **Select ➤ Grow** menu sequence to access a **Grow Selection** dialog and enter a value of 9 pixels, just like you did in Photoshop. Hit the **OK** button, and you will have your mask selection ready to use as a layer mask.

Since this selection makes the outside of the ring black (transparent), and you want the inside of your ring to be transparent, use GIMP's **Select ➤ Invert** menu sequence, as shown in Figure 10-6.

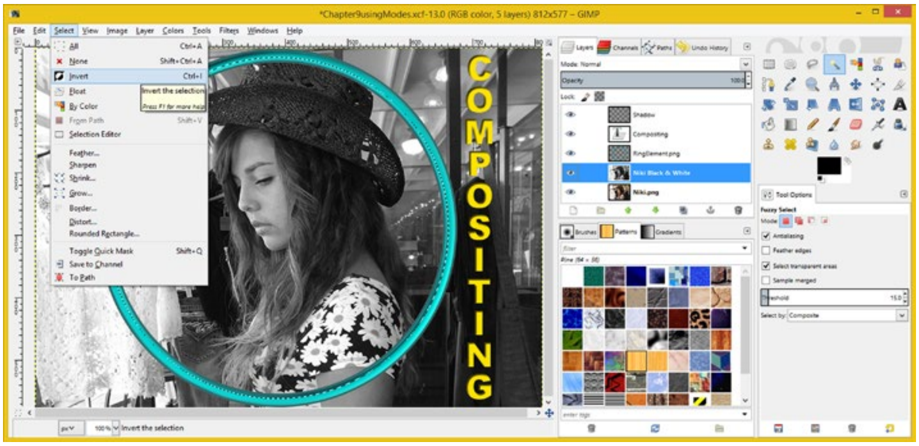


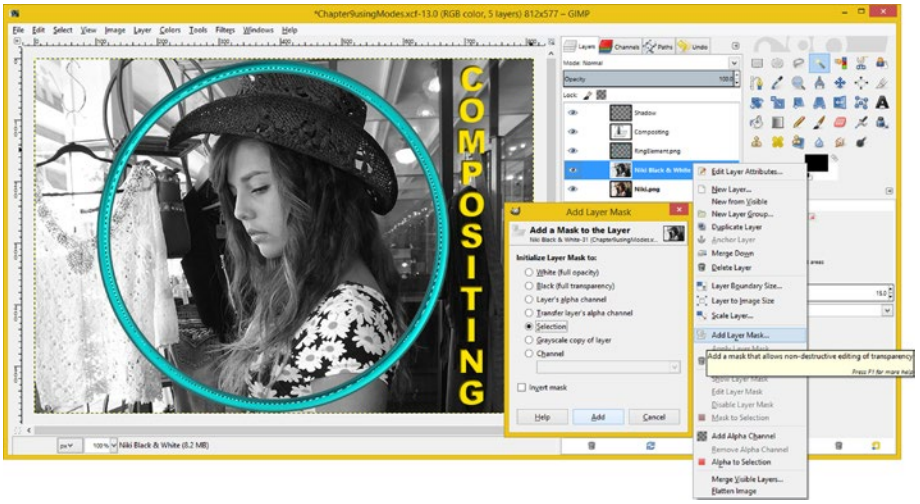
Figure 10-6. To make the inside of the ring transparent rather than the outside, select the Niki Black & White layer and invert the selection

If you want a black-and-white Niki.png version to be inside of the RingElement instead of the color version, then leave the selection as is, as you do not then need to invert it.

Now we're ready to add a layer mask to the black-and-white Niki.png layer, which will allow a color Niki.png layer to show through. This is precisely what your Photoshop adjustment layer mask was doing, but it was doing it "under the hood" in system memory, rather than up-front in the Layers palette, as you are doing here in GIMP. Different work process, same end result.

GIMP Layer Mask: Adding a Layer Mask

To add a layer mask to the Niki Black & White layer, so that the Niki.png (color) layer will show through, you'll need to select the **Niki Black & White layer** for this modal operation and then right-click on it. You'll also need to select the **Add Layer Mask** menu option, as shown in Figure 10-7, along with its pop-up tool tip and the **Selection** option in the **Add Layer Mask** dialog and then click the **Add** button. This will add the currently active selection as the Layer Mask for your Niki Black & White layer.


*ChapterSusingModes.xcf-13.0 (RGB color, 5 layers) 812x577 – GIMP

Add Layer Mask

Add a Mask to the Layer

Niki Black & White-31 (ChapterSusingModes.x...

Initialize Layer Mask to:

- ( ) White (full opacity)
- ( ) Black (full transparency)
- ( ) Layer's alpha channel
- ( ) Transfer layer's alpha channel
- (●) Selection
- ( ) Grayscale copy of layer
- ( ) Channel

☐ Invert mask

Add a mask that allows non-destructive editing of transparency

Press F1 for more help

Figure 10-7. *To get the Niki.png color layer to show through, select the Niki Black & White Layer and then Add Layer Mask*

The layer mask will then allow a portion of the Niki.png color pixel data to show through the layer above it, as you can see in Figure 10-8.

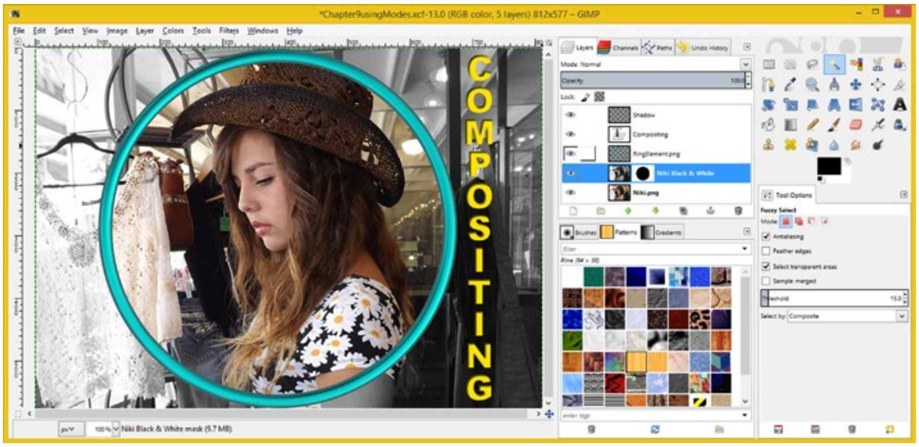

*Chapter9usingModes.xcf-13.0 (RGB color, 5 layers) 812x577 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
COMPOSITING
Layers Channels Paths Undo History
Mode: Normal
Opacity 100.0
Lock:
Shadow
Compositing
RingElement.png
Niki Black & White
Niki.png
Brushes Patterns Gradients
Tool Options
Fuzzy Select
Antialiasing
Feather edges
Select transparent areas
Sample merged
Threshold 15.0
Select by: Composite
100% Niki Black & White mask (9.7 MB)

Figure 10-8. *Layer Mask revealing Niki.png Layer pixel color data*

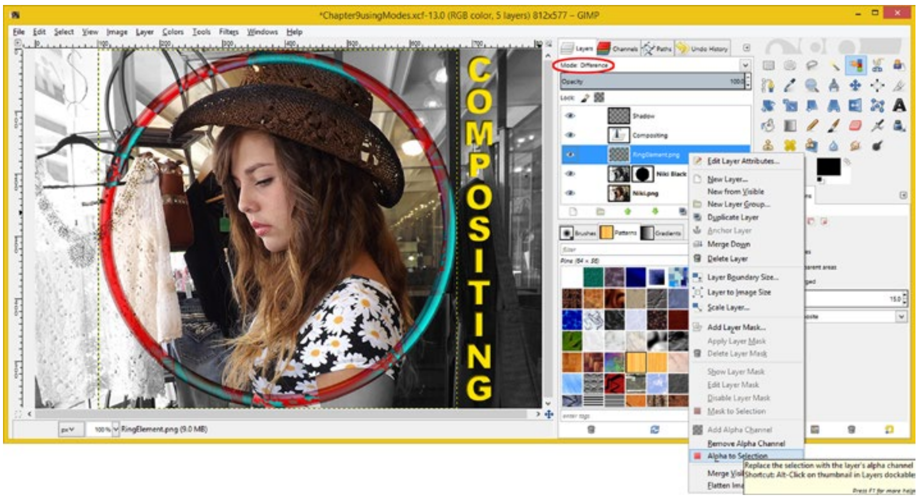
Note that if you have any other layer selected, and then perform the same Add Layer Mask work process, you won't get the desired end result.

To complete the GIMP simulation of what you have done in your Photoshop image composite, you need to add a drop shadow to your RingElement. This will give you more opportunity to experience modal operation. Let's go through GIMP layers, tools, options, selections, masks, modes, and work processes.

GIMP Special Effects: Creating a Ring Drop Shadow

I almost forgot to change the RingElement blending mode to **Difference**, so I will do that now. The blending mode does not affect the actual pixel data, so regardless of what mode we may have that layer set to utilize, you can still create your drop shadow effect to composite it with the layers underneath it.

As you can see at the top of Figure 10-9, I have set the Mode to Difference and selected the RingElement layer so that I can cull a selection set from its transparency areas. This will be accomplished using the **Alpha to Selection** function in GIMP.



Replace the selection with the layer's alpha channel
Shortcut: Alt-Click on thumbnail in Layers dockable
Press F1 for more help

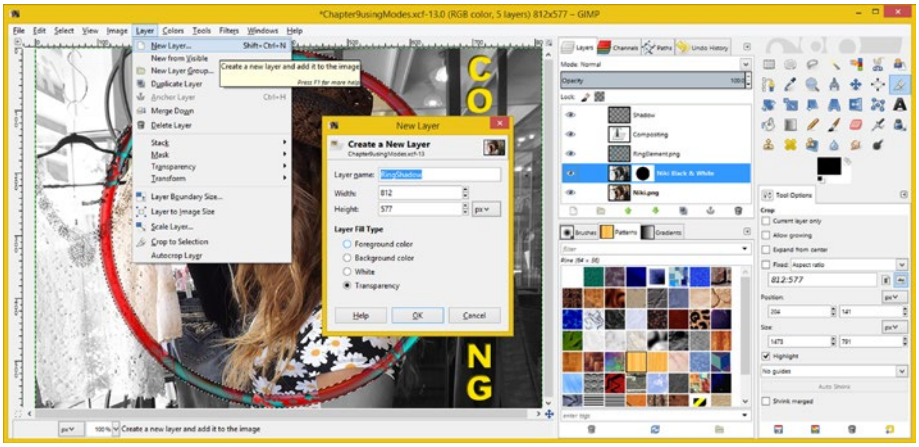
Figure 10-9. *Select a RingElement layer and Alpha to Selection*

To access this feature, right-click on a selected layer. At the bottom of the context-sensitive menu, you should see the **Alpha to Selection** option. If you select this, GIMP will select the area occupied by your ring element.

The next step in your work process (your modal operation sequence) is to create a new layer underneath RingElement, which we will call **RingShadow**, in which to hold your drop shadow pixels.

This needs to be added underneath your RingElement layer, as a shadow logically would be underneath the ring element.

To create it, select a layer underneath RingElement—for this composite, it would be the **Niki Black & White** layer—and follow your **Layer ➤ New Layer** menu sequence, as seen in Figure 10-10, and name it **RingShadow**.



Layer name: RingShadow
Width: 812
Height: 577 px
Layer Fill Type: Transparency

Figure 10-10. To create a new layer underneath the RingElement, start by selecting the Niki Black & White layer and use Layer ➤ New Layer menu sequence

Notice that, due to the modal operation of the digital imaging software, the selection exists independently of the layer, so you can cull, or pull, if you prefer, this selection from the RingElement layer, and then fill it once you have selected your RingShadow layer.

This is an example of how modal operation can be powerful, as well as how to leverage it creatively.

Now, select your new **RingShadow Layer**, shown as selected in Figure 10-11, and use the **Edit ➤ Fill with FG Color** menu sequence to fill the same area as the RingElement (albeit underneath it) with a flat black area of color. You will notice that your ring will turn back to being its blue metal color, as the black will now block the pixel color from the Niki layers. There is currently no need to use the Difference mode, so for now I'm going to change this mode setting back to **Normal**, until we address this problem a bit later on, during this modal operation heavy work process.

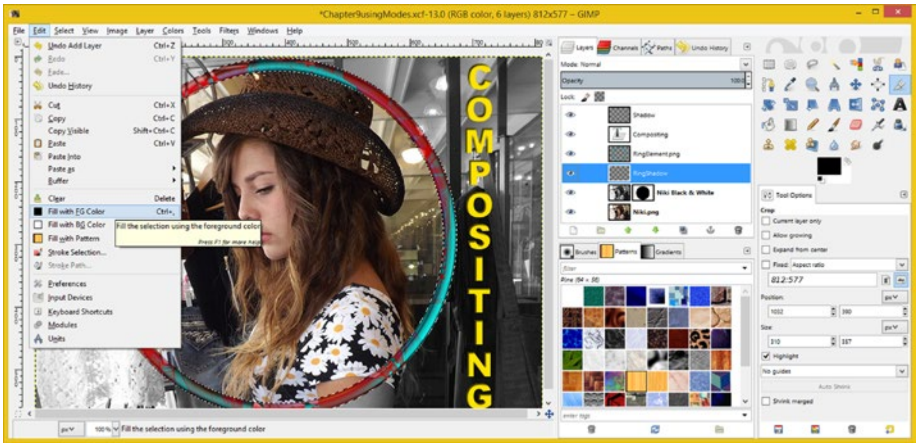


Figure 10-11. *To fill the selection on your new layer with a black color, select your new Ring Shadow Layer and use the Edit ➤ Fill with FG Color menu sequence*

Now we can use one of GIMP's powerful blur functions and create a realistic drop shadow effect underneath the blue ring.

Since you have already utilized the Gaussian Blur, let's use one of the other GIMP Blur algorithms, which can be seen on the submenu shown in the middle of Figure 10-12.

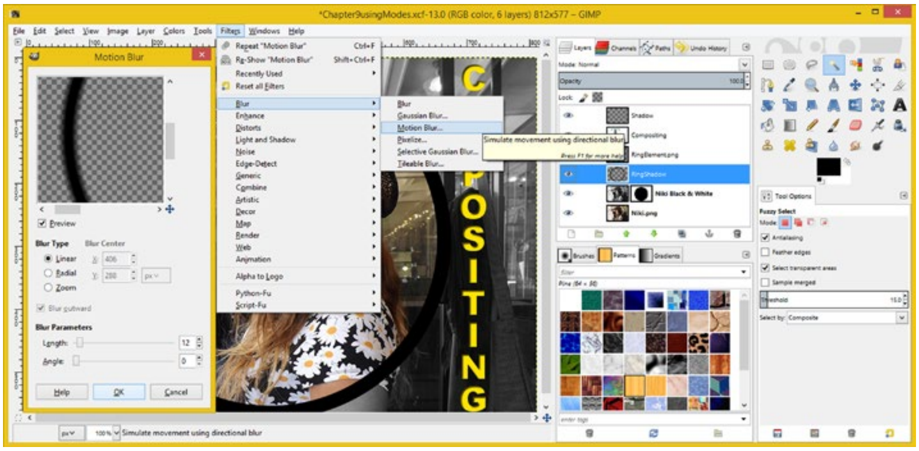


Figure 10-12. *When using your Motion Blur algorithm, follow the Filter ➤ Blur ➤ Motion Blur sequence and set the Length to 12*

I am going to use the **Motion Blur** algorithm for a few reasons: First, it's Hollywood oriented; second, I will not have to move what I blur using the Move tool, as this algorithm will move those pixels for me; and third, I want the blur to run straight to the left, which is generally how a motion blur will be implemented (applied). Also, since the ring element touches the top and bottom edge of this composite, the shadow needs to be directly to your left, so that it is not cut off at the top or bottom of this digital image composite.

With your **RingShadow** layer selected, follow the **Filter ➤ Blur ➤ Motion Blur** menu sequence to open the Motion Blur dialog shown on the left side of Figure 10-12. Then, set an **Angle** of **0**, with a **Length** of **12** pixels of Motion Blur, and click the OK button. The 0–360 degrees starts at 9 o'clock and progresses clockwise, so you want to create your Angle, and the resulting direct left setting, using the value 0.

Now, select the **RingElement** layer, and then the **Normal** blend mode, if you have not done so already, as is shown at the top of Figure 10-13. This will allow you to see the effect of the motion blur shadowing with your solid blue metal ring element in place. You'll notice what a great job this GIMP Motion Blur algorithm can do at creating a directional drop shadow special effect!

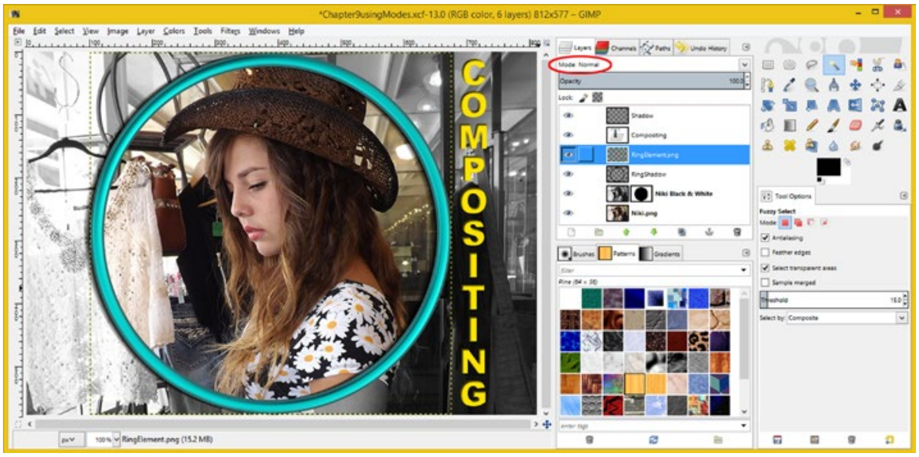


Figure 10-13. *To see the effect of the motion blur shadowing on your solid blue metal ring element, select the RingElement layer and the Normal blend mode*

It is also important to note that we cut out a couple of steps that you did for the Inner Shadow effect work process, as you did not have to use the Move tool to position this effect!

I just noticed that I forgot to apply the Compositing Text layer's colored glass effect, as we did in Photoshop using the Hard Light mode, so next, let's select the **Compositing** text layer and set the **Hard Light** mode.

This will make the vertical text look much better, as can be seen in Figure 10-14.

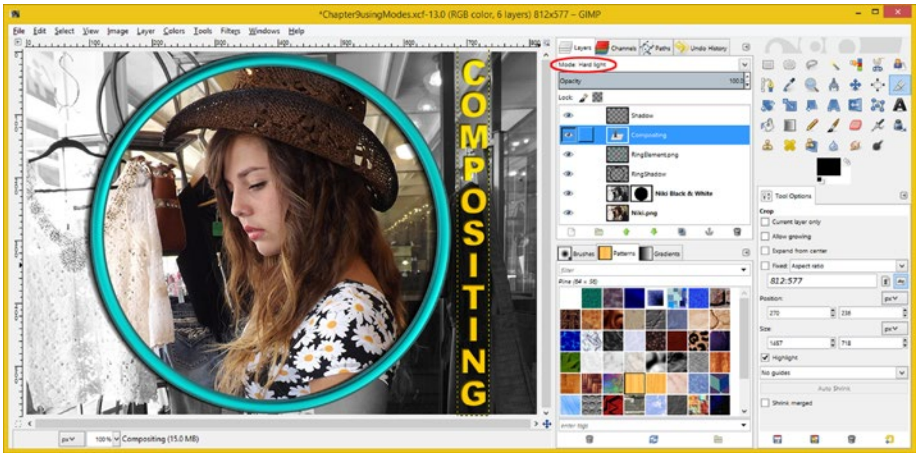


Figure 10-14. *To apply a colored glass effect to the text in GIMP, select the Compositing layer and set the Hard Light mode*

Now you are ready to address this problem of your RingShadow layer blocking the pixels of the Niki layer from the Difference mode algorithm. We need to deal with this as we seek to simulate our Photoshop compositing result here, using GIMP.

To fix the problem, select your **RingElement** layer, then right-click it and again use **Alpha to Selection**. Next, click your **RingShadow** layer to select it, and hit your **Delete** key.

As you can see in Figure 10-15, this process removes the black (unused or unseen) pixels underneath your RingElement. I turned off the layer visibility so you could see this clearly. Now the Difference algorithm can process the Niki layer pixels and your drop shadow effect will still be in place.

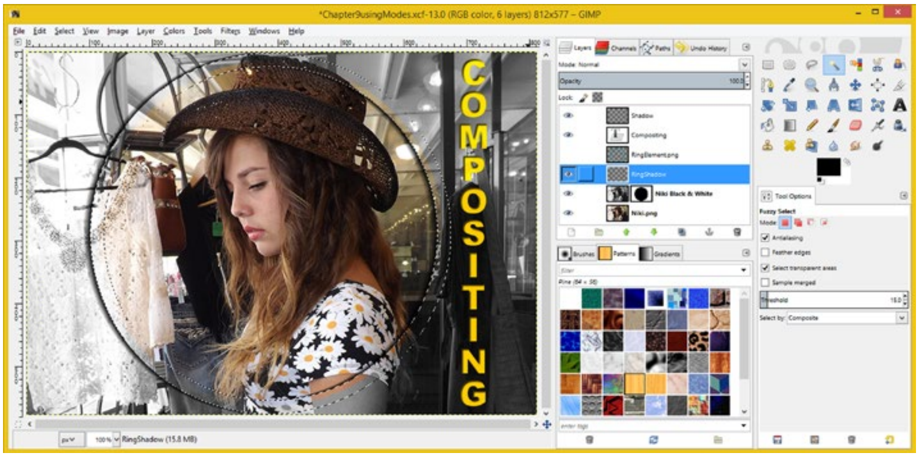


Figure 10-15. *To remove the black (unseen) pixels underneath your RingElement, select the RingShadow layer and hit Delete*

Now, get rid of the selection by using the **Select ➤ None** menu sequence, and let's finish up!

With your RingElement layer selected, set your mode back to **Difference**, which, as you can see in Figure 10-16, now gives you the intended result. You have simulated all the effects you created in four layers in Photoshop using six layers in GIMP.

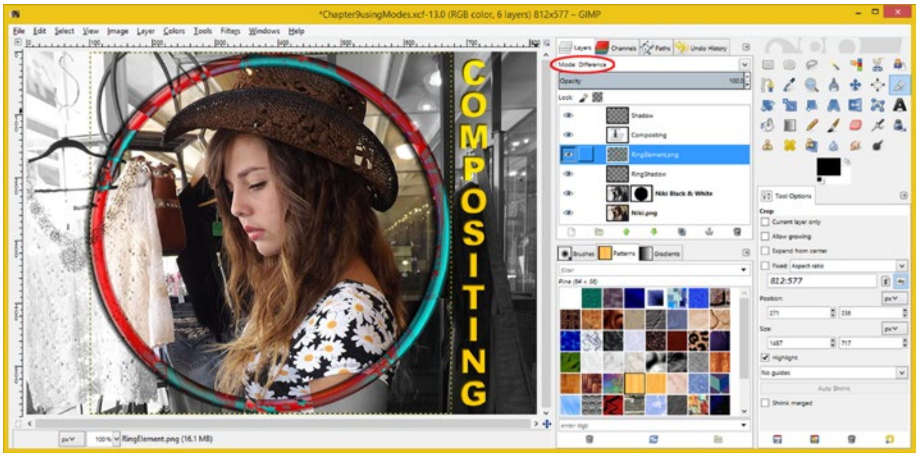


Figure 10-16. *Select the RingElement layer and set the mode to Difference to color the ring element with your special effect*

It is important to observe here, in this layer-by-layer comparison, that two of your Photoshop layers utilize (fx) sublayers. One could look at that as meaning that Photoshop also uses six layers, making the number of layers in the two programs quite similar.

As you can see, both 2D digital image compositing software packages are able to produce stunning visual results, using the proper work process and the correct modal operation sequences (order). This is what I intended to demonstrate in this chapter, and now we can move along to look at **special effects** applications, using the **Filter** menu in both Photoshop and GIMP.

It is important to note that if you want to install **plug-ins** in your digital imaging software, the Filter menu is where you can access these specialized effect algorithms.

Summary

In this tenth chapter, we went over how the **modal operations** in Photoshop and GIMP will greatly affect the success (and failure) of your digital image compositing endeavors. Every once in a while, a mistake in mode settings will also end up producing some very cool unintended results, which you may claim that you intended to create!

We also looked at how digital image editing and compositing software tools, tool options, layers, channels, selections, modes, and menus all work together modally to create your collective end result based on their individual settings.

You saw that complex image compositing is not as easy as it might look from the outside. After we looked at some of the general considerations of modal operation, we created some more advanced compositing with Photoshop.

Later in the chapter, we replicated the same visual results using GIMP 2.8.14, along with different tools, modal operations, and work processes.

You added adjustment layer masking that only colored the pixels inside of your ring element in the RingElement layer, using Photoshop. In order to make your GIMP image composite identical to your Photoshop digital image composite, you then simulated that effect in GIMP and created the drop shadowing effect for your RingElement layer, and the glass text effect for your Compositing text layer.

In Chapter 11, you'll learn more about digital image compositing software's **third party plug-ins** and how to access them along with integrated (permanent) plug-ins, using the Filter menu in both software packages.

CHAPTER 11

The Automation of Digital Imaging: Plug-In Filters

Now that we have gone over the primary fundamental topics in digital image compositing, let's take a chapter to have some fun and take a look at plug-in software. **Plug-ins** can automate specialized imaging effects for you, by allowing you to simply set a number of options and click the OK button in the dialog! These plug-ins are generally found in the **Filter** (Photoshop) or **Filters** (GIMP) menu in your digital image compositing software package.

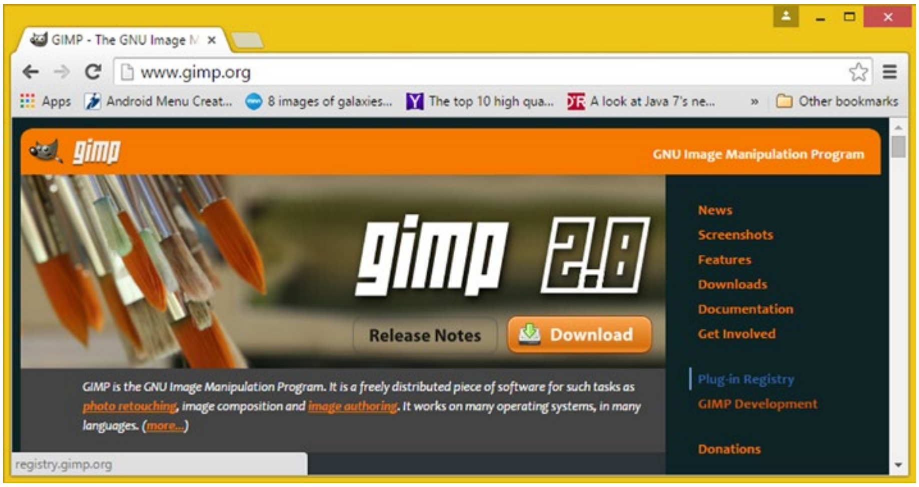
The first thing that we will do, since plug-ins for GIMP are free, is to see how you would find, install, and utilize plug-in filters for the GIMP 2.8.14 software package.

Photoshop also has a plethora of plug-ins, most of which are already a part of the Photoshop CS6 software package, and there are a couple of third-party companies that provide additional plug-ins for Photoshop as well. To find these on the Internet, do a Google search, using the term "Photoshop plug-ins."

After that, we will take a look at how to incorporate the plug-in filter into the digital image compositing work process, and continue to apply special effects using your Chapter 10 image compositing creations, for both Photoshop CS6 and GIMP 2.8.14.

GIMP Plug-In Filters: Enhancing GIMP 2.8

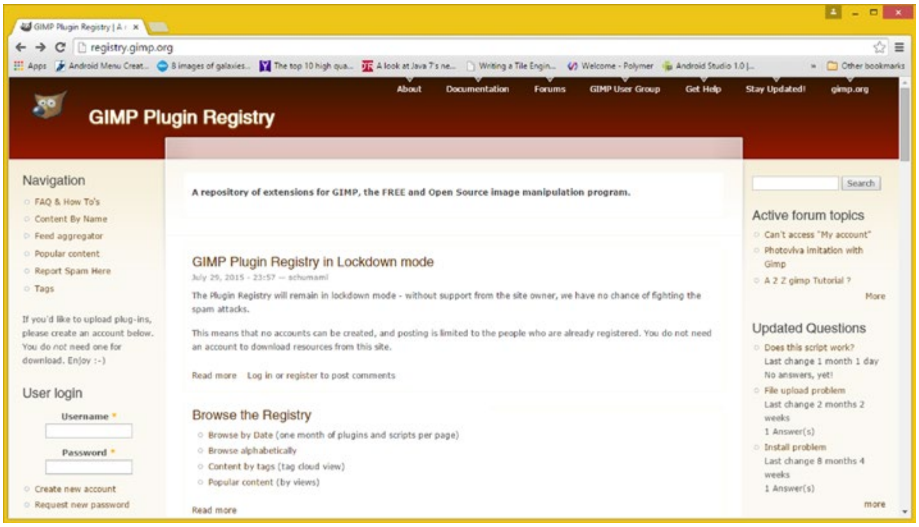
There are dozens of plug-ins already installed in GIMP when you buy the program, including the most popular plug-in filters, under the Filters menu and its many submenus. In this section of the chapter, I am going to show you how to access hundreds more of these plug-in filters from the Gimp.org web site, and then, over the remainder of the chapter, we will take a look at how to use and apply plug-in filters to your digital image compositing work process. If you go to the GIMP home page at gimp.org, you will see a **Plug-in Registry** link in blue on the lower-right-hand side of the home page links, as shown in Figure 11-1. This link accesses the plug-in registry, where you will find hundreds of plug-ins that you can download. These plug-ins are created by people and companies for other users of GIMP to use in their digital image compositing work process. You can also write your own plug-ins using the PyGIMP, Scheme or Script-Fu languages which are a part of GIMP. I will be covering this in Chapter 14.



GIMP is the GNU Image Manipulation Program. It is a freely distributed piece of software for such tasks as photo retouching, image composition and image authoring. It works on many operating systems, in many languages. (more...)

Figure 11-1. *On the right side of the gimp.org home page, you can find a Plug-in Registry link*

After you click the blue Plug-in Registry link, you will be taken to the `registry.gimp.org` page, which is shown in Figure 11-2. Look for the **Browse the Registry** section, which has links allowing you to browse the plug-ins by date, tags, or alphabetically. If you find any plug-ins that you want, click the corresponding link to download the file to your hard drive. Later you can transfer the file to the Gimp/Downloads folder, which I am going to show you how to find on your hard drive next.



GIMP Plugin Registry

Navigation

- FAQ & How To's
- Content By Name
- Feed aggregator
- Popular content
- Report Spam Here
- Tags

If you'd like to upload plug-ins, please create an account below. You do not need one for download. Enjoy :-)

User login

Username *

Password *

- Create new account
- Request new password

A repository of extensions for GIMP, the FREE and Open Source image manipulation program.

GIMP Plugin Registry in Lockdown mode

July 29, 2015 - 23:57 — schumaml

The Plugin Registry will remain in lockdown mode - without support from the site owner, we have no chance of fighting the spam attacks.

This means that no accounts can be created, and posting is limited to the people who are already registered. You do not need an account to download resources from this site.

Read more Log in or register to post comments

Browse the Registry

- Browse by Date (one month of plugins and scripts per page)
- Browse alphabetically
- Content by tags (tag cloud view)
- Popular content (by views)

Read more

Active forum topics

- Can't access "My account"
- Photoviva imitation with Gimp
- A 2 Z gimp Tutorial ?

More

Updated Questions

- Does this script work?
  Last change 1 month 1 day
  No answers, yet!
- File upload problem
  Last change 2 months 2 weeks
  1 Answer(s)
- Install problem
  Last change 8 months 4 weeks
  1 Answer(s)

more

Figure 11-2. *In the Browse the Registry section on the registry.gimp.org page, you can browse the plug-ins by various criteria*

To install a plug-in, simply place it into your **GIMP Plug-Ins folder** and restart the program. It will then load into system memory. It is important to note that the Plug-Ins folder is not under your `C:/Program Files/Gimp` 2 folder but is instead located in the Users subfolder. On my machine, the path to this folder is `C:/Users/Default.Default-PC/.gimp-2.8/plug-ins`, as seen in the navigation bar in Figure 11-3.

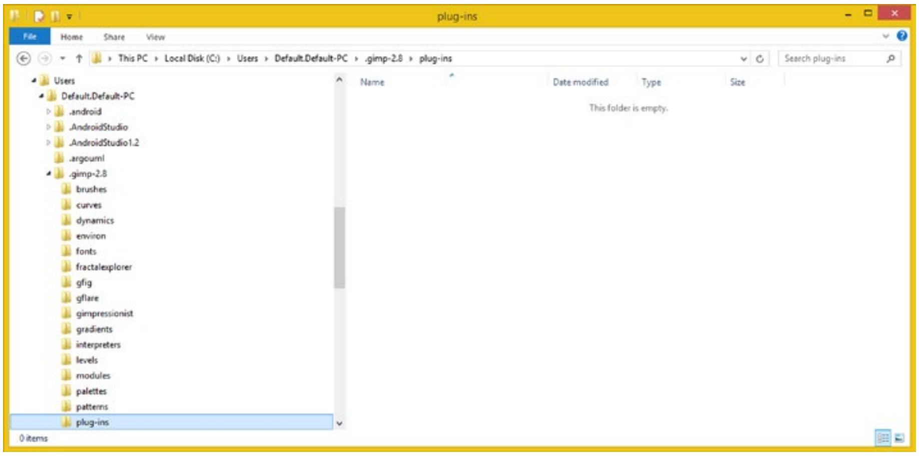


Figure 11-3. *You can find the Plug-Ins folder in the Users subfolder, which, for my computer, is at* `C:/Users/Default.Default-PC/.gimp-2.8/plug-ins`

Since I just installed GIMP 2.8 for the book, the folder is empty, as the plug-ins that are a part of GIMP are no longer stored in it. In this chapter, I'm going to cover plug-ins that are a part of GIMP 2.8.14, as these have been selected and approved by the GIMP software development team.

GIMP's Filters Menu: Using Plug-In Filters

Let's get right down to business and start making our digital image compositing project have even more visual impact than it has already. We will do this by applying filters to the backplate, or background image if you prefer that term, and I will also show you how to do this using a non-destructive work process. In this chapter we will start with the GIMP Filters menu, and then finish up by taking a look at the Photoshop Filter menu. Both packages have a significant number of impressive, and powerful, plug-in filter offerings that you will enjoy exploring and using to enhance your artistic creations.

Start by opening your **Chapter10modalOperation.xcf** native GIMP format file and left-clicking the Niki Black & White layer to select. Right-click and choose **Duplicate Layer**, to duplicate the layer, and to implement your non-destructive editing, as seen in Figure 11-4.

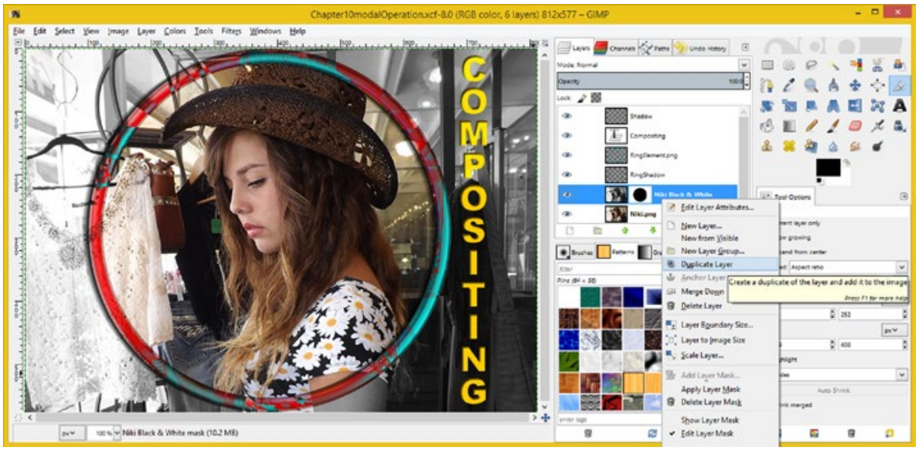
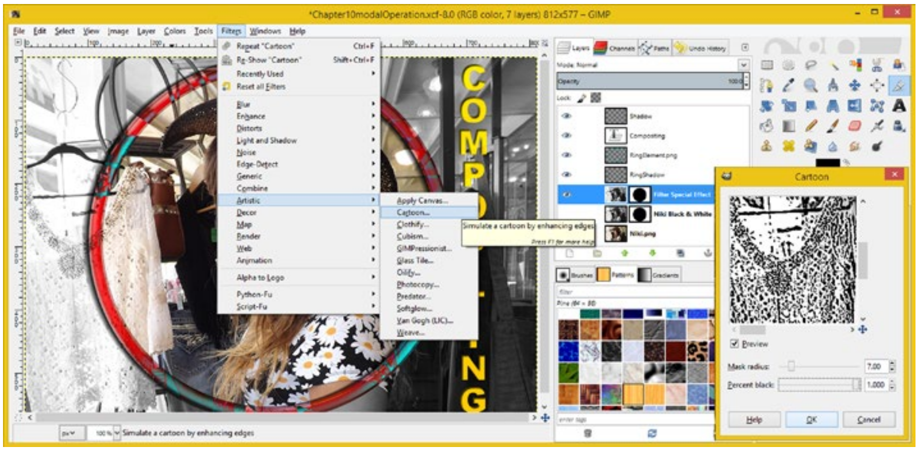


Figure 11-4. To use Plug-In Filters, first select the Niki Black & White layer, then right-click on the layer and select the Duplicate Layer menu option

Notice that along with the black-and-white version of Niki, your Layer Mask (alpha channel data) is also copied, so that we can apply a cool pixel processing effect to the background, behind the ring, and keep the photograph of Niki in color by using the Layer Mask. First, I double-clicked the layer name (copy), renaming it "Filter Special Effect." With this new layer selected, I next used the **Filters ➤ Artistic ➤ Cartoon** menu sequence to access a **Cartoon** dialog, where I set a **7** pixel mask radius, and 100 percent black value of **1.000** (which is 100 percent using floating point math terms), as shown in Figure 11-5.



Mask radius: 7.00
Percent black: 1.000

Figure 11-5. To access the Cartoon dialog, I selected a layer and used the Filters ➤ Artistic ➤ Cartoon menu sequence

If you select the **Preview** check box in any Filter dialog, you will be able to see the effect of that filter's settings. You can also drag the preview window area to show areas of the image you are interested in previewing. In this case, it was the lace blouse that I wanted to preview, because I wanted to bring it out (make it "pop") by using this Cartoon filter.

Notice the white outline in the selected layer in both Figures 11-5 and 11-6, showing the respective preview icon that I selected for the filter to operate on (and use for a preview) before I invoked the filter. This is another example of a **modal operation**, as the filter looks to a selected layer preview icon, to see whether to operate on the layer image or the layer mask.

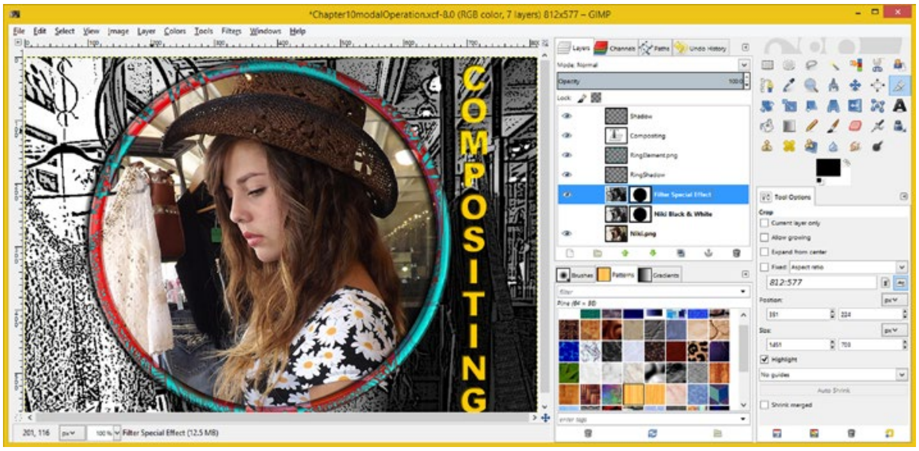


Figure 11-6. *A masked cartoon effect greatly enhances the image background outside the ring*

This also lends more flexibility to your special effects application, as you can apply any of these filters to either an image or a mask (alpha channel), which allows far more artistic latitude when you want to create visually impressive special effects.

Let's now take a look at how to use plug-ins in Photoshop. The plug-in filters are one of the features that are different between GIMP and Photoshop. Filters are essentially image special effects algorithms. You will not be able to get exactly the same **background outside of the ring effect** in both software packages.

However, you can get the same result by using Photoshop and GIMP at the same time! That's right—I do it all the time, by creating a computer **clipboard** feature to transfer pixel arrays back and forth between the different layers in the programs, using the following modal operations (1 and 2 apply to the source; 3 and 4 apply to the target):

1. Select the layer containing filtered effect
2. **Select ➤ All**, and then: **Edit ➤ Copy**
3. **Layer ➤ New Layer** (bottom of layer stack)
4. **Edit ➤ Paste** (with the new layer selected)

I'm hoping that this book will assist you in thinking outside of the "digital image compositing box" a little bit!

Photoshop's Filter Menu: Using Plug-In Filters

Let's simulate the same compositing process that we just completed in GIMP, in Photoshop. Since the adjustment layer does not have a filter mask, we will need to change the set-up for our adjustment layer, essentially reversing it, which will show you some new modal operation "moves," as I call them, to set the Photoshop composite up to be able to do what we were doing in the GIMP composite.

Start your modal work process by opening up your Chapter 10 Photoshop file, `Chapter10modalOperation.psd`, and selecting the back plate (bottom layer) background image layer. Then use your **Layer ➤ Duplicate Layer** menu sequence to access the Duplicate Layer dialog, shown in Figure 11-7, and name your new layer, "**Plug-In Filter Effect**."

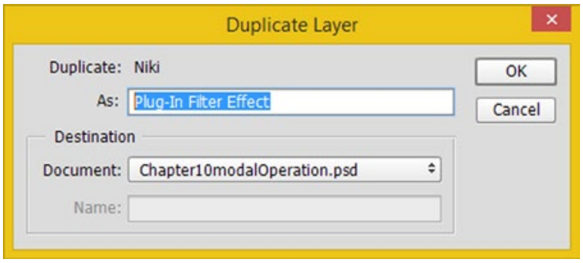

Duplicate Layer
Duplicate: Niki
As: Plug-In Filter Effect
Destination
Document: Chapter10modalOperation.psd
Name:

Figure 11-7. *To start the compositing process in Photoshop, follow the Layer ➤ Duplicate Layer menu sequence to access the Duplicate Layer dialog*

The second step in this work process, to simulate what you have done with GIMP, is to **drag** this Plug-In Filter Effect layer to the **bottom** of your layer stack, thereby making it the backplate, and guaranteeing that when you apply the plug-in filter to it, it will be **behind** or **underneath** everything else that is in the compositing pipeline (the pixel processing order).

The next step is to drag the **adjustment layer mask icon** out of (from) the orange adjustment layer, and drop it on your **Niki** layer, which should now be directly underneath your adjustment layer, as shown in Figure 11-8. You will see a **white outline** around the Niki layer when your icon is over the layer correctly. When you see this, **drop** the layer mask icon. This will change what this mask is referred to, from the adjustment layer mask, to being a **layer mask**.

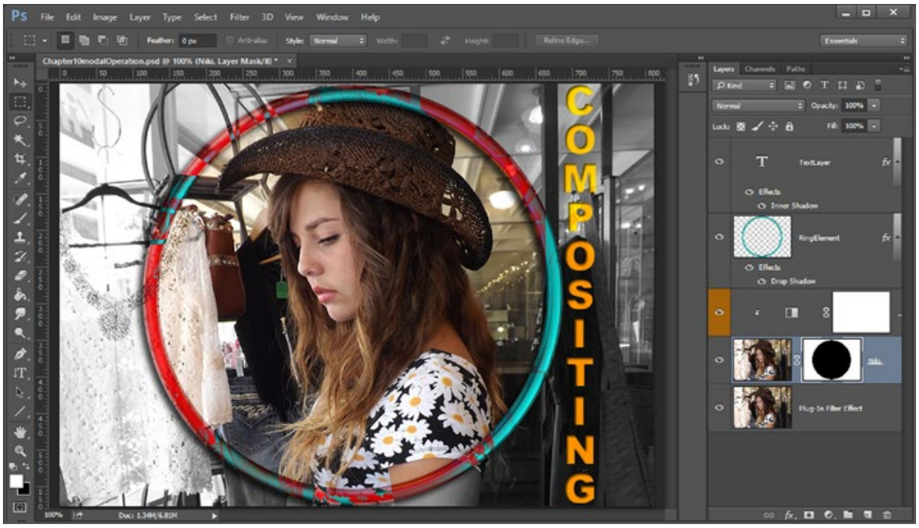


Figure 11-8. *The second step of the compositing process is to drag the adjustment layer mask onto the Niki layer*

The adjustment layer mask is now back to being all white, as you need it to be to apply adjustments to the entire backplate. Notice that Photoshop has more advanced drag and drop capabilities, such as this ability to simply drag layer mask data between layers, that GIMP 2.8.14 does not yet feature. There is always the chance that GIMP 3.0 will add these capabilities at some later date in the software development cycle.

The next step in the work process is to **drag** the orange adjustment layer so that it goes **underneath** your Niki layer, but over the Plug-In Filter Effect layer, as can be seen in Figure 11-9. Since this adjustment is now only applied to the backplate, you don't need the **clip to layer** option, which attaches (isolated the adjustment) only to layers which are directly underneath it.

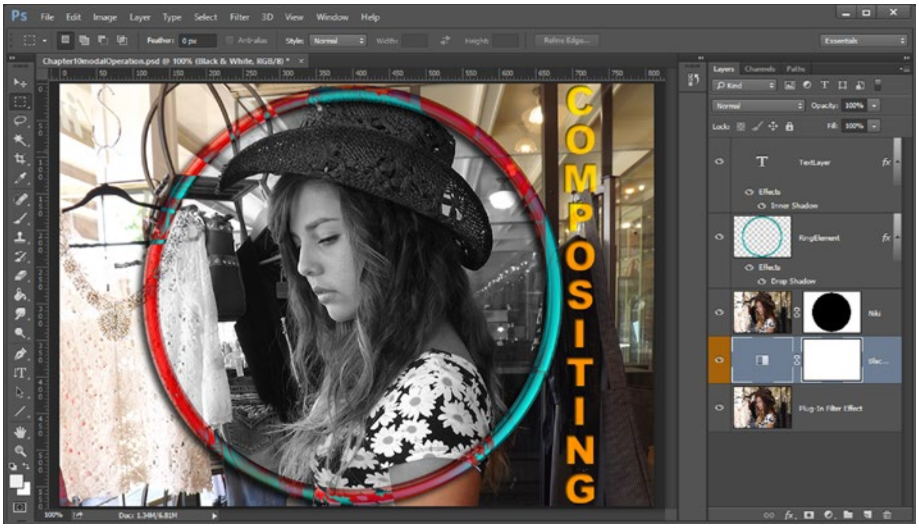

File Edit Image Layer Type Select Filter 3D View Window Help
Chapter10modalOperation.psd @ 100% (Black & White, RGB/8) *
COMPOSITING
Layers Channels Paths
TextLayer
Effects
Inner Shadow
RingElement
Effects
Drop Shadow
Niki
Plug-In Filter Effect

Figure 11-9. *The next step is to drag the adjustment layer underneath your Niki layer*

As you can see in Figure 11-9, the desired effect is the opposite of what we need. If you want the effect, you will need to keep the Niki layer mask as is; if you wanted the effect you created in GIMP, you would need to invert the layer mask.

The work process to accomplish the objective will involve selecting the **Niki layer**, then the **layer mask icon**, within your Niki layer, indicating to Photoshop that this is the data you wish to submit for image adjustment processing.

Remember, using the modal operation is the key to the successful usage of digital image compositing software. In this case, this would involve the selection of the layer mask component of the Niki layer before you invoke your Invert Adjustment Mask operation in Photoshop CS6 by selecting the Adjustment Mask and inverting it, which we'll do next.

Again, you can see that the Niki layer is selected using the gray color, as is shown in Figure 11-10, and that the layer mask is selected using a white outline with four corner squares.

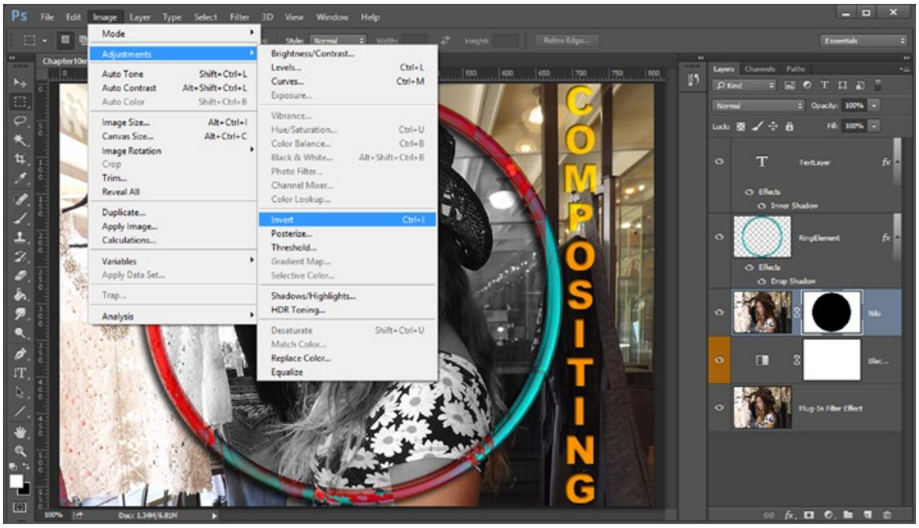


Figure 11-10. *To invert the layer mask, select Niki Layer Mask and invert the layer mask pixels*

Next, you will use the **Image ➤ Adjustments ➤ Invert** menu sequence, also shown in Figure 11-10, to invert the selected layer mask. This will result in the outside of your RingElement layer object being changed to black and white, while the inside becomes color, due to the reversal of what the layer mask is telling the adjustment layer to process and display.

As you can see in Figure 11-11, the desired effect is now in place, and you can now select your **Plug-In Filter Effect** layer and use the **Filter ➤ Filter Gallery** menu sequence to access the Photoshop CS6 artistic effects plug-in filter collection.

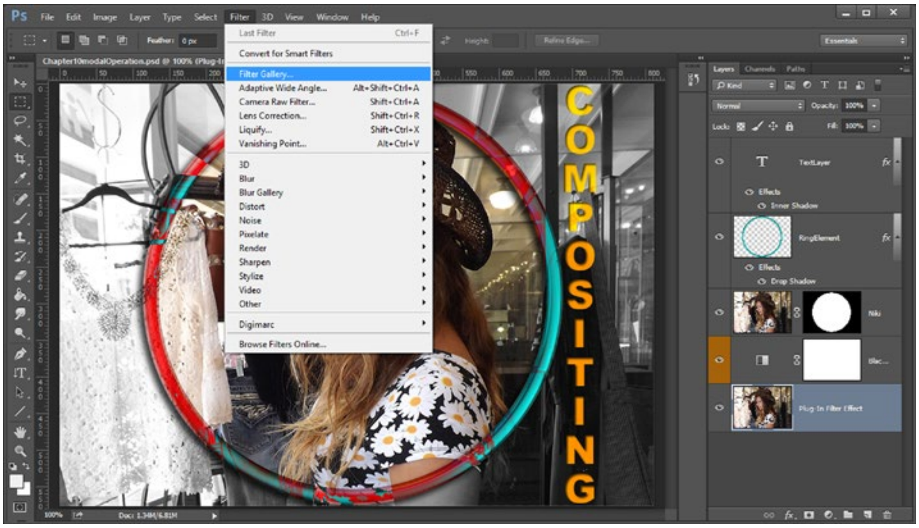


Figure 11-11. *To access the artistic effects plug-in filter collection, select the backplate and follow the Filter ➤ Filter Gallery menu sequence*

The Filter Gallery plug-in collection, at one time, consisted of individual plug-in filters, all accessed using the Photoshop Filter menu. These have now been consolidated into one user interface, which can be seen in Figure 11-12. I selected the **Brush Strokes** section, and then clicked an **Ink Outline** filter to apply it.

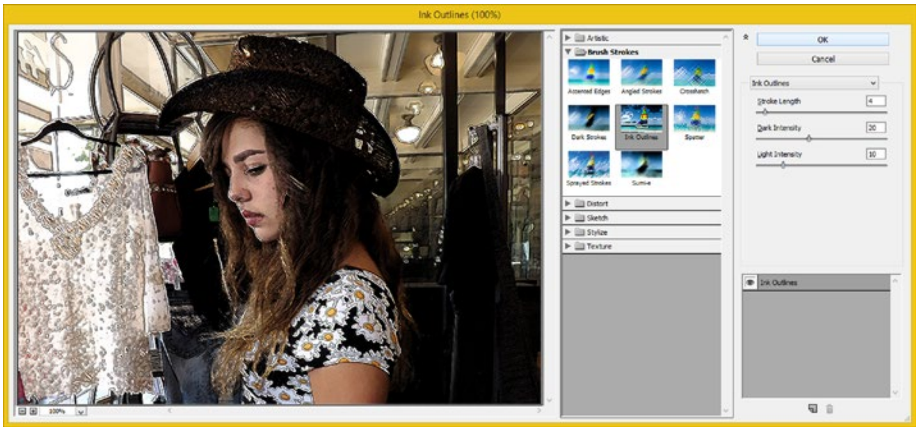


Figure 11-12. *Results of selecting the Brush Stokes section and clicking the Ink Outline filter to apply it*

As you can see, in Figure 11-13, although not identical, you have achieved a similar visual upgrade to your compositing project as you did in GIMP: the outside of the ring element with the text is stylized and the inside is original photographic quality imagery.

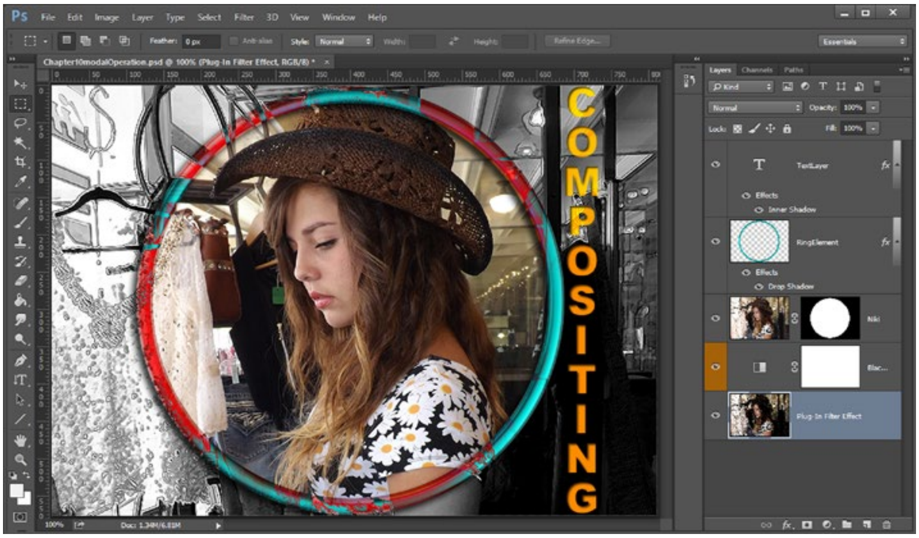

File Edit Image Layer Type Select Filter 3D View Window Help
Chapter10modalOperation.psd @ 100% (Plug-In Filter Effect, RGB/8) *
COMPOSITING
Layers Channels Paths
TextLayer
Effects
Inner Shadow
RingElement
Effects
Drop Shadow
Niki
Plug-In Filter Effect

Figure 11-13. *Your visual upgrade will result in the backplate being differentiated using a special effect*

In the next chapter, we will take a closer look at your compositing pipeline for this project, and what it is doing, as well as further enhance the image composite and learn some new software features in both Photoshop CS6 and GIMP 2.8.14.

Summary

In this eleventh chapter, we went over the **plug-in filters** in Photoshop and GIMP and how they can be used as “miniature programs within a program” to add even more special effects wow factor to your digital image compositing pipeline and projects. You learned about GIMP’s plug-in repository and then implemented special effects, using filters inside both GIMP and Photoshop.

In GIMP, you again implemented non-destructive editing by adding another layer that you can use for applying the special effects filter. This time, the work process (modal operation) was easier, based on how GIMP works, as you simply had to duplicate one single layer that conveniently includes the layer mask that you need.

In order to achieve similar image compositing processing pipeline results in Photoshop, you had to change your layer, layer mask, adjustment layer, and adjustment layer mask processing pipeline around. This showed you how to use some really useful new moves, work processes, and artistic filter features in Photoshop CS6.

In Chapter 12, we will learn more about digital image compositing software pipelines and what exactly is going on under the hood, so you can see why the modal operation is important.

CHAPTER 12

The Work Process of Digital Imaging: Compositing

Now that we have had some fun using plug-in filter software, which you could probably experiment with for the rest of your life, it's time to get back into some more advanced "under the hood" thinking about how the digital image compositing pipeline works, and what it is doing in system memory. This is important if you are also going to leverage digital image compositing concepts and techniques in this book in your new media programming projects using popular widespread platforms such as Java, JavaFX, Android, or HTML5. Since I'm a programmer, I tend to think along these lines anyway, and with complex 2D and 3D software packages, knowing what is happening at the core processing level can be an advantage and help you to understand what the software tool is doing (and what it is going to do).

Just as a good programmer can write "dense" code, a good image compositor can get more mileage using fewer layers in the compositing pipeline (layer stack). We will take a look at this during the chapter, as well as further enhance the professional compositing project end result that we are trying to achieve. I wanted to make sure you have a relatively complex digital image compositing project in place, by the end of the book. Even though this is a fundamentals level book, I wanted to be sure and give you your money's worth, in both information, and in experience.

Let's again start with GIMP for this chapter, just so we can even the score, since we've been starting with Photoshop CS6 for most of this book. You will continue to refine your visuals for this composite by leveraging features of the existing layer hierarchy, which we have not yet enabled, to see how much pixel pipeline processing can be achieved by using only seven layers. I think you will be amazed at what you can achieve, using only seven compositing layers in both Photoshop CS6 and GIMP 2.8.14.

GIMP Compositing: Using Layer Features

To begin digital image compositing in GIMP 2.8.14, open up the **Chapter11pluginFilter.xcf** file, as shown in Figure 12-1, and select the Filter Special Effect layer, which currently contains special effect pixels and a layer mask. Add a blending mode to this layer by changing the Normal mode (signifying that no blending mode is enabled) to a **Value** blending mode (color).

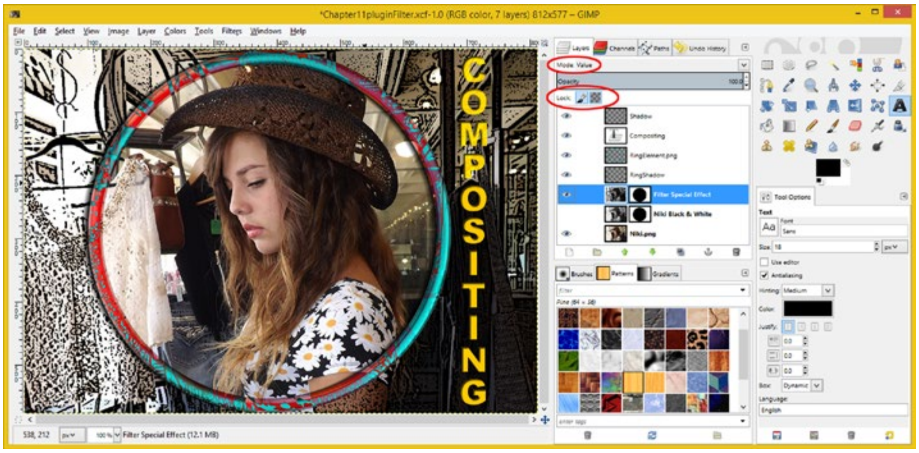

*Chapter11pluginFilter.xcf-1.0 (RGB color, 7 layers) 812x577 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
COMPOSITING
Layers Channels Paths Undo History
Mode: Value
Opacity 100.0
Lock:
Shadow
Compositing
RingElement.png
RingShadow
Filter Special Effect
Niki Black & White
Niki.png
Brushes Patterns Gradients
Tool Options
Text
Size 18
Use editor
Antialiasing
Hinting Medium
Color
Justify
Box Dynamic
Language English
538, 212 px 100 % Filter Special Effect (12.1 MB)

Figure 12-1. *To learn about layer feature usage in GIMP, start by selecting the Filter Effect layer and setting the Value mode*

What this does, from the digital image compositing pipeline perspective, is to take the color values from the Niki.png layer, underneath the Filter Special Effect layer, and use the layer mask to apply these color values to the areas of the special effect layer that were white.

Since the Niki Black & White layer is excluded from the pixel processing pipeline, because the eye (visibility) icon is turned off, the compositing engine will consider the color Niki layer to be underneath the Filter Special Effect layer. Thus, the visibility icon will not only affect what you see, but also **what is processed** by an imaging software's internal compositing engine.

If you want to make the compositing engine reprocess the pipeline without the color data, simply block the color data by turning on the visibility of your Niki Black & White layer.

You can also lock the mask and pixel data for the Filter Special Effect layer if you are finished working with it, using the **Lock: Brush** and **Lock: Alpha** icons at the top of your Layers palette. These are modal icons, and as such you must select the layer you want to lock, as shown in Figure 12-1, and then click either (or both in this case) of these icons to lock your layer data. As you select different layers, they will turn blue if they're enabled or remain white if they're not.

Since your composite is getting kind of "busy" using all of these effects, let's set the RingElement layer back to using **Normal** (no blending) and color shift the metal color to **Wine** to better complement the golden glass text layer effect.

To accomplish this, select the **RingElement** layer and set the mode to **Normal**, as shown in Figure 12-2, to get back the cyan metal ring. Next, use the **Colors ➤ Hue-Saturation** menu sequence to open the **Hue-Saturation** dialog. Shift your Hue **140** degrees around the color spectrum, creating a nice wine color, to offset your gold text.

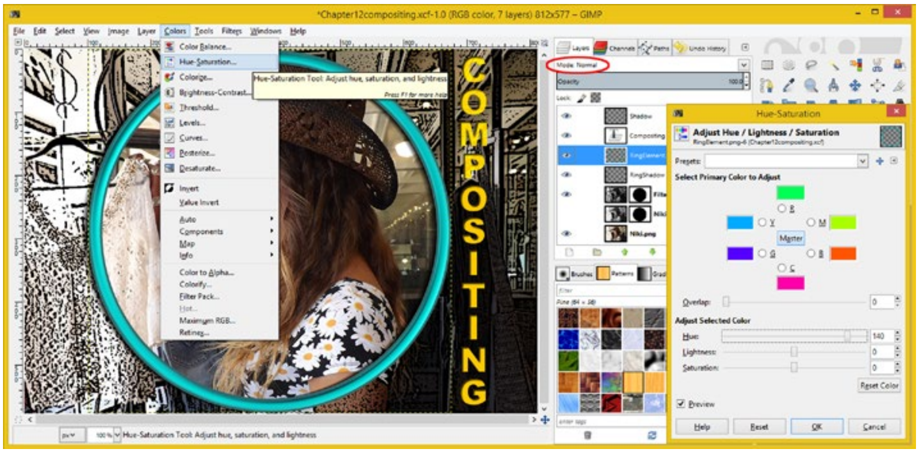


Figure 12-2. *To tone down the colors in your image, set the RingElement to Normal mode and shift the Hue 140 degrees*

I attended USC, so this color combination is especially appealing. The next issue with this composite that you need to address is the realism of the drop shadow that's underneath the RingElement. To add a subtle level of darkening with a blending mode, select the **Darken Only** mode, seen in Figure 12-3, and set the Opacity to **70%**. This will lighten the shadow layer and make it look much more realistic.

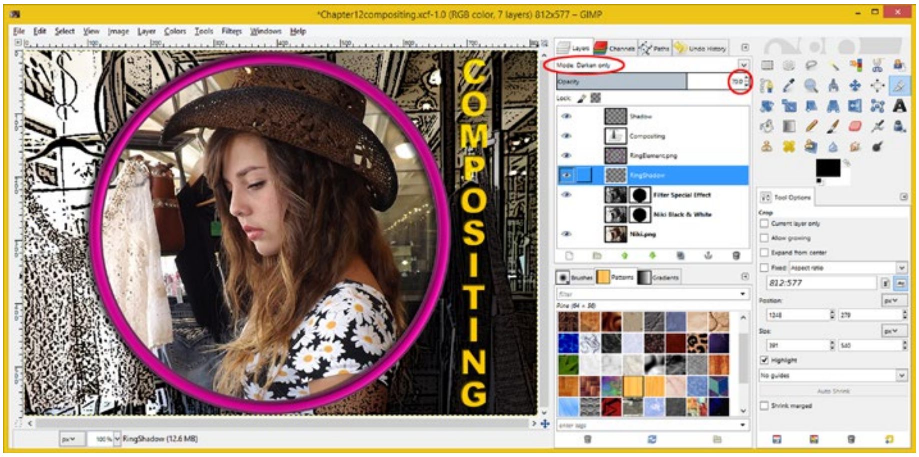
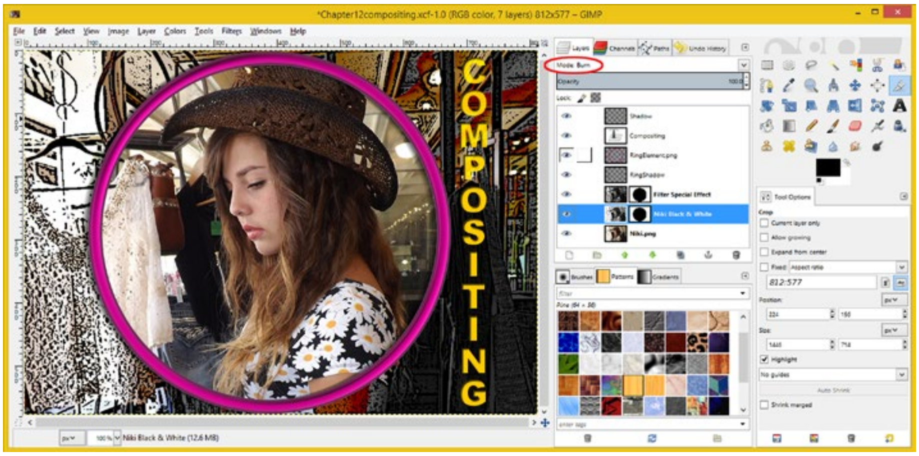


Figure 12-3. *To add a subtle level of darkening, set the RingShadow to use the Darken Only mode, and 70% Opacity*

The Opacity can be set by using a numeric spinner, or by dragging the vertical bar to the right of the gray area, inside of the Opacity level indicator bar.

Since we're leveraging our existing layer pipeline here, let's turn the Niki Black & White layer back on and use it for more **detail enhancement** in the ring background special effect.

Then, select the **Niki Black & White** layer and set a **Burn** mode, as shown in Figure 12-4. This will create a stained glass effect, by filtering the color backplate pixels, before passing them to the Value mode for further processing inside the Filter Special Effects layer.


*Chapter12compositing.xcf-1.0 (RGB color, 7 layers) 812x577 – GIMP

Mode: Burn

Opacity 100.0

Layers: Shadow, Compositing, RingElement.png, RingShadow, Filter Special Effect, Niki Black & White, Niki.png

Figure 12-4. *To create a stained glass effect, select the Niki Black & White layer and select the Burn layer blending mode*

Now you are starting to see how a fairly complex pixel-processing pipeline can be set up with a modern-day compositing engine (via software) using all of the things I've been adding on, chapter by chapter, during the course of this book.

It is important to note that you can also do all of this digital image compositing pipeline pixel processing in code as well, using Android Studio, Java JDK, JavaFX 8, HTML5, iOS, C#, JavaScript, and even CSS3.

Next, let's add even more processing to your current compositing processing pipeline and tone down the stained glass effect that you got in the previous step. We will do this by again using the **Opacity** engine (slider) component of the image compositing pipeline.

This time, we will use the Opacity capability of the layer to "blend" the previous special effect with the new special effect, giving us another way to fine-tune our result.

This will give us a visual compromise between the two effects we are seeking, which should give us the perfect visual end result.

To start, make sure the Niki Black & White layer is still selected and drag the Opacity slider to the left to a setting of **60%**, or enter **60.0** in the numeric entry component of the user interface element, shown circled in red at the top right of the Layers tab in Figure 12-5.

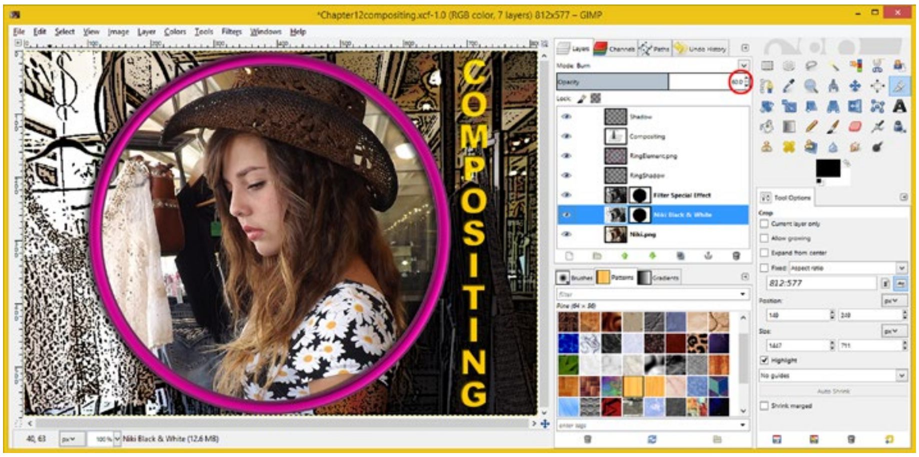


Figure 12-5. You can subdue the Burn mode by setting the layer Opacity to 60%

This will essentially “mix” the two layers together and give you the result seen in Figure 12-5. The image has some of the enhanced colors created by using the Burn mode in your Niki Black & White layer, 60 percent of which then get processed by a Value mode implemented using the Filter Special Effect Layer mode.

Now let’s take a look at the compositing pipeline process from the bottom of the layer (back plate or background image) up, and see what is really going on here. **Niki.png** is the backplate PNG24 image, where the Normal blending mode (no blending) and 100% Opacity is set.

The **Niki Black & White** layer processes a black-and-white pixel image through the white pixels in a layer mask. This allows the color pixels underneath it to pass through the black areas of the mask and applies the **Burn** mode algorithm to the color back plate pixels, again using only the white areas of the mask.

Your **Filter Special Effect** layer processes 60 percent of the Burn mode using a **Value** mode with 100% strength (Opacity), using the same mask and passing the result up through the **RingShadow** layer. Most of this layer is transparent (an alpha channel). It applies a shadow effect using a few motion-blurred pixels that are processed using a **Darken only** mode set at a 70% strength, or Opacity, and passes these up to the **RingElement** layer.

The RingElement layer uses an alpha channel to overlay a hue-shifted wine-colored metal ring over the bottom four layers with no blending mode and 100% Opacity, or pixel strength.

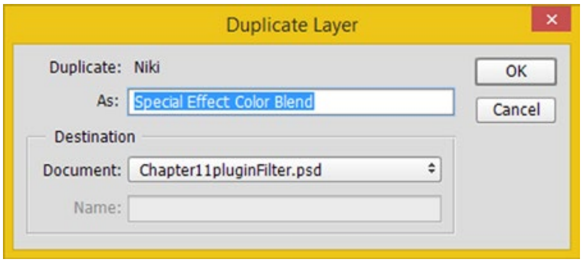
The **Compositing** text layer also uses an alpha channel to overlay text on the bottom five layers using a **Hard Light** blend mode and 100% Opacity, or pixel strength.

A **Shadow** drop shadow effect layer uses an alpha channel and a **Multiply** mode, with 100% Opacity, or strength, to overlay the Inner Shadow effects on top of the Compositing text layer.

Try to visualize these layers and their alpha channel, or mask, blending mode, Opacity setting, and pixel data, as being processed through a digital image algorithm compositing pipeline. Forming a mental image of this process may help you to understand what's happening under the hood, and will help you to become a better digital image compositor.

Photoshop Compositing: Using Layers

Since Photoshop has a different codebase and compositing pipeline features than GIMP, we have to approach adding these new features to your digital image composite in a different way. The first thing that you want to do is to duplicate the **Niki** layer, along with its layer mask, by selecting it and then using the **Layer ➤ Duplicate Layer** menu sequence. This will open the **Duplicate Layer** dialog, seen in Figure 12-6, where you will name the layer **Special Effect Color Blend**, which is what the layer function will provide once you set-up the layer. Then, click the **OK** button and create the new layer, to control your fine-tune color blending for your special effect.


Duplicate Layer

Duplicate: Niki

As: Special Effect Color Blend

Destination

Document: Chapter11pluginFilter.psd

Name:

Figure 12-6. *Name your layer, "Special Effect Color Blend" using the Duplicate Layer dialog*

You want to control color saturation blending outside of your Ring, so invert the layer mask, as shown in Figure 12-7.

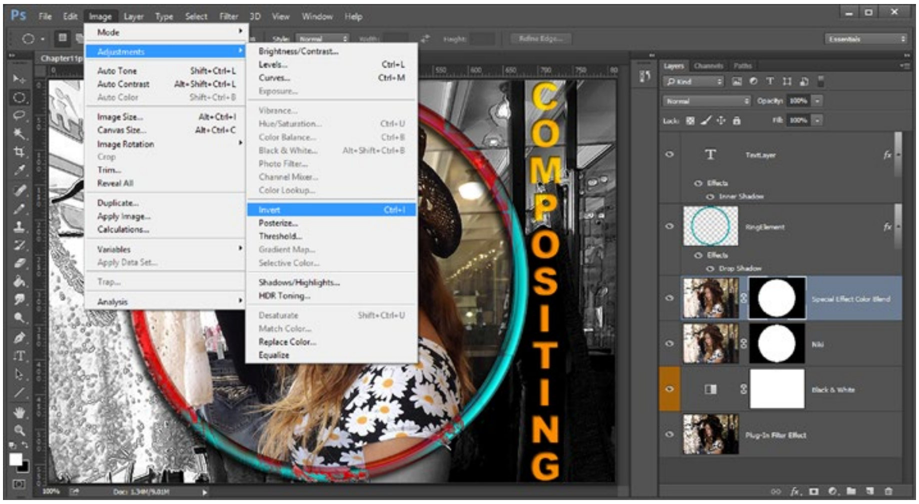


***Figure 12-7.** To control color saturation blending outside the ring, invert your Special Effect Color Blend layer mask*

Remember that you can select the layer mask inside of a specific layer, as can be seen in Figure 12-7, using the white outline, featuring four corner areas. After you select this and use the **Image ➤ Adjustments ➤ Invert** menu sequence, you will see that your layer mask has been inverted, as shown in Figure 12-8, and that it is now affecting the outside area of your Ring Element, instead of the inside area. If you were to leave this layer at the Normal blending mode, with 100% Opacity, using this layer would not be necessary, as you would simply manipulate your adjustment layer to blend color in. You are going to attach an advanced mode and opacity to it later on, to simulate what we did using GIMP 2.8.14 earlier in the chapter.

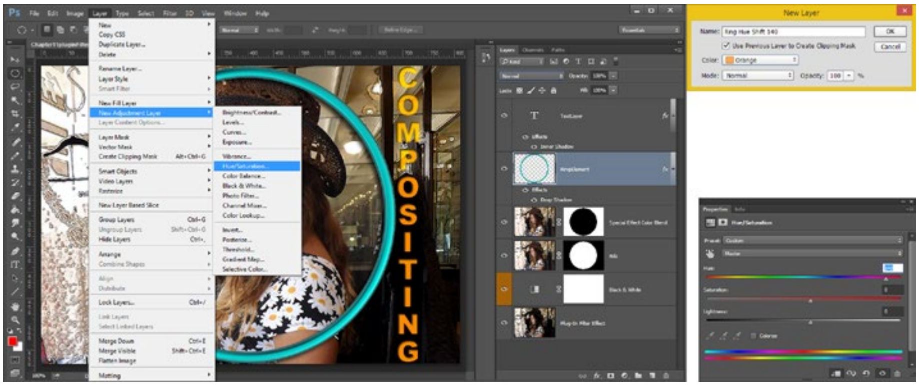


***Figure 12-8.** Add an adjustment layer to the RingElement layer*

The next steps you need to take are to select the **Ring Element** layer, set the **Normal** mode, and color shift the color to be a **Wine** color. This is done non-destructively in Photoshop using an adjustment layer. Use the **Layer ➤ New Adjustment Layer ➤ Hue-Saturation** menu sequence, as is shown in Figure 12-8.

Name the new layer **Ring Hue Shift 140** (referring to 140 degrees) and set the orange color that we're using to denote adjustment layers.

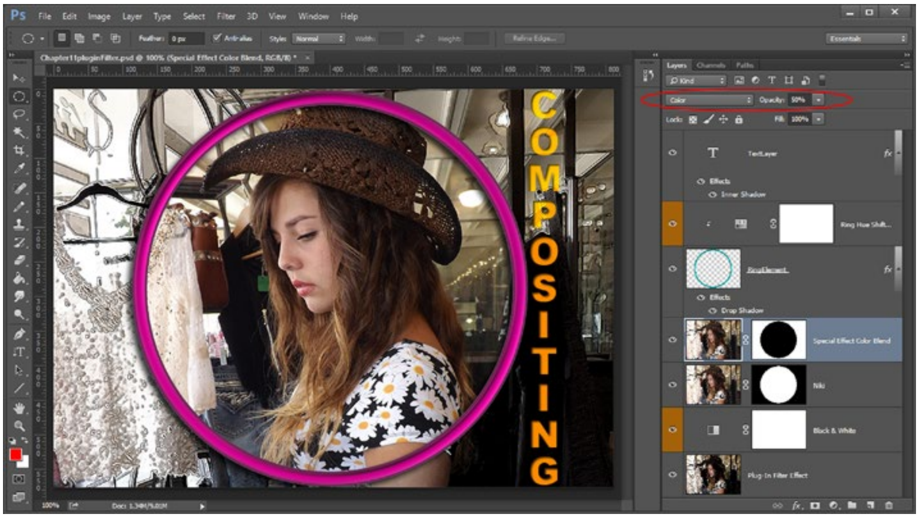
Be sure not to color layers just because it is fun! You should have a good reason; in this case, we're highlighting the adjustment layers by using the color orange.

Photoshop's Hue-Saturation dialog is similar to GIMP's, allowing you to drag the slider to a setting of 140 or enter the value directly into the numeric text data entry field on the right hand side.

Next, select the **Special Effect Color Blend** layer again and set a **Color** (Value) mode, which GIMP calls a "Value" mode, to have an Opacity value of **50%**.

This will decrease the saturation that you see on the outside of your RingElement in Figure 12-8 and give you a more subtle (and more professional) result.

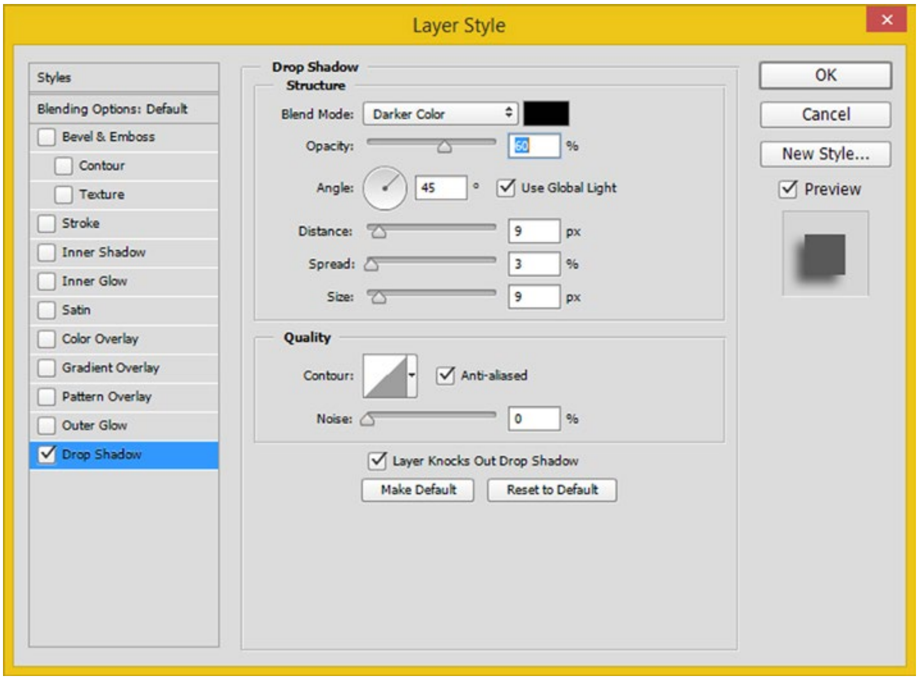
The result can be seen in Figure 12-9, along with the selected layer, and the Mode and Opacity settings, which are encircled in red, at the top right of the screenshot.



Color
Opacity: 50%
Fill: 100%

Figure 12-9. *You can set the Color mode and Opacity to 50% to get special effects blending*

The next upgrade you did in GIMP was to make your Ring drop shadow look more realistic. The way to do this in Photoshop is to edit your **Drop Shadow Layer Style**, whose dialog is shown in Figure 12-10. Since we used the Darken Only mode in GIMP, we will select the equivalent **Darker Color** blend mode, using the drop-down menu, and match the **60%** Opacity setting that we used in GIMP as well. This will provide a more realistic shadow.



Drop Shadow
Structure
Blend Mode: Darker Color
Opacity: 60 %
Angle: 45 ° [✓] Use Global Light
Distance: 9 px
Spread: 3 %
Size: 9 px
Quality
Contour: [✓] Anti-aliased
Noise: 0 %
[✓] Layer Knocks Out Drop Shadow

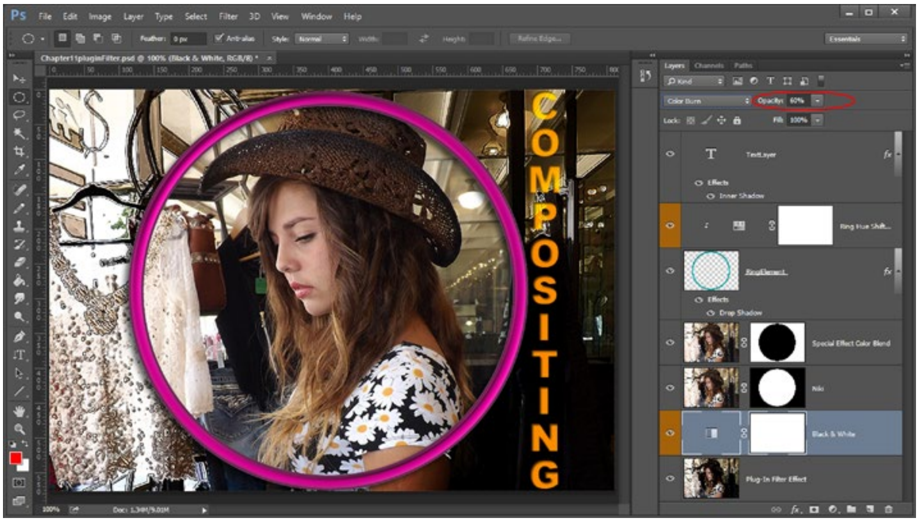
Figure 12-10. *To provide a more realistic shadow in Photoshop, change the Drop Shadow to use the Darker Color mode*

The final move that we made in GIMP was to enhance the color and contrast of your backplate's special effect digital painting algorithm (filter) application by applying the **Burn** blend mode in the Niki Black & White layer above the Niki.png color layer.

In Photoshop, we can do this by setting a **Color Burn** mode on the Black & White adjustment layer! The adjustment layer can apply its digital image adjustment algorithm and then run that result through the blending mode algorithm as well. This allows you to "stack" multiple compositing pipeline processing special effects into one single image compositing special effect layer.

After all, that is what this chapter is focusing on: how to maximize the compositing capabilities of the digital imaging software package in order to get the maximum result by using the minimum amount of layers.

In Figure 12-11, I have set the Opacity at **60%** to tone down the Color Burn mode algorithm application, exactly like I did in GIMP. As you can see, you have achieved a similarly professional result even though you are using a different plug-in filter to create your special effect.



Color Burn Opacity: 60%
Fill: 100%

__Figure 12-11.__ Results of setting the Color Burn mode and 60% Opacity on the Black & White layer

Let's take a look at the compositing pipeline process from the bottom of your layer stack up, to see what is actually going on here under the hood. In Photoshop, your **Plug-In Filter Effect** is your back plate, again with the Normal (no) blending mode and 100% Opacity. This is significantly different from how this composite was set up using GIMP, which is interesting to note.

Your **Black & White** adjustment layer then creates a black-and-white pixel image using the Black & White algorithm and passes that data over to the **Color Burn** blend mode algorithm.

All this intra-layer pixel pipeline processing should be applied to each pixel of your Plug-In Filter Effect layer, as prescribed by the all-white adjustment layer mask, which delineates a full opacity processing on all pixels in the digital image composite.

The **Niki** layer essentially uses a layer mask along with a Normal (no blending) mode and 100% Opacity to overlay imagery in a circular masked region of the original (unprocessed) image of the model inside (and under the edges of) the RingElement.

The **Special Effect Color Blend** layer uses this same Niki image with the opposite (inverted) layer mask to overlay color pixels on the outside area (and under the edges) of the Ring Element.

Setting a 50% Transparency (blend), and processing the pixels from the layers underneath this layer, on the outside of the Ring Element, through the **Color** mode algorithm, allowed you to add some color to that Black & White special effect filter you applied.

A **RingElement** layer primarily leverages an alpha channel and a Drop Shadow Effects (fx) algorithm sublayer to overlay a cyan-colored metal ring over the bottom four layers using the Normal (no blending) mode and 100% Opacity (full pixel strength).

Your **Ring Hue Shift 140** adjustment layer applies the hue-shifting algorithm to invoke a positive 140-degree hue shift, as needed to reproduce the same wine color that you used in GIMP.

The **Textlayer** also leverages an alpha channel to overlay text on the bottom six layers using a **Hard Light** blend mode and a 100% Opacity, or pixel strength, along with an Inner Shadow FX (Effects) layer, which creates your cut-out text special effect.

Now that you've put together an impressive digital image compositing pipeline on steroids, it's now time to switch gears and to take a look at saving files, data footprint optimization and how to use open source programming languages in the digital image compositing work process.

Be sure and play around with all of the things which you have learned about during the first dozen chapters of this book so that you can continue to master these concepts and features!

Summary

In this twelfth chapter, you learned more about, and looked specifically at, **digital image compositing pipeline processing** in both Photoshop and GIMP. You fine-tuned your digital image compositing project, adding subtle color effects and detail to the special effects filter you applied in Chapter 11, and made your RingElement have better color composition with the rest of the image composite.

In GIMP you color shifted your cyan metal ring and added layers that will give you some new mode special effects, and that you can blend with your special effects filter results.

In order to achieve a similar digital image compositing processing pipeline result in Photoshop CS6, you again had to change around the layer, layer mask, and adjustment layers; add an adjustment layer; and update some of the Layer Style Effects (fx) settings.

In the next chapter, you will learn about how to save out your files, and about **data footprint optimization** concepts and techniques.

CHAPTER 13

The Data Footprint of Digital Images: Optimization

Now that we have gone over how to create professional digital images using the powerful features in digital image compositing software packages, which you could probably base a lucrative career on, it's time to take a look at how to output, or "export," these digital image composites, as highly optimized files. In this chapter, you will learn about the considerations of **data footprint optimization**, so that you can export (save) a digital image file in the proper format for what you're doing. The objective in this chapter is to use the smallest possible number of bytes possible to create the file that contains your digital image compositing pipeline creation; that is, your resulting visual image multimedia asset for use in whatever digital playground you happen to be playing in at the time.

This is important if you are planning to use the digital images you create with the compositing concepts and techniques in this book in your new media programming projects, using popular and widespread platforms like Java, JavaFX, Android, and HTML5.

Since I am a programmer, I am going to devote the next two chapters in this book to how to optimize (Chapter 13) and interface (Chapter 14) digital imaging assets, with digital deliverables (as I call them), such as web sites and apps.

Photoshop Optimization: Save for Web

To start the optimization, open the **Chapter12compositing.psd** file, as seen in Figure 13-1, and follow the **File ➤ Save for Web** menu sequence.

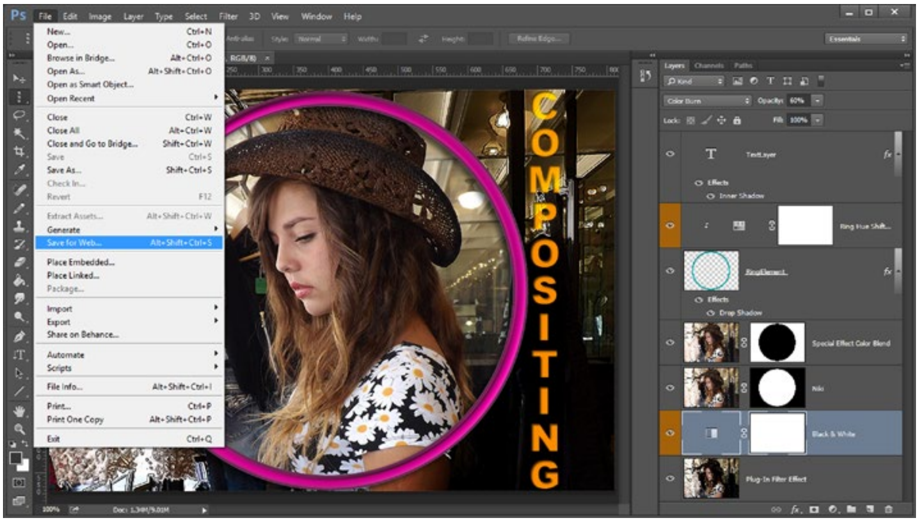
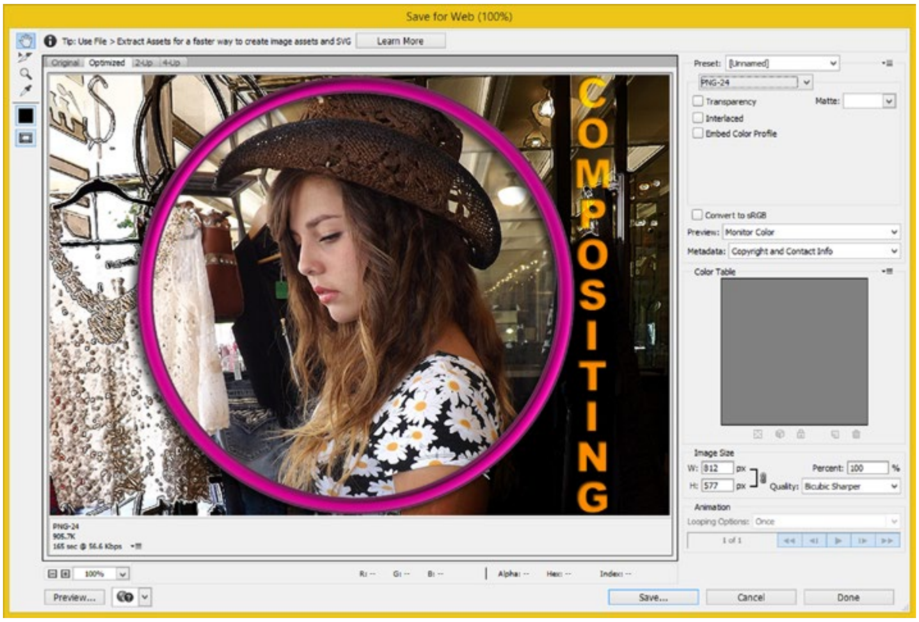


Figure 13-1. *To start the optimization process in Photoshop, follow the File ➤ Save for Web menu sequence*

This will bring up the **Save for Web** dialog, shown in Figure 13-2, which, in the case of this utility, is more like a command console. It has a **preview area** that reflects what the settings are doing at the left, along with **tool icons** for panning, zooming, and color sampling; a data footprint **information printout** at the bottom left side of the console; **settings** at the top right; and **scaling** (resampling) tools at the bottom right.

Photoshop has really done a great job, especially when you compare the data footprint optimization work process in GIMP, of providing an image data optimization control panel, allowing you to select settings, see the results, tweak these results, and even do things you would normally have to exit the dialog to accomplish, such as resampling (scaling) your image.

The highest quality result, which has the largest data footprint, is obtained using a **PNG-24** format, which also can be written as **PNG24**, seen in Figure 13-2.



PNG-24
905.7K
165 sec @ 56.6 Kbps

100%

Preset: [Unnamed]
PNG-24
Transparency
Matte:
Interlaced
Embed Color Profile
Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
Image Size
W: 812 px
H: 577 px
Percent: 100 %
Quality: Bicubic Sharper
Animation
Looping Options: Once

Figure 13-2. *To get the highest-quality result, select PNG-24, using the file format drop-down selector*

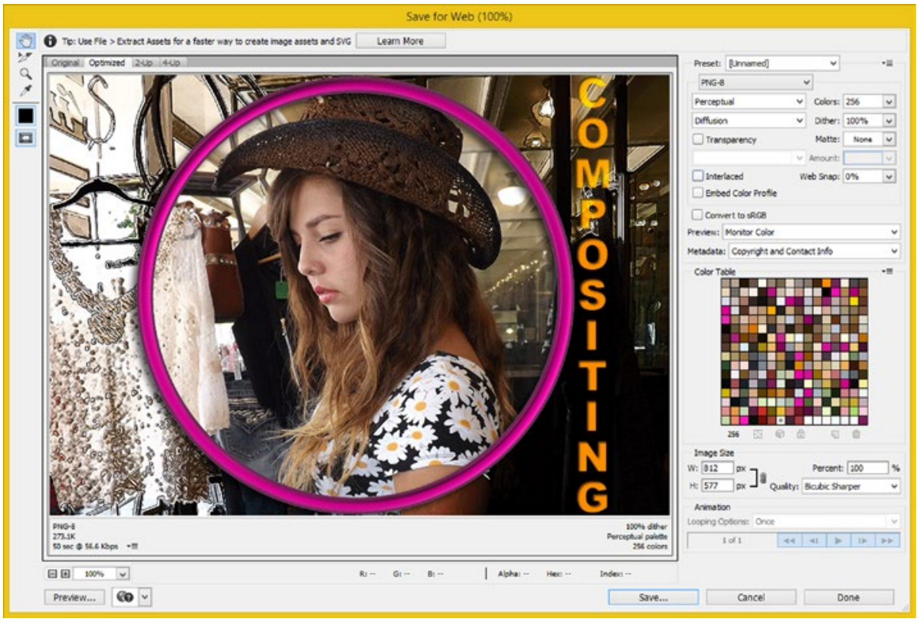
This format also supports an alpha channel; hence the **Transparency** option seen on the top right side of the dialog. If this is selected, the file becomes a **PNG-32,** also written as PNG32, because you have added an 8-bit transparency data plane.

You would use the Transparency option if you were exporting an object and its mask, such as the `RingElement.png` you used in this book. The **Interlaced** option may add data footprint overhead, as it causes additional bytes to be added to the file size. This allows the image to "stream" on slow connections, so that it writes to the screen as the image data arrives over the Internet. Obviously, this option is closely associated with web page design, development and user experience.

As you'll see when you drop down the file format options list, there is also a PNG-8 option for using the **indexed color** format, which we covered in Chapter 5. Let's look at that next.

Indexed Color Format: Using Less Color to Reduce File Size

With the **PNG-8** indexed color algorithm option, seen selected in the drop-down in the top-right corner of Figure 13-3, you can set a color selection algorithm, a number of colors, the dithering type and amount, and an **Interlaced** option.



Preset: [Unnamed]
PNG-8
Perceptual
Colors: 256
Diffusion
Dither: 100%
Transparency
Matte: None
Amount:
Interlaced
Web Snap: 0%
Embed Color Profile
Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
256
W: 812 px
H: 577 px
Percent: 100 %
Quality: Bicubic Sharper
Looping Options: Once
1 of 1

PNG-8
273.1K
50 sec @ 56.6 Kbps

100% dither
Perceptual palette
256 colors

100%
R: -- G: -- B: -- Alpha: -- Hex: -- Index: --

Figure 13-3. *Selecting PNG-8 from the file format drop-down selector will allow you to choose from a number of additional indexed color optimization options*

If you select Transparency, you will need a **Matte** color, since only a 1-bit (black and white only, no gray values) alpha channel is supported, which has no anti-aliasing (gray) values.

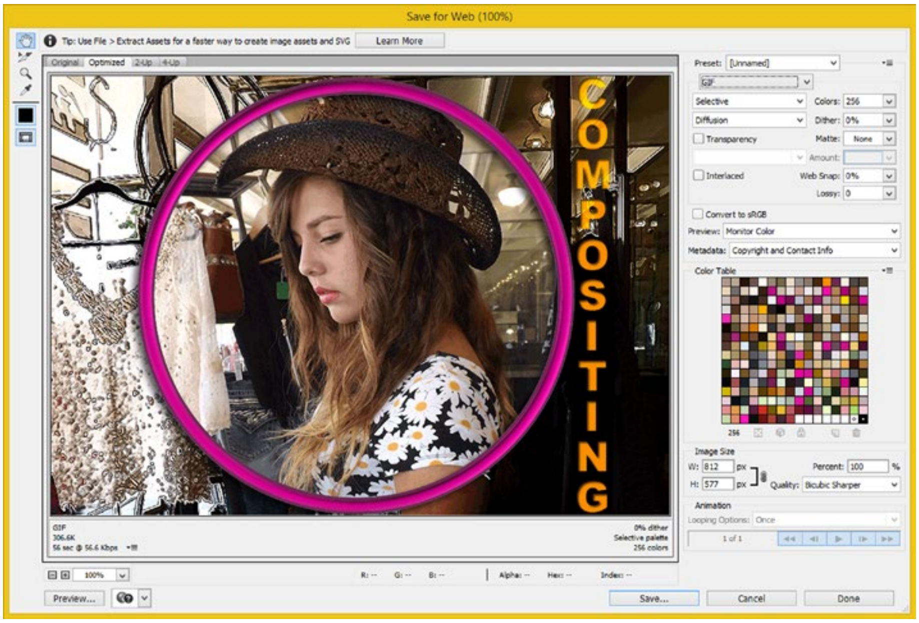
Choosing PNG-32 will give you a far-better compositing (edge quality) result than choosing PNG-8, since it uses 8-bit, or 256 grayscale, levels for its alpha channel data representations. So always use PNG-32 if you can, as it will represent any alpha-based objects with perfect quality.

The PNG-8 option gives you a **332 percent**-greater data footprint reduction than PNG-24—from 906K to 273K. As you set, or tweak, these different options, this dialog (optimization engine) will calculate a new data footprint, shown at the lower-left portion of your display, along with how long it should take to transfer over the Internet, for users using old fashioned 56KB modem connections.

As you can see in the cheek areas of Niki's face, a quality trade-off does exist when using indexed color formats.

If the pixels are small enough, and you use diffusion dithering, this will not be noticeable. Small pixels are found in printers, high resolution images, and modern "super fine dot pitch" consumer electronics devices.

The PNG-8 compression algorithm, or codec, is more recent than the Compuserve **GIF** indexed color codec, and thus gives you a better data footprint result. I took a screenshot (Figure 13-4) using the same settings to show you the GIF compression result of **307KB**. This has a **12 percent-larger** data footprint, so for this image, PNG-8 is 12 percent more efficient than GIF, just from an algorithmic standpoint. The GIF settings and the data footprint result can be seen in Figure 13-4. The only reason I use GIF is for its animation capability. This is referred to as the **aGIF**, **AnimGIF**, or **Animated GIF** indexed color file format.



Preset: [Unnamed]
GIF
Selective
Colors: 256
Diffusion
Dither: 0%
Transparency
Matte: None
Amount:
Interlaced
Web Snap: 0%
Lossy: 0
Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
Image Size
W: 812 px
H: 577 px
Percent: 100 %
Quality: Bicubic Sharper
Animation
Looping Options: Once
1 of 1

GIF
306.6K
56 sec @ 56.6 Kbps

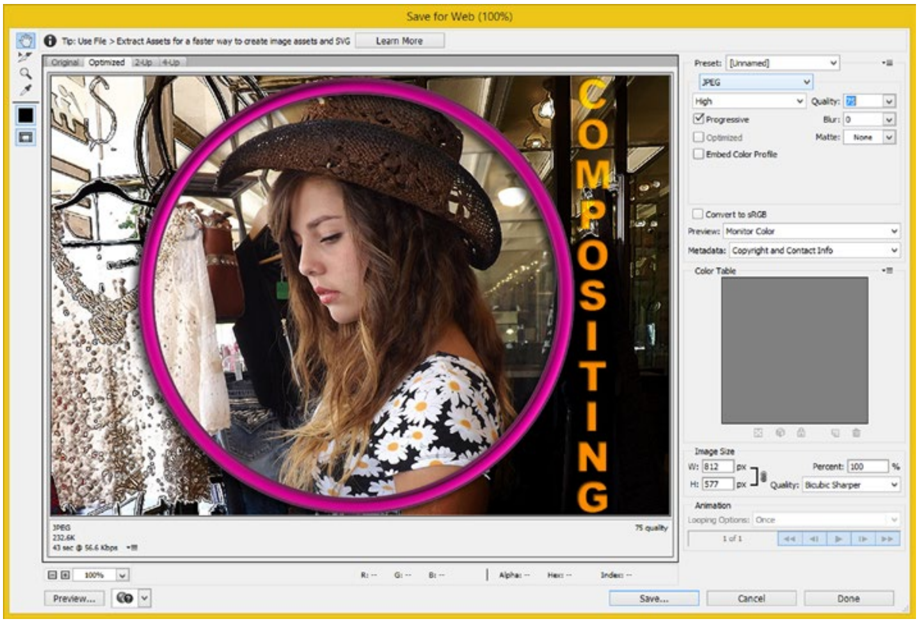
0% dither
Selective palette
256 colors

Figure 13-4. *To compare the compression capabilities of GIF, select GIF from the file format drop-down selector in the upper right corner*

So if you're using animation, use GIF, if you are using static (still) indexed color digital imagery, use PNG-8. Next, let's take a look at the lossy JPEG codec, so we can see how it changes the original image data, to reduce the data footprint.

Lossy Compression: Using JPEG to Reduce File Size

With the **JPEG** true-color algorithm option chosen from the drop-down menu in Figure 13-5, you can set a quality level and numeric value (amount), which will tell the JPEG algorithm how much of the original image pixel quality can be changed. Below that, there is a **Progressive** option that you can check off; it is similar to the indexed color Interlaced option, but using it results in a smaller file size, rather than a larger file size!



Preset: [Unnamed]
JPEG
High
Quality: 75
☑ Progressive
Blur: 0
☐ Optimized
Matte: None
☐ Embed Color Profile
☐ Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info

Image Size
W: 812 px
H: 577 px
Percent: 100 %
Quality: Bicubic Sharper

Animation
Looping Options: Once
1 of 1

JPEG
232.6K
43 sec @ 56.6 Kbps
75 quality

100%
R: -- G: -- B: -- | Alpha: -- Hex: -- Index: --

Figure 13-5. *To set a quality level and numeric value (amount), select JPEG, using the file format drop-down selector*

You can also apply a **Blur** factor to the image, using the Blur checkbox in the upper right hand corner. Blur will reduce data footprint when used with a JPEG algorithm, because edges, especially sharp (high adjacent pixel contrast) edges, are hard (difficult) for the JPEG algorithm to optimize efficiently. Use a small value for the Blur drop-down, if you use any at all.

The proper way to use a JPEG algorithm is to tweak the **Quality** percentage while at the same time observing visual preview quality and the resulting data footprint at the bottom left side of this dialog. You will have to decide what your trade-off between data footprint reduction and image quality will be relative to your intended usage of an image. In this case, I utilized a high **75% Quality** setting with **Progressive** downloading, and I got a **233KB** data footprint, which is **389 percent** smaller than the PNG-24 data footprint. JPEG Quality values between 60% and 100% will generally give you an excellent visual result; values under 20% generally will not.

Layer Visibility: Selecting Which Pixels Will Be Saved

Before we look at GIMP data footprint optimization, which will follow the same principles, as it uses the same image codecs, I wanted to show you a principle that holds across both Photoshop and GIMP. I am only going to show you this principle for Photoshop, as the same exact principle will work using GIMP.

As you may have surmised on your own, the **Save for Web** (Photoshop) and **File Export** (GIMP) engines will only save data from **visible layers** into your image files. So if you wanted to save a **PNG-32** with the image of Niki inside of the ring to use elsewhere, you would turn off Textlayer, Special Effect Color Blend, Black & White, Plug-In Filter Effect, and Effects (the RingElement sub-layer), as can be seen in Figure 13-6.

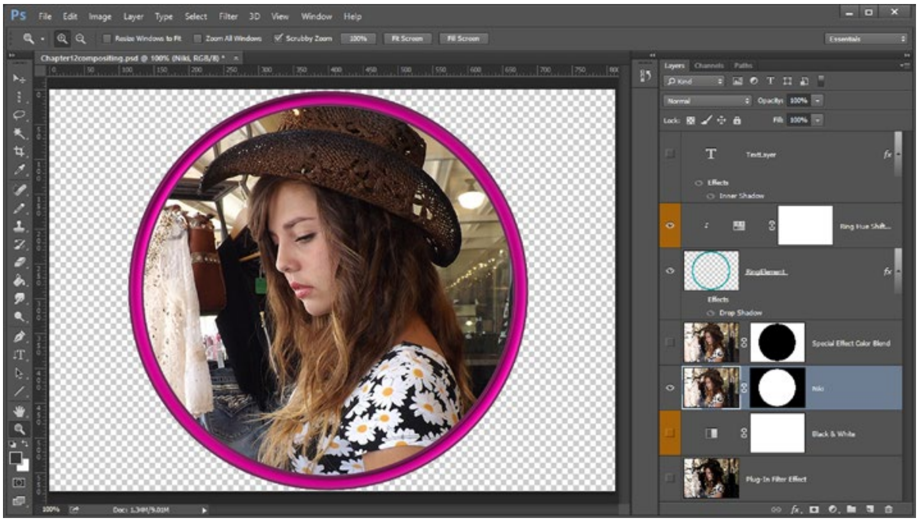
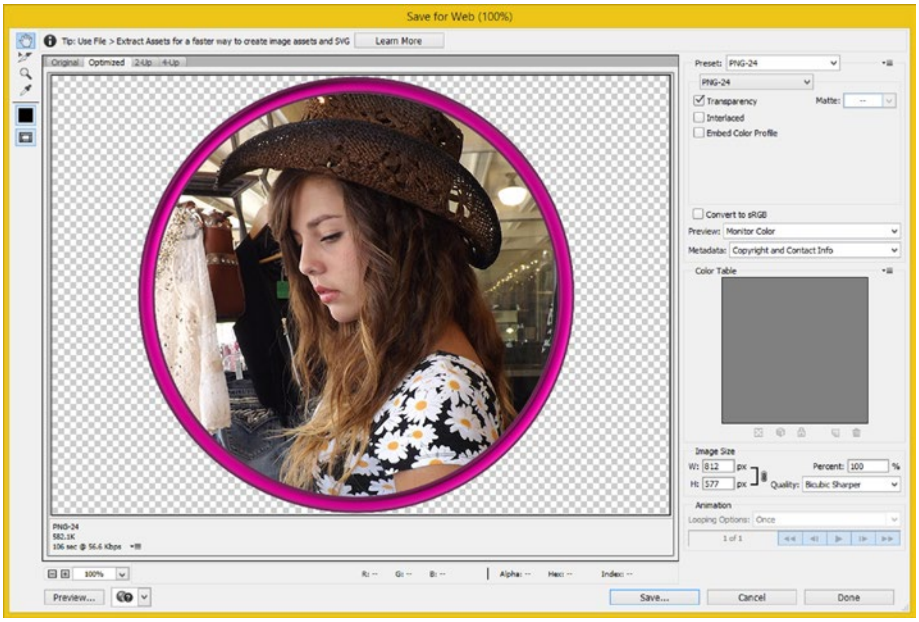


Figure 13-6. *You can turn off layers whose data you don't want to appear in the final image file export operation*

This will create a wine-colored metal ring to encircle Niki, perfectly masked on top of a transparency, as can be seen in both Figure 13-6 and in the Photoshop Save for Web preview, shown in Figure 13-7, as I'm exporting in a PNG-32 file format.



Preset: PNG-24
PNG-24
☑ Transparency Matte: --
☐ Interlaced
☐ Embed Color Profile
☐ Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
W: 812 px
H: 577 px
Percent: 100 %
Quality: Bicubic Sharper
Looping Options: Once
1 of 1

PNG-24
582.1K
106 sec @ 56.6 Kbps

100%
R: -- G: -- B: -- Alpha: -- Hex: -- Index: --

Figure 13-7. *Select PNG-24 and Transparency to create a PNG-32*

If you check off the **Transparency** option, as seen in Figure 13-7, the alpha channel will appear as a checkerboard pattern, just like in Photoshop. If you don't select this option, the exported image will use a white background color.

At this point, you are probably wondering to yourself, What if I'd left the Drop Shadow Layer style selected before I chose Save for Web? We are going to take a look at that next, in order to see if Photoshop will put transparency data from the FX settings dialog into an alpha channel for us. I think you will be pleased with the results!

To take a look, select your **Drop Shadow Effects** visibility eye icons, and turn the RingElement drop shadow effect back on, as seen underneath your gray (selected) RingElement layer, on the right-hand side of the screen in Figure 13-8.

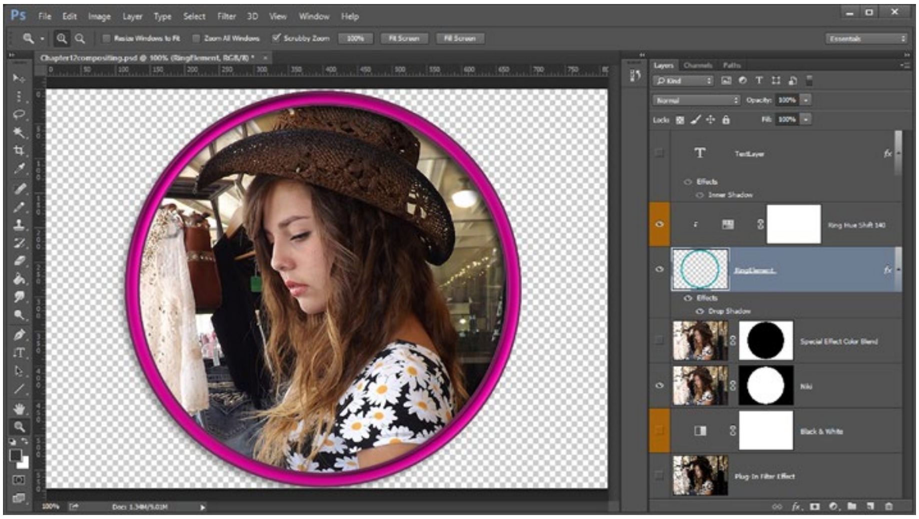
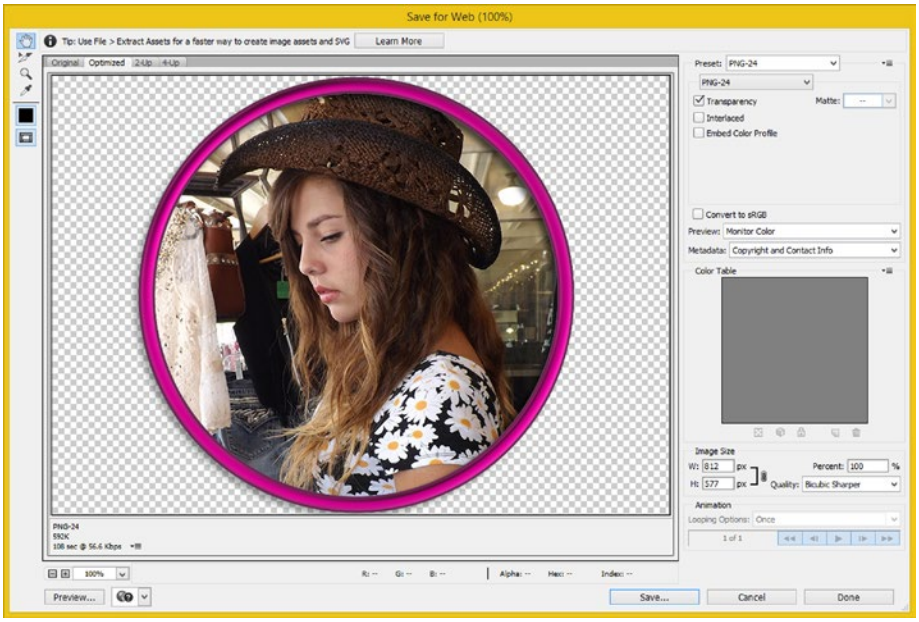

File Edit Image Layer Type Select Filter 3D View Window Help
Chapter12compositing.psd @ 100% (RingElement, RGB/8) *
Layers Channels Paths
TextLayer
Effects
Inner Shadow
Ring Hue Shift 140
RingElement
Effects
Drop Shadow
Special Effect Color Blend
Niki
Black & White
Plug-In Filter Effect

Figure 13-8. *In order to add a drop shadow to the PNG32, select your Drop Shadow Effects visibility icons*

As you can see in the Save for Web dialog in Figure 13-9, the drop shadow has the subtle transparency you were hoping it would, so this transparent object will indeed carry its own shadow with it.


Save for Web (100%)
Tip: Use File > Extract Assets for a faster way to create image assets and SVG
Learn More
Original Optimized 2-Up 4-Up
Color Table
Image Size
Animation
Preview...
Save...
Cancel
Done

Preset: PNG-24
PNG-24
☑ Transparency Matte: --
☐ Interlaced
☐ Embed Color Profile
☐ Convert to sRGB
Preview: Monitor Color
Metadata: Copyright and Contact Info
W: 812 px Percent: 100 %
H: 577 px Quality: Bicubic Sharper
Looping Options: Once
1 of 1
PNG-24
592K
108 sec @ 56.6 Kbps
100%

Figure 13-9. *To create a PNG-32, select PNG-24 and Transparency*

If you're going to create the additional special effect, be sure to add a dozen or so pixels of height to the bottom of the image size, so that the bottom of your drop shadow effect does not get cut off.

As you have already learned, this should be accomplished by using the **Canvas Size** function, not the Image Size function, which scales the image, and which would just make the cut-off shadow artifact (mistake) larger and more noticeable.

Using Canvas Size makes the canvas larger, so that the bottom of the drop shadow special effect has the room (pixels) it needs to render the complete drop shadow including the fade out portion which makes it look more realistic.

Now we're ready to take a look at GIMP's File ➤ Export As work process, which is quite different from Photoshop's.

GIMP Optimization: File ➤ Export As

Saving files, or "**exporting**" files, as it is called in GIMP, is in no way as intuitive as it is in Photoshop. However, if you know how the GIMP file export system works, you can do just about anything that you can do in Photoshop, by using the GIMP File ➤ Export As menu sequence.

Let's see how file exporting works in GIMP, by opening the **Chapter12compositing.xcf** file, which can be seen in Figure 13-10. Then, follow the **File ➤ Export As** menu sequence to bring up an **Export Image** dialog, which is quite similar to the File Save dialog of any standard operating systems, except that it has a button labeled **Export**, instead of Save.

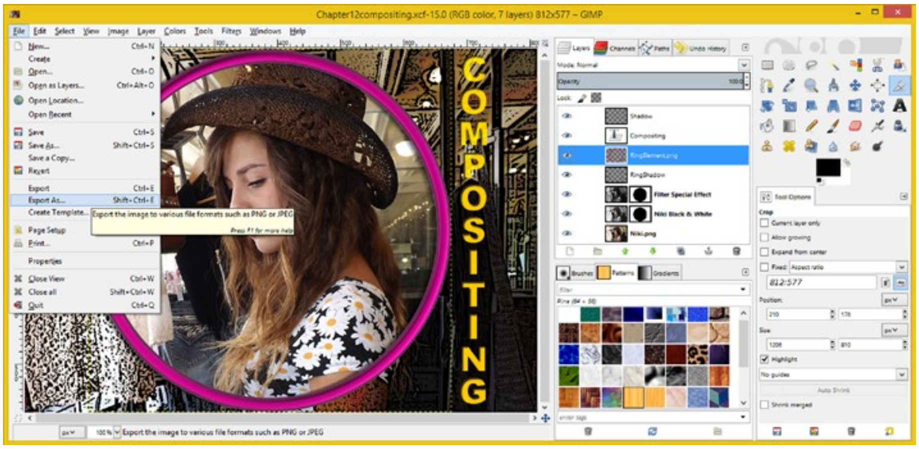
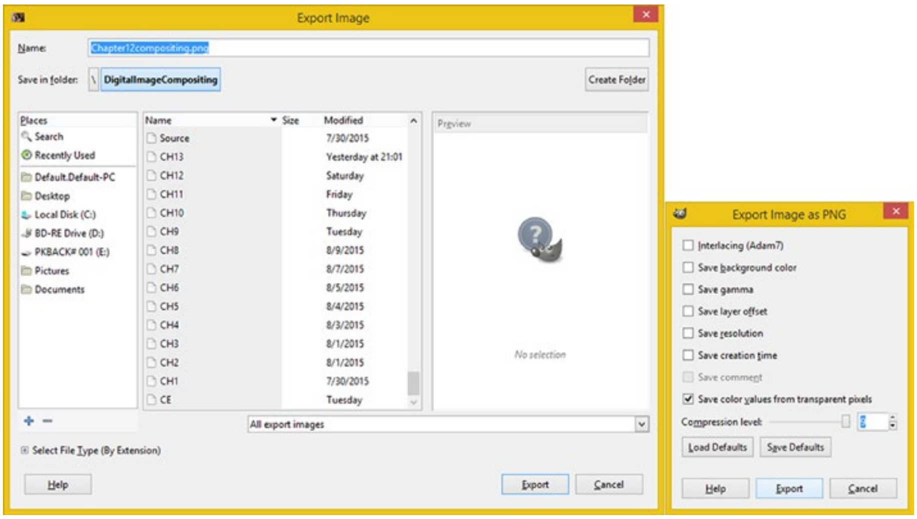


Figure 13-10. *To bring up the* **Export Image** *dialog in GIMP, use the File ➤ Export As menu sequence*

If you use **File ➤ Save** or **File ➤ Save As** in GIMP you will be saving the GIMP image compositing project itself, which uses the **.XCF** file extension, which is native to GIMP and not supported on many (if any) other platforms, even open source ones such as HTML5, Java, JavaFX, or Android Studio.

The **Export Image** dialog will select an image file format codec based on the File Name extension. Let's call the PNG-24 format file **Chapter12compositing.png**, as is highlighted in Figure 13-11.



Name: Chapter12compositing.png

Save in folder: \ DigitalImageCompositing

Name	Size	Modified
Source		7/30/2015
CH13		Yesterday at 21:01
CH12		Saturday
CH11		Friday
CH10		Thursday
CH9		Tuesday
CH8		8/9/2015
CH7		8/7/2015
CH6		8/5/2015
CH5		8/4/2015
CH4		8/3/2015
CH3		8/1/2015
CH2		8/1/2015
CH1		7/30/2015
CE		Tuesday

- ☐ Interlacing (Adam7)
- ☐ Save background color
- ☐ Save gamma
- ☐ Save layer offset
- ☐ Save resolution
- ☐ Save creation time
- ☐ Save comment
- ☑ Save color values from transparent pixels

Compression level: 9

Figure 13-11. *To export a file in GIMP, enter filename.png, select your desired options, and then hit the Export button*

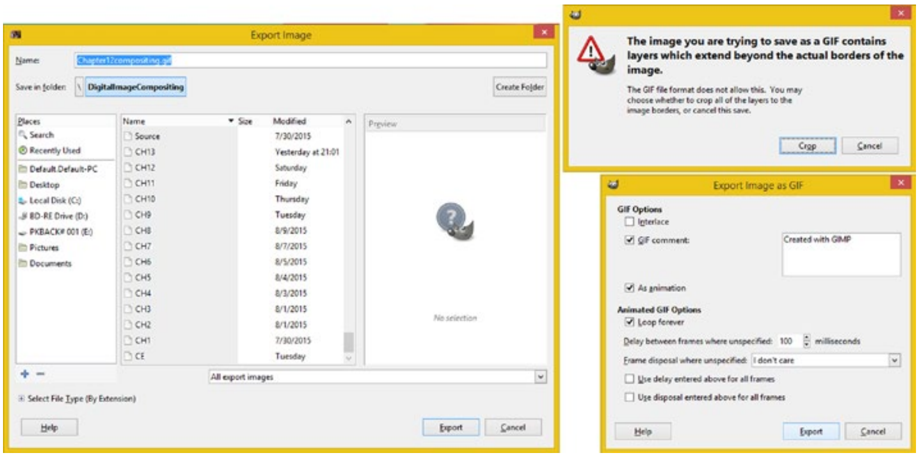
The way GIMP knows whether to use PNG-24 or a PNG-8 is that it looks at the **Color Mode** setting for your composite, which we covered in Chapter 5, and is found under the **Image** menu, and then under the **Mode** sub-menu.

Once you hit the **Export** button, it will open the File Format Options dialog—in this case **Export Image as PNG**—allowing you to select the options and **Compression Level** that you wish to use, as shown in Figure 13-11 on the bottom-right hand side.

As you can see in the figure, I selected the **Save Color Values from Transparent Pixels** and set the maximum compression level of 9, leaving the other options unselected to save on the data footprint for the file, as these are not necessary for the application I am using the digital image composite for.

Notice at the bottom right of Figure 13-11 that you can also save and load default settings, so if you have certain option combinations, you can save these using unique file names if you like. Then you don't have to be checking option checkboxes all of the time during this stage of your image data footprint optimization work process.

To save a file as a GIF, simply change the extension to `.gif` and click the Export button. If you're using Truecolor mode, GIMP will convert the file to indexed color and give you the **Export Image as GIF** settings dialog, shown in Figure 13-12. I got this error dialog as well, seen in the upper-right corner of the figure, so I selected the **Crop** option to trim the unused transparency.



Name: Chapter12compositing.gif

Save in folder: \ DigitalImageCompositing

Name	Size	Modified
Source		7/30/2015
CH13		Yesterday at 21:01
CH12		Saturday
CH11		Friday
CH10		Thursday
CH9		Tuesday
CH8		8/9/2015
CH7		8/7/2015
CH6		8/5/2015
CH5		8/4/2015
CH4		8/3/2015
CH3		8/1/2015
CH2		8/1/2015
CH1		7/30/2015
CE		Tuesday

All export images

The image you are trying to save as a GIF contains layers which extend beyond the actual borders of the image.

The GIF file format does not allow this. You may choose whether to crop all of the layers to the image borders, or cancel this save.

GIF Options
- [ ] Interlace
- [x] GIF comment: Created with GIMP
- [x] As animation

Animated GIF Options
- [x] Loop forever

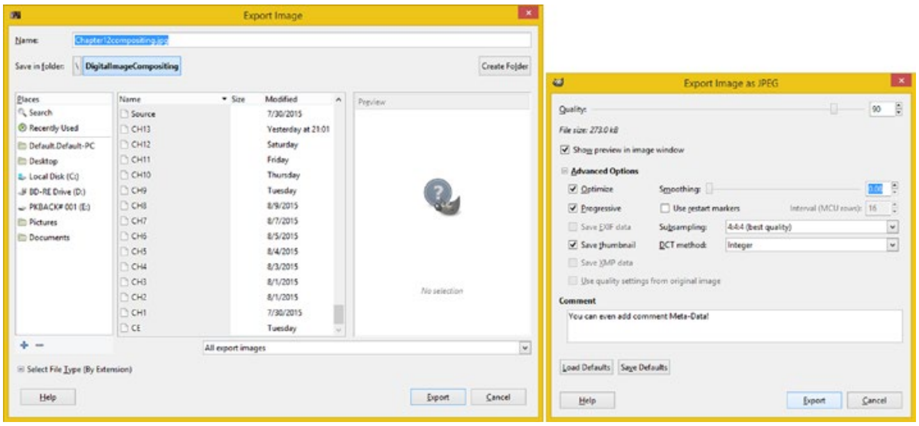
Delay between frames where unspecified: 100 milliseconds

Frame disposal where unspecified: I don't care

- [ ] Use delay entered above for all frames
- [ ] Use disposal entered above for all frames

Figure 13-12. *To save a file as a GIF, enter filename.gif, select your desired options, and then click Export*

If you want to save the file as a JPEG, change the extension to `.jpg`, click the **Export** button, and set the options in the **Export Image as JPEG** dialog, shown on the right-hand side of Figure 13-13. Check off the **Show Preview in Image Windows** box to see a preview of your settings just like you would (and need to) with Photoshop's Save for Web function.



Name: Chapter12compositing.jpg

Save in folder: \ DigitalImageCompositing

Name	Size	Modified
Source		7/30/2015
CH13		Yesterday at 21:01
CH12		Saturday
CH11		Friday
CH10		Thursday
CH9		Tuesday
CH8		8/9/2015
CH7		8/7/2015
CH6		8/5/2015
CH5		8/4/2015
CH4		8/3/2015
CH3		8/1/2015
CH2		8/1/2015
CH1		7/30/2015
CE		Tuesday

All export images

Quality: 90

File size: 273.0 kB

- [x] Show preview in image window

Advanced Options
- [x] Optimize — Smoothing: 0.00
- [x] Progressive — [ ] Use restart markers — Interval (MCU rows): 16
- [ ] Save EXIF data — Subsampling: 4:4:4 (best quality)
- [x] Save thumbnail — DCT method: Integer
- [ ] Save XMP data
- [ ] Use quality settings from original image

Comment

You can even add comment Meta-Data!

Figure 13-13. *To save your file as a JPEG, enter filename.jpg, select desired options, and click the Export button*

The **Optimize** and **Progressive** options can be used to make your JPEG file size smaller. You can save a thumbnail image for your desktop using the **Save Thumbnail**. You can also apply Blur with the **Smoothing** slider. I used a value of **0.08**, to apply a blur, which you cannot see but which will improve compression.

You can even specify advanced **Subsampling** configurations and you can also specify which **DCT** (Discrete Cosine Transform) **method** will be utilized by the JPEG Codec's primary algorithm.

Next, let's replicate what we did with Photoshop CS6 by saving out a PNG-32 image with the RingElement and Niki model photograph in the center on top of a transparent alpha channel.

Since your GIMP 2.8.14 digital image composite utilizes a different compositing structure than your Photoshop composite does, we'll need to use a different work process to accomplish this objective. Let's get into that next, and then we will be finished with our data footprint optimization for this chapter.

Creating a Masked PNG-32 Object in GIMP

In order to create a masked PNG-32 object, the first thing you need to do is to transfer and invert the mask used on the second and third layer (I tried to drag and drop it, but no such luck), and apply it to the first (bottom) layer.

The work process to do this is to select either your second or third layer, and then right-click on the layer and select the **Mask to Selection** option, as shown in Figure 13-14.

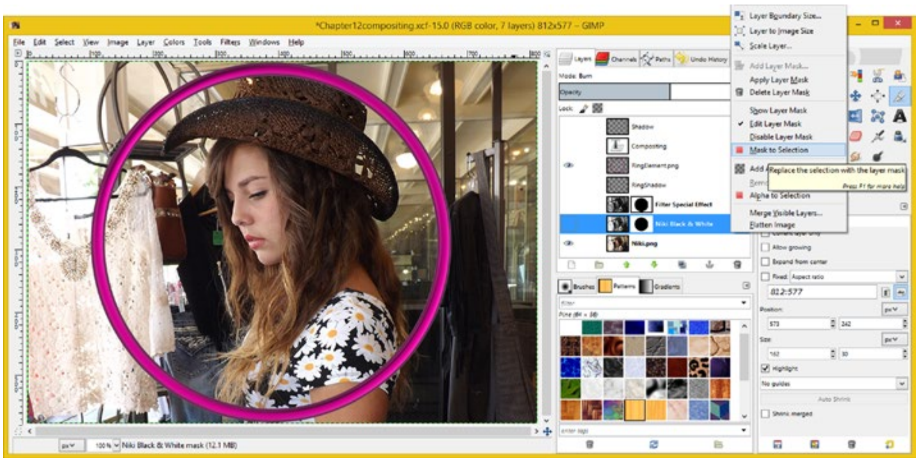


Figure 13-14. *To transfer a layer mask to an interim selection set, select the Layer Mask and use the Mask to Selection option*

We will have to use the following work process, to bridge a layer mask across from the Filter and Black & White layers, to the Niki.png color layer that does not yet have a layer mask: Mask to Selection, Invert Selection, Select a Target Layer, and Add Layer Mask (Selection to Mask).

Next, use a **Select ➤ Invert** menu sequence and invert the selection, as seen in Figure 13-15.

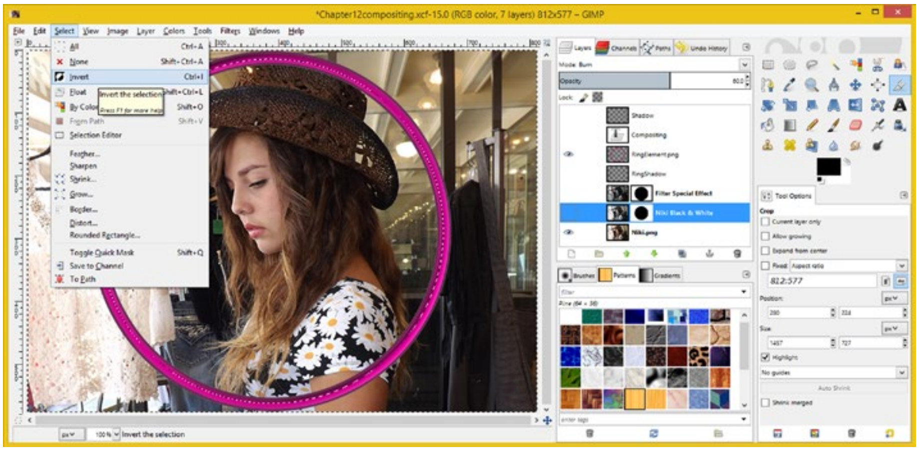
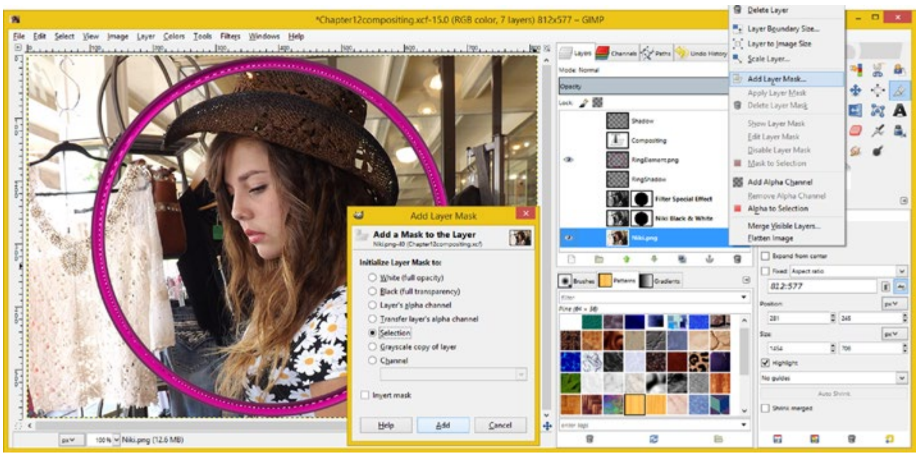


Figure 13-15. *To invert a mask (as a selection) use Select ➤ Invert*

The source layer that you have selected, either Filter Special Effect or Niki Black & White, makes no difference, as they both have the same layer mask that you can turn into the selection prior to the Invert Selection operation. However, you must select the Niki.png layer, before you use Add Layer Mask.

It is important to note that your selections are held in system memory, and are not attached to any particular layer—not until you use the Add Layer Mask to attach them, that is!

Now you are ready to add your layer mask. To do this, select your **Niki.png** layer and then right-click and select the **Add Layer Mask** option. In the dialog that comes up, check off the **Selection** option and click the **Add** button, shown in Figure 13-16. This will turn the inverted selection into a layer mask that looks like an exact opposite of the other two layer masks, which are black circular areas on a white background.



Initialize Layer Mask to:
White (full opacity)
Black (full transparency)
Layer's alpha channel
Transfer layer's alpha channel
[✓ Selection]
Grayscale copy of layer
Channel
Invert mask

Figure 13-16. *To add your layer mask, select the Niki.png layer and right-click on the selected layer and select Add Layer Mask*

As you can see in Figure 13-17, this gives you a result you were looking for—the color photographic model image inside the wine metal ring element on a transparency background—which is being created with alpha channel data. If you export with a .png extension, you'll get a PNG-32 image, just as you desired.

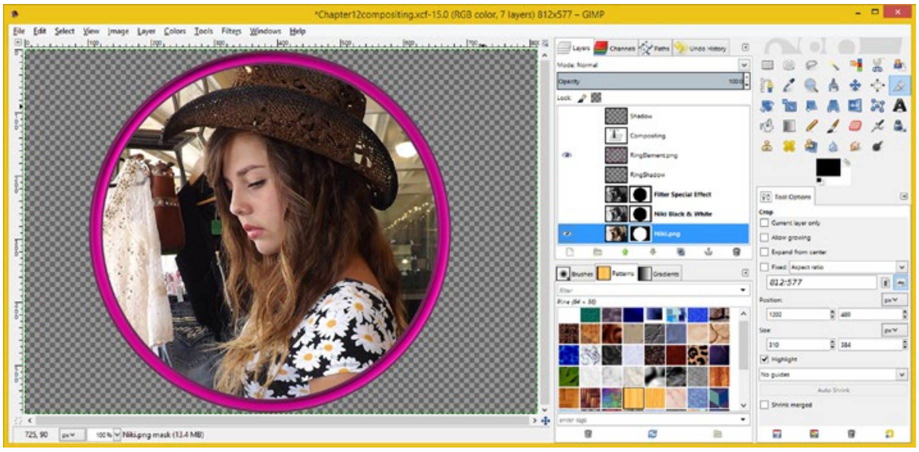

*Chapter12compositing.xcf-15.0 (RGB color, 7 layers) 812x577 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Layers Channels Paths Undo History
Shadow
Compositing
RingElement.png
RingShadow
Filter Special Effect
Niki Black & White
Niki.png
Niki.png mask (13.4 MB)

Figure 13-17. *After turning your inverted selection into a layer mask, the Niki.png layer mask is now the exact opposite of layers two and three as shown selected in the Layers palette*

We'll continue to look at exporting files during Chapter 14, when we go over how to "bridge" digital image compositing with various open source programming languages and platforms.

Summary

In this thirteenth chapter, we reviewed data footprint optimization during the File Save for Web and File Export As work processes for Photoshop and GIMP. We looked at how to do these work processes for the three most widely used digital image file formats—PNG, GIF, and JPEG. You first learned how to optimize for these files in Photoshop, then you saw how layer visibility in an image composite can affect what is saved in the file format, and finally you found out about the same exact work processes that you can utilize with GIMP 2.8.14.

In the final chapter, you will learn more about some of the important considerations involved with bridging your work in your digital image compositing software packages, looking at some of the most popular coding environments, such as Python, Java, JavaFX, HTML5, CSS3, JavaScript, and Android Studio.

CHAPTER 14

The Automation of Digital Imaging: Programming

Now that we have looked at how to create professional digital images using the powerful features in both of the digital image compositing software packages and to export them to the most popular file formats, it's time to take a look at those programming platforms themselves, just in case you want to take your digital image compositing career to the next level. In this chapter, you will learn about the internal programming languages used for GIMP (Python) and Photoshop (JavaScript) as well as external programming languages that support digital imaging such as C# (.NET); ObjectiveC (iOS); Java (Android Studio, Linux); JavaFX; JavaScript; HTML5; and CSS3 (WebKit Browsers, HTML5 OSs).

This is important information to know if you plan to use the digital imagery from this book's compositing concepts and work processes in your own programming projects, **open-software development platforms**, or, if you have any interest in learning more about adding programming to your vast repertoire.

The platforms that we will cover in this chapter run a majority of the consumer electronics industry's hardware devices, and include **Java** (Android Studio and WebKit); **JavaFX** (Android, iOS, Windows, Linux, Mac OS X, Solaris); and **JavaScript** with CSS3 and HTML5 scripting (WebKit browsers).

This chapter is not going to teach you programming—that would take an number of books (and coding experience) to do—but it will expose you to what's possible if you extend the journey you are on from digital image compositing to new media software development. Everything that we will be covering in the chapter is free for commercial use, and you can easily download Android Studio (IntelliJ), Java and JavaFX (NetBeans), HTML5 (NetBeans) and Python (Eclipse).

We will start with the internal scripting languages for Photoshop and GIMP, and then cover the popular open platforms.

Internal Scripting Languages: Automation

Both Photoshop and GIMP include several scripting languages for automating work processes within their software packages. This automation is traditionally referred to in the computer industry as "batch processing"—a term which comes from the old mainframe days where data was input during the day by employees and then batch processed at night, by the computers, while the employees got some sleep. I will cover the various scripting languages used by Photoshop and GIMP in these first two sections of the chapter, and then look at other languages that support digital image compositing pipelines over the rest of the chapter. This will show you what is available to bridge your digital compositing pipeline from digital imaging software to programming.

Photoshop Scripting: JavaScript and ExtendScript

A script is a series of programming statements that instructs a digital image compositing application to perform a series of tasks, or what I refer to as “moves” or a work process. Adobe applications like Photoshop support basic scripting languages, including JavaScript and Adobe’s own extended JavaScript, which is called **ExtendScript**. JavaScript is a scripting language that was originally developed as ECMAScript and was used to make Web pages interactive. In fact, it is still being used for this purpose, and it’s powerful enough to write casual games.

If you use Mac OSX, you'll also have the choice of using **AppleScript**, and if you use Windows, you can also use **VBScript**, known as **VBA**, or **Visual Basic**. A version of Visual Basic, VBScript is the programming language developed by Microsoft and does not include all of the program’s advanced features. VBScript can talk to host applications using ActiveX Scripting. You can find a number of professional VBScript editors on the Internet. If you use Microsoft Office, you can find the built-in Visual Basic editors by following the **Tools ➤ Macro ➤ Visual Basic Editor** menu sequence. I recommend using JavaScript, since it’s open source, and is therefore it is not proprietary, like AppleScript and VBScript are.

You should get into the habit of appending your JavaScript scripts using a `.jsx` extension, rather than the standard `.js` extension, in case you upgrade to using the Adobe ExtendScript (JSX) Photoshop scripting language.

JavaScript also has another important advantage over VBScript and AppleScript, if you like to share your scripting work, in that JavaScript can be used in either Microsoft Windows OS or in Apple Macintosh OSX.

In Photoshop, you can access only **.jsx** files from within the application. Another advantage to using JavaScript is that you must run AppleScript or Visual Basic script from outside of the application. There’s a lot to be said for learning to code with open-platform languages, as they are the ones exploding in use.

To write script in JavaScript, you can use a text editor or you can use the **ESTK ExtendScript tool kit** provided with the Adobe application. ESTK has many features, which makes it easier to use than text editors. These include a built-in syntax checker that identifies where the problems are in your script and the ability to run the scripts right from the ESTK without saving them out to a file. This feature can save you an incredible amount of development time since you have to test your scripts a significant number of times to get them to work.

To run a JavaScript, whether it was written by you or someone else, you use the **File ➤ Scripts ➤ Browse** menu sequence, as seen in Figure 14-1.

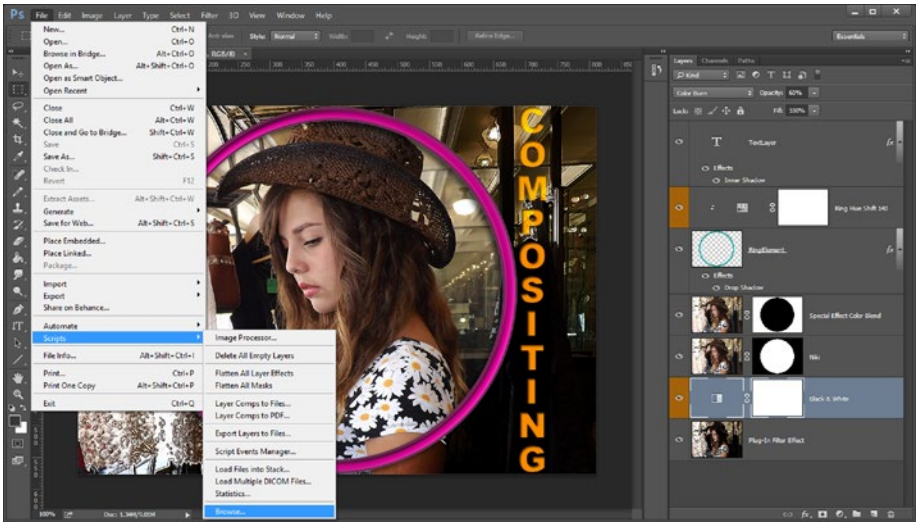


Figure 14-1. *To load JavaScript, go to File ➤ Scripts ➤ Browse*

As you have probably surmised, scripting in GIMP is very different than it is in Photoshop and uses different languages.

GIMP Scripting: Scheme, Script-Fu, and Python

GIMP has two different levels of scripting complexity. There is a LISP programming language variant, called **Scheme**, built into GIMP that runs a **Script-Fu** (hey, your guess is as good as mine!) scripting language. There is also **PyGIMP**, a custom version of **Python** supporting GIMP features, which can access low-level API (application programming interface) calls. PyGIMP is more powerful than Scheme. So if you're writing complex plug-ins, use PyGIMP; if you're automating a work process, use Script-Fu.

You'll find a plethora of free Script-Fu scripts available online that you can download and install inside of your copy of GIMP by using a fairly simple work process. Once you download your scripts, you should copy or move them to your GIMP **scripts** directory. If you don't know the path to the scripts directory for your OS, you can find it using the GIMP **File ➤ Preferences** menu sequence to open a **Preferences** dialog, as seen in Figure 14-2.

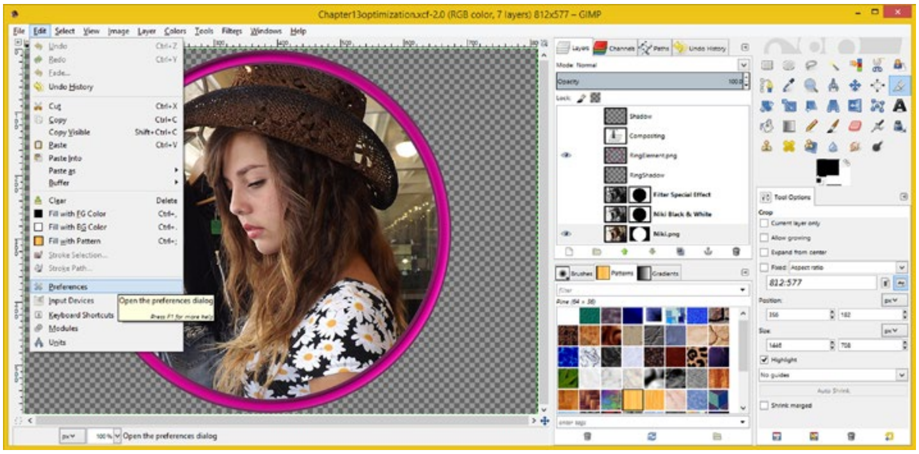
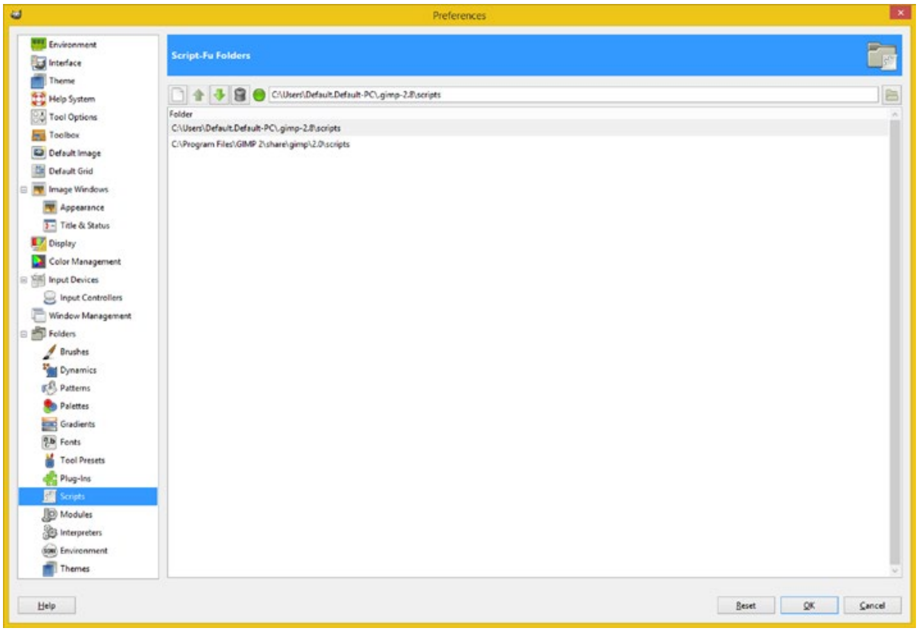


Figure 14-2. To find the path to the scripts directory for your OS, start with the File ➤ Preferences menu sequence

Once you have the **Preferences** dialog open, look in the **Folders ➤ Scripts** area for the scripts path, as shown in Figure 14-3. Mine was in: **C:/Users/PC-Name-Here/.gimp-2.8/scripts**.



C:\Users\Default.Default-PC\.gimp-2.8\scripts

C:\Users\Default.Default-PC\.gimp-2.8\scripts
C:\Program Files\GIMP 2\share\gimp\2.0\scripts

Figure 14-3. After you open the Preferences dialog, look in the Folders ➤ Scripts area for the path to your scripts directory

To install these scripts that you have copied into your scripts folder, you could both exit and restart GIMP, or you could simply refresh GIMP's menu structure.

This can be done using the **Filters ➤ Script-Fu ➤ Refresh Scripts** menu sequence, as is shown in Figure 14-4. After you do this, your new scripts will appear in one of your menus. If you don't find them, look for them under the **Filters** menu. If they don't appear anywhere, something went wrong with the Script-Fu and it produced syntax (programming or scripting) errors.

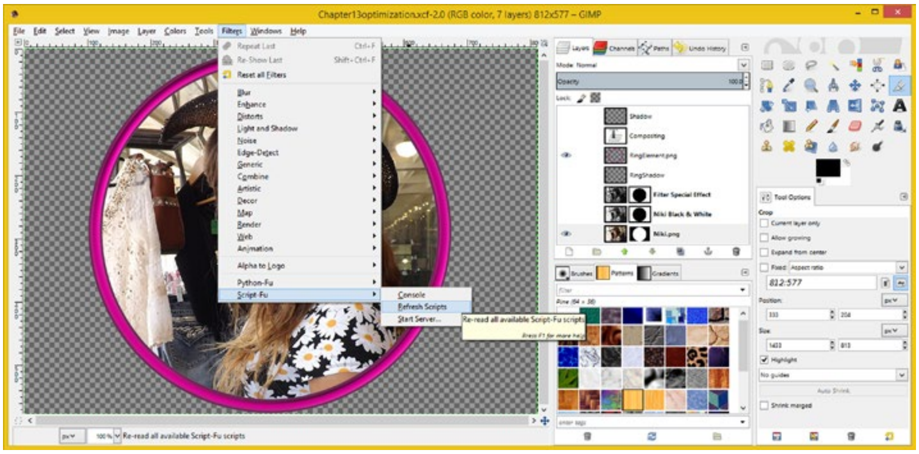
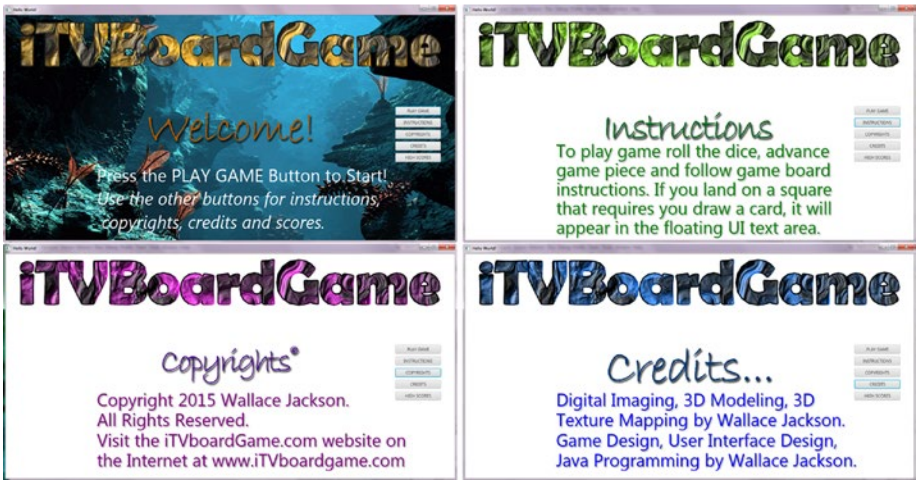


Figure 14-4. You can access GIMP scripts by following the Filters ➤ Script-Fu menu sequence

As you can see, if you need to automate the work process in GIMP or Photoshop, the programming tools are there to do so. In the next section, let's take a look at digital image compositing in some of the most popular platforms: Java and JavaFX, HTML5 and CSS3, and Android Studio for Android OS.

Java and JavaFX: javafx.scene.effect API

Digital image compositing pipelines can be built and controlled using code in the Java programming language, as can be seen in Figure 14-5. The back plate imagery is a PNG24, the 3D logo is a PNG32, and the script text images are PNG32 as well. Java has a library called **JavaFX** that provides support for expansive new media assets, spanning digital imaging, digital audio, digital video, and i3D real-time OpenGL rendering. Most of the digital imaging modes, filters, and effects we've been using in this book are located in the **javafx.scene.effects** library. Java is used in Android Studio, HTML5 Browsers, Windows, Macintosh OSX, Linux Distributions, and Open Solaris OS. JavaFX apps can run on Android and iOS; thus Java 7 and 8, the world's most popular programming language, is truly "Code once, deliver everywhere."



Welcome!
Press the PLAY GAME Button to Start!
Use the other buttons for instructions, copyrights, credits and scores.

Instructions
To play game roll the dice, advance game piece and follow game board instructions. If you land on a square that requires you draw a card, it will appear in the floating UI text area.

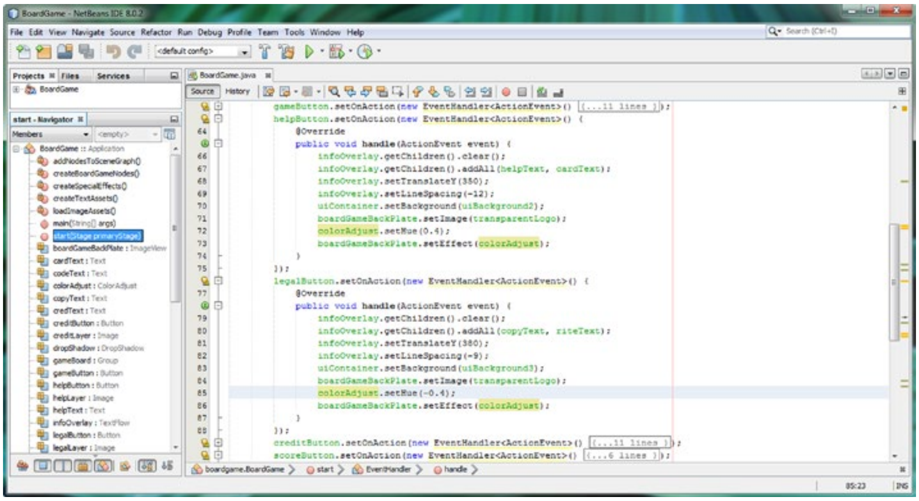
Copyrights
Copyright 2015 Wallace Jackson.
All Rights Reserved.
Visit the iTVboardGame.com website on the Internet at www.iTVboardgame.com

Credits...
Digital Imaging, 3D Modeling, 3D Texture Mapping by Wallace Jackson.
Game Design, User Interface Design, Java Programming by Wallace Jackson.

Figure 14-5. *Image compositing pipelines using Java and JavaFX*

The splash screen for a game I am coding for my upcoming "Pro Java Games Development" title, coming out from Apress in 2016, can be seen in Figure 14-5. The upper-left quadrant is the splash screen itself and uses a PNG24 back plate image overlaid with PNG23 composite images (two), text elements, and user interface button elements.

The Java and JavaFX code can be seen in Figure 14-6. The JavaFX API is part of the Java API; thus, JavaFX is Java. These were separate programming languages until JavaFX was acquired by Sun Microsystems, right before it was acquired by Oracle.



```
gameButton.setOnAction(new EventHandler<ActionEvent>() {...11 lines });
helpButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        infoOverlay.getChildren().clear();
        infoOverlay.getChildren().addAll(helpText, cardText);
        infoOverlay.setTranslateY(350);
        infoOverlay.setLineSpacing(-12);
        uiContainer.setBackground(uiBackground2);
        boardGameBackPlate.setImage(transparentLogo);
        colorAdjust.setHue(0.4);
        boardGameBackPlate.setEffect(colorAdjust);
    }
});
legalButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        infoOverlay.getChildren().clear();
        infoOverlay.getChildren().addAll(copyText, riteText);
        infoOverlay.setTranslateY(380);
        infoOverlay.setLineSpacing(-9);
        uiContainer.setBackground(uiBackground3);
        boardGameBackPlate.setImage(transparentLogo);
        colorAdjust.setHue(-0.4);
        boardGameBackPlate.setEffect(colorAdjust);
    }
});
creditButton.setOnAction(new EventHandler<ActionEvent>() {...11 lines });
scoreButton.setOnAction(new EventHandler<ActionEvent>() {...6 lines });
```

```
BoardGame :: Application
    addNodesToSceneGraph()
    createBoardGameNodes()
    createSpecialEffects()
    createTextAssets()
    loadImageAssets()
    main(String[] args)
    start(Stage primaryStage)
    boardGameBackPlate : ImageView
    cardText : Text
    codeText : Text
    colorAdjust : ColorAdjust
    copyText : Text
    credText : Text
    creditButton : Button
    creditLayer : Image
    dropShadow : DropShadow
    gameBoard : Group
    gameButton : Button
    helpButton : Button
    helpLayer : Image
    helpText : Text
    infoOverlay : TextFlow
    legalButton : Button
    legalLayer : Image
```

Figure 14-6. *Java and JavaFX code creates splashscreen variants*

I show the code for two of the buttons—the **Instructions** and the **Copyrights** buttons, called **"helpButton"** and **"legalButton"** in the Java code—in Figure 14-6. I do not seek to teach you coding during this chapter; however Java code is understandable enough for explaining what is going on for this particular compositing pipeline, which is shown in Figure 14-5.

In Figure 14-6, the `boardGameBackPlate.setImage(transparentLogo);` Java statement is changing my digital image asset in my back plate layer, using terms that you are now familiar with from this book.

The `colorAdjust.setHue(0.4);` Java statement color shifts the logo 40 percent (you previously color shifted it 140 percent, going from a cyan to a wine color) around the color wheel. The `colorAdjust.setHue(-0.4);` Java code color shifts the logo 40 percent around the color wheel in the opposite direction. With a color slider visualization, this would be 40 percent to your right (positive) and 40 percent to your left (negative), respectively.

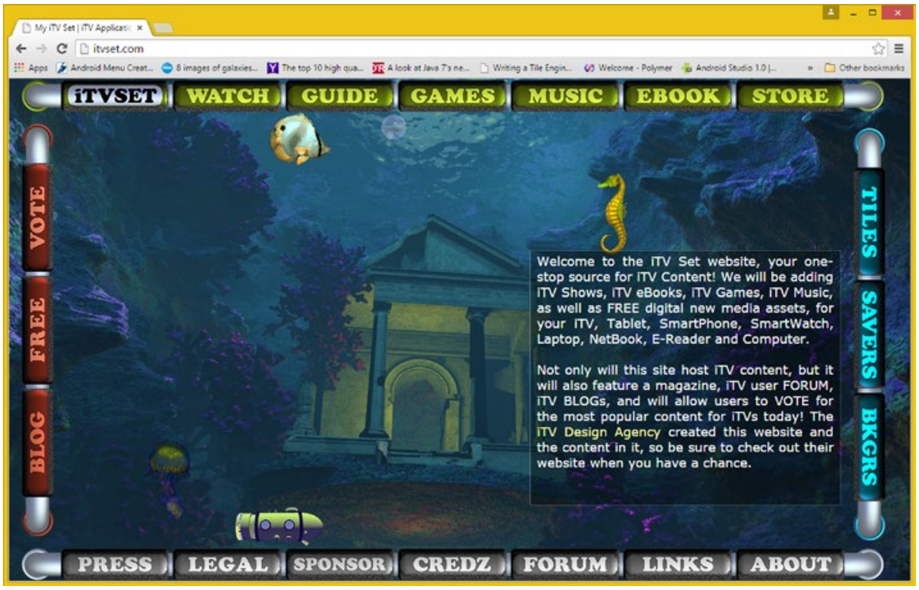
Vertical, *y*-axis pixel positioning is being accomplished using the `infoOverlay.setTranslateY(350);` and the Java code and text spacing are being accomplished using an `infoOverlay.setLineSpacing(-12);` Java statement. This is how an image composite is created with code!

Next, let's take a look at image compositing pipelines, implemented using only basic markup languages (HTML5 and CSS3).

HTML5 and CSS3: Markup Compositing

Whereas Java (JavaFX) is the most popular programming language, the HTML5 and CSS3 markup languages are the most widespread as far as usage is concerned. This is because the latter two are featured in every browser, using an API called **WebKit**, which is similarly used by Android and iOS and so HTML5 and CSS3 are therefore also supported in these OSs. There are also several HTML5 OSs out now, not surprisingly, from the makers of the HTML5 browsers, including Chrome OS, Firefox OS, and Opera OS. They are used in smart phones, tablets, and iTV sets. For instance, Panasonic iTV sets use Firefox OS; Sony Bravia iTV sets use the Opera OS; Google has a range of Chrome hardware products; and Alcatel uses the Mozilla Firefox OS for its smart phones internationally.

I composited the HDTV resolution `www.iTVset.com` web site, using only lossless codecs and using a combination of PNG24 and animated GIF for backplate sections, and PNG32 for all i3D UI element overlays. As you can see, in Figure 14-7, this entire site looks like a truecolor digital image, but is animated and interactive. The site was coded using only 24 lines of HTML5, and features a less-than-2-MB total graphic image asset data footprint overhead.



Welcome to the iTV Set website, your one-stop source for iTV Content! We will be adding iTV Shows, iTV eBooks, iTV Games, iTV Music, as well as FREE digital new media assets, for your iTV, Tablet, SmartPhone, SmartWatch, Laptop, NetBook, E-Reader and Computer.

Not only will this site host iTV content, but it will also feature a magazine, iTV user FORUM, iTV BLOGs, and will allow users to VOTE for the most popular content for iTVs today! The iTV Design Agency created this website and the content in it, so be sure to check out their website when you have a chance.

Figure 14-7. *A website I composited using HTML5 and CSS3 markup*

This small data footprint was made possible by using this digital image compositing pipeline, because it allowed the granularization of digital image assets into smaller file sizes for faster data transfer speeds over the Internet.

A good analogy for this data granularization is sand through an hourglass, the finer the sand, the smoother and the faster it will flow through any bottlenecked scenario.

A lot of **indexed color** assets were used, allowing for a site that looks like it is truecolor, but in fact, is not. For instance, all six animated 3D elements on this page can use an indexed color **animGIF** file format with the **1-bit premultiplied alpha channel**. I premultiplied the anti-aliasing color using an average background color, to hide anti-aliasing in the average surrounding background color value. For the Butterfly fish this was a bluish water color value, for instance.

The i3D user interface button elements use the **PNG32** format, where the alpha channel composites the UI seamlessly over any background image, in any section of the web site, including digital video, Java (JavaFX) applications, 3D animation as seen on the home page, and full-screen imagery.

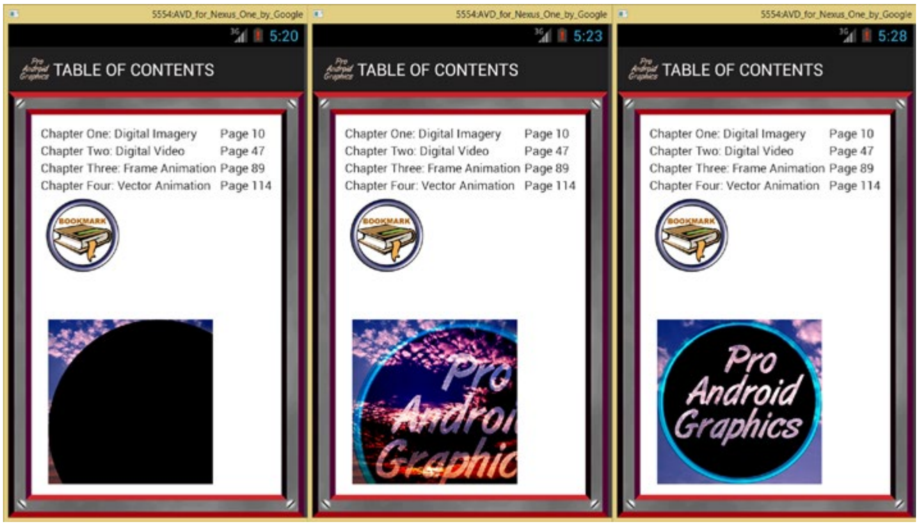
Graphics elements are held in HTML5 <**DIV**> tags, and CSS3 is used for **blending**, **opacity**, **positioning,** and **interactivity**. I do not obstruct the right-click action with my code, in any way, so you can right-click on the site to "view as source" to look at any of this code, at any time during this site's development. The text is rendered by WebKit in its own <DIV> tag regions, using HTML5 to define the content and metatags, and using CSS3 to define transparency, font, color, size, and the margins and padding (text <DIV> container spacing) values.

As you can see, just like Java or JavaFX, HTML5 and CSS3 can provide you with an image compositing pipeline that can be almost as powerful as the custom image compositing pipelines that you can create using GIMP and Photoshop. However, you will have to be a creative and a savvy programmer in order to "pull" these capabilities out of these popular, open source, computer programming languages. It's worth the effort, I promise!

Android Studio: The PorterDuff Class

The Google Android OS platform is running more smart phones, e-book e-readers, tablets, iTV sets, game consoles, smart watches, and IoT devices than any other OS platform in the history of the world. In fact, I have written a series of **Pro Android** titles for Apress, from 2013 through 2016, including *Pro Android Graphics* (2013), *Pro Android UI* (2014), and *Pro Android Wearables* (2015), and I am currently writing *Pro Android IoT* (2016). I cover how to code for PorterDuff pixel blending and pixel transfer modes in *Pro Android Graphics*.

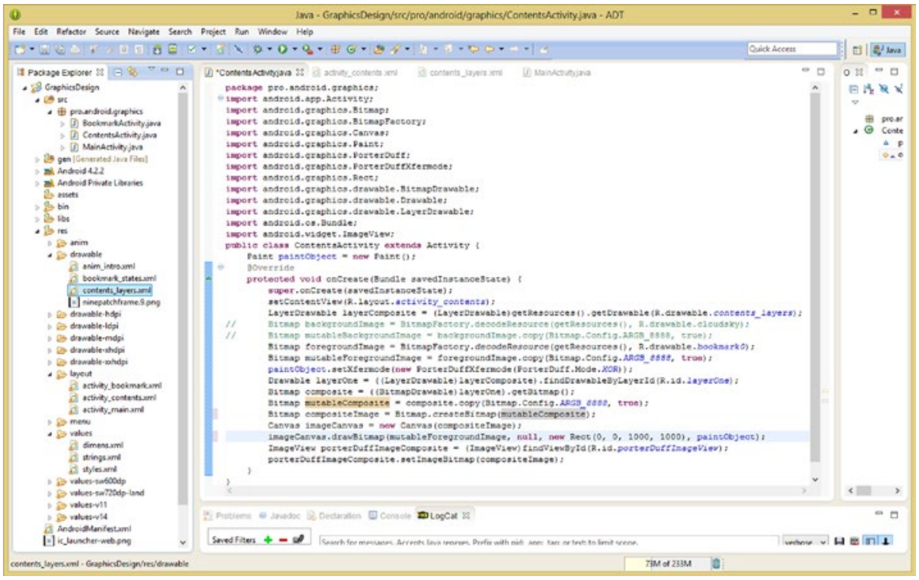
Figure 14-8 shows one of the screenshots from that book, showing three different blending modes in use in the Nexus One Emulator in the Eclipse IDE. As of Android 5, the IntelliJ IDE is the primary IDE that Android Studio developers will now use for their Android OS application development.


5554:AVD_for_Nexus_One_by_Google
5:20
TABLE OF CONTENTS
5554:AVD_for_Nexus_One_by_Google
5:23
TABLE OF CONTENTS
5554:AVD_for_Nexus_One_by_Google
5:28
TABLE OF CONTENTS
BOOKMARK
Pro Android Graphics

Chapter One: Digital Imagery	Page 10
Chapter Two: Digital Video	Page 47
Chapter Three: Frame Animation	Page 89
Chapter Four: Vector Animation	Page 114

Figure 14-8. *Screenshot from Pro Android Graphics using a PorterDuff mode example*

The Java code to put together this compositing pipeline is quite complex, including a PNG24 back plate, a PNG32 ring element, a PNG32 3D logo, and a mask-controlled (alpha channel) black fill color, as can be seen in Figure 14-9.



```
package pro.android.graphics;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;
import android.graphics.Rect;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.LayerDrawable;
import android.os.Bundle;
import android.widget.ImageView;
public class ContentsActivity extends Activity {
    Paint paintObject = new Paint();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_contents);
        LayerDrawable layerComposite = (LayerDrawable)getResources().getDrawable(R.drawable.contents_layers);
//      Bitmap backgroundImage = BitmapFactory.decodeResource(getResources(), R.drawable.cloudsky);
//      Bitmap mutableBackgroundImage = backgroundImage.copy(Bitmap.Config.ARGB_8888, true);
        Bitmap foregroundImage = BitmapFactory.decodeResource(getResources(), R.drawable.bookmark0);
        Bitmap mutableForegroundImage = foregroundImage.copy(Bitmap.Config.ARGB_8888, true);
        paintObject.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.XOR));
        Drawable layerOne = ((LayerDrawable)layerComposite).findDrawableByLayerId(R.id.layerOne);
        Bitmap composite = ((BitmapDrawable)layerOne).getBitmap();
        Bitmap mutableComposite = composite.copy(Bitmap.Config.ARGB_8888, true);
        Bitmap compositeImage = Bitmap.createBitmap(mutableComposite);
        Canvas imageCanvas = new Canvas(compositeImage);
        imageCanvas.drawBitmap(mutableForegroundImage, null, new Rect(0, 0, 1000, 1000), paintObject);
        ImageView porterDuffImageComposite = (ImageView)findViewById(R.id.porterDuffImageView);
        porterDuffImageComposite.setImageBitmap(compositeImage);
    }
}
```

Figure 14-9. *The Java code implementing a compositing pipeline*

I will now go through what the statements in Figure 14-9 do so that you can see a compositing code pipeline that matches up with the layer-based compositing pipeline that you have become familiar with over the course of this book.

Layers are called **LayerDrawable** in Android, so the first line of code loads the LayerDrawable with a **contents_layers.png** asset using a `getResources().getDrawable()` method call "dot chain."

I commented out a backgroundImage plate, for testing, so I will just cover the foregroundImage plate code here. To make the image layer, I create a Bitmap object named **foregroundImage** and then I load it with an image asset named **cloudsky.png** using a **BitmapFactory.decodeResource()** method call.

I then make that Bitmap object **mutable** (changeable) by putting it into memory using a **.copy()** method call specifying the **32-bit ARGB** color depth (this also can be called a "color space"). **ARGB_8888** means each of the channels uses 8-bit data values.

I then set a PorterDuff transfer mode (sometimes called a "blending mode," although technically, some blending modes will transfer pixels, rather than blending them together) on a Paint object using the `.setXfermode()` method, using the **XOR** mode.

I next created my Drawable object, named **layerOne**, and I then load it with a Bitmap object named "**composite**," and load that into memory as a mutableComposite.

Using that object, I then create a Bitmap object named "**compositeImage**." I then created a Canvas object to draw on, named **"imageCanvas"** and then I loaded that Canvas object with the compositeImage Bitmap object.

Next I draw the Bitmap object on the Canvas object using the **imageCanvas.drawBitmap()** method call that specifies an area using a square 1,000-pixel **Rect** object, with the blending mode using the **paintObject** and a **mutableForegroundImage** as a Bitmap object. This is the bitmap or raster image that I specified as having a color depth or color space of 32-bit, or ARGB_8888.

I create an **ImageView** named "**porterDuffImageComposite,**" to hold (display) this digital image compositing pipeline in my user interface design, and I then loaded this ImageView, by using the **.setImageBitmap()** method call.

As you can see, there are many of the elements that I've covered over the course of this book in this Java code, and you can bet that all of these concepts are supported in each of the major platform (Java, JavaFX, HTML5, CSS3, JavaScript, Android) areas that I've covered in this chapter.

A lot of the reason for this is because these platforms are all open source, and so are all of the Porter Duff modes, OpenGL, JPEG, GIF, and PNG formats, alpha channels, and even the GIMP digital image compositing software.

Since all of the areas of digital image compositing will be "Free for commercial use," it is logical for the open source platforms to completely incorporate them.

All of this is certainly excellent news for all of you digital image compositing aficionados, to be sure! I hope you have all enjoyed learning more regarding each of the different digital image compositing fundamentals over the course of these fourteen chapters.

Summary

In this fourteenth and final chapter, we got into some advanced topics that relate to computer programming and how different programming languages can support your digital image compositing endeavors, either inside digital image compositing software, as work process or special effects plug-in scripting tools, or outside your digital image compositing software. These languages can take your creation to the next level, by adding interactivity, as well as other useful features; the possibilities are limited only by your imagination!

First, we went over VBScript, AppleScript, JavaScript, and ExtendScript languages, for Photoshop. Then, we looked at the LISP-based Scheme, and Python-based PyGIMP languages used for scripting in GIMP, and how they are accessed inside of the digital image compositing software packages, in case you wanted to immediately utilize other scripts written by third parties. After that, we reviewed some of the popular open-source platform programming languages and how they can be utilized to create the same digital image compositing effects set-up, using Photoshop and GIMP.

We first went over Java and its JavaFX new media engine and saw how that platform supports digital image compositing as well as advanced blending modes and special effects algorithms. Then we took a look at HTML5 and CSS3, and saw that this platform could also implement the digital image compositing concepts and techniques you learned in this book using only markup languages for deliverables such as web sites and HTML5 OS applications.

Finally, we took a look at the Android 5.3 OS, which is currently using Java 7, but will run both HTML5 and JavaFX apps in the future. We looked at the PorterDuff class and some advanced code, showing that a compositing pipeline with blending modes can be coded in Android for advanced apps that are digital image compositing savvy.

I hope you have enjoyed this book, and the digital image compositing concepts, techniques, and fundamentals I've covered in it. Go forth, and blend into your surroundings!

Index

A

Alpha channels, 39
- blending, 40
- channels palette, 41
- checkerboard pattern, 41
- Expand Selection dialog, 44
- GIMP, 41, 45
- layer mask, 47
- layers palette, 41
- modal, 45
- Niki layer, 44
- Niki.png layer, 42
- Photoshop, 40, 43
- red mask, 41
- RingElement layer, 41, 43
- Select ➤ Invert menu, 47
- shift modifier, 53

Aspect ratio
- definition, 15
- digital image, 15
- image size, 17
- math, 16
- screen shapes, 16

B

Batch processing, 151
BitmapFactory.decodeResource() method, 160
Brush Strokes section, 120

C

CMYK (cyan, magenta, yellow, and black), 23–26
Color depth, 31
- high color, 34
- HRDI, 32
- indexed color, 32
  - color selection algorithm, 33
  - conversion dialog, 34
  - dithering selection algorithm, 33
  - Floyd-Steinberg diffuse dithering algorithm, 34
  - interlaced option, 33
  - restrictive algorithms, 33
  - web dialog, 32
- TGA, 32
- TIFF, 32
- true color
  - 24-bit imagery, 35
  - 32-bit imagery, 36

Color selection algorithm, 33
Colors ➤ Hue-Saturation menu, 124
Color theory, 23
- CMYK
  - color profile, 24
  - conversion process, 25
  - image compositors, 25
  - match resolution, 25
  - plug-ins, 26
  - profile dialog, 25
  - Separate+ plug-in, 26
  - subtractive color model, 24
- RGB
  - additive color model, 26
  - GIMP, 27
  - hexadecimal data, 28
  - levels of intensity, 28
  - modal palette, 28
  - Photoshop Preferences dialog, 27

Cool effects, Photoshop, 86
- Hard Light, 88
- Layer Style, 90
- Linear Burn, 88

Cool effects, Photoshop (*cont.*)
- Luminosity, 90
- Soft Light, 87

CSS3, 157

D

Data footprint optimization. *See* Photoshop optimization process

Digital Imaging
- digital images, 5
- GIMP, 1
- raster *vs.* vector, 6
- rendering, 6

Dithering selection algorithm, 33

E

ESTK ExtendScript tool kit, 152

ExtendScript, 152

F

Filter ➤ Filter Gallery menu, 119

Filters ➤ Artistic ➤ Cartoon menu, 114

Filter Special Effect layer, 127

Floyd-Steinberg diffuse dithering algorithm, 34

G

getResources().getDrawable() method, 160

GIMP
- Layer Mask, 102
- masked PNG-32 object creation, 147
- Ring Drop Shadow, 103

GIMP compositing, 123
- Burn blend mode, 131–132
- Burn layer blending mode, 126
- Burn mode, 126–127
- color mode and opacity, 130
- Compositing text layer, 127
- Darken Only mode, 125
- Darker Color mode, 131
- Drop Shadow Layer Style, 130
- Duplicate Layer dialog, 128
- Filter Special Effect layer, 127
- Image ➤ Adjustments ➤ Invert menu, 129
- intra-layer pixel pipeline, 132
- layer feature, 124
- Layer ➤ Duplicate Layer menu, 128
- Layer ➤ New Adjustment Layer ➤ Hue-Saturation menu, 130
- Lock: Brush and Lock: Alpha icons, 124
- Niki Black & White layer, 124, 126
- Niki.png, 127
- Opacity engine, 126
- RingElement layer, 124, 129, 132
- Textlayer, 133

GIMP optimization process, 144–147

GIMP plug-in filters
- adjustment layer mask icon, 116
- Brush Stokes section, 120
- cartoon effect, 115
- Duplicate Layer menu option, 114
- Filter ➤ Filter Gallery menu, 119–120
- Filters ➤ Artistic ➤ Cartoon menu, 114
- GIMP 2.8, 111–113
- Image ➤ Adjustments ➤ Invert menu, 119
- Layer ➤ Duplicate Layer menu, 116
- Niki layer mask, 117–118

GIMP scripting, 153–155

H

HDTV display screen, 8

High Dynamic Range Image (HRDI), 32

HTML5, 157

Hue-Saturation dialog, 124

I

imageCanvas.drawBitmap() method, 160

Image distortion, 15

Image ➤ Adjustments ➤ Invert menu, 119

Image size
- canvas size, 19
- crop tool, 18
- GIMP, 19
- Photoshop options, 21
- user interface design, 18

Internal scripting languages
- GIMP scripting, 153
- photoshop scripting, 152

J, K

Java
- image compositing pipelines, 156
- splashscreen variants, 156

JavaFX
    image compositing pipelines, 156
    splashscreen variants, 156
javafx.scene.effect API, 155
JavaScript, 152

L

Layers palette
    GIMP, 65
        color picker, 74
        drop shadowing, 77
        duplicate layer, 76
        Hue-saturation dialog, 77
        non-destructive editing, 75
        organization and masking, 67
        photoshop, 75
        preview option, 77
        TexLayer, 73
    Photoshop, 65
        adjustment layer, 71
        drop shadowing, 70
        Inner Shadow effect, 70
        organization and effects, 66
        vertical text tool, 69
LISP programming language
        variant, 153

M, N

Masked PNG-32 object creation, 147
Masking, 49
    anti-alias option, 55
    black-and-white mode, 58
    Channels palette tab, 53
    edge detection algorithm, 58
    Fill menu sequence, 54
    fine-tuning stage, 55
    high resolution image, 58
    invert menu sequence, 55
    Magic Wand, 50
        Contiguous selection, 51
        point sample, 50
        Tolerance settings, 51
    overlay feature, 56
    RGB channel, 54
    Save Selection dialog, 53
    scissors tool
        brush tool, 62
        close curve pointer, 62
        curve fit algorithm, 59
        edge detection, 60
        GIMP, 63
        nodes, 60
        Toggle Quick Mask, 62
    shift modifier, 52
Mathematical
        matriculation, 16
Matte option, 33
Modal operation, 95
    GIMP, 100
        Layer Mask, 102
        Ring Drop Shadow, 103
    Photoshop, 96
        Custom Layer Mask, 98
        Layer Mask, 98
        Magic Wand Tool, 97
    pipeline processing, 96

O

Optimization process
    GIMP, 144
    in Photoshop
        highest-quality result
            (PNG-24 selection), 136–137
        indexed color format, 137
        layer visibility, 141
        lossy compression, 139
        start up, 135–136

P, Q

Photoshop
    cool effects, 86
        Hard Light, 88
        Layer Style, 90
        Linear Burn, 88
        Luminosity, 90
        Soft Light, 87
    Custom Layer Mask, 98
    Layer Mask, 98
    Magic Wand Tool, 97
Photoshop optimization process
    highest-quality result (PNG-24
            selection), 136–137
    indexed color format, 137
    layer visibility, 141
    lossy compression, 139
    start up, 135–136

Pixel blending modes, 79
  in GIMP, 81–86
  in Photoshop, 81, 86–92
pluginFilter.xcf file, 123
PorterDuff Class, 159–161
Porter Duff modes theory. *See* Pixel blending modes
PyGIMP, 153
Python, 153

R

Resolution
  definition, 7
  device resolutions, 8
  GIMP, 11
  image pixels, 8
  image size, 9
  re-sampling, 12
  statistics, 10
Restrictive algorithms, 33
RGB (red, green, and blue), 23, 26–29
RingElement layer, 127
RingShadow layer, 127

S

Scheme, 153
Scissors tool
  brush tool, 62
  close curve pointer, 62
  curve fit algorithm, 59
  edge detection, 60
  GIMP, 63
  nodes, 60
  Toggle Quick Mask, 62
Script-Fu, 153
Separate+ (Plus), 26
.setImageBitmap() method, 161
.setXfermode() method, 160
Subtractive color model, 24

T, U, V

Tagged Image File Format (TIFF), 32
Targa (TGA), 32

W, X, Y, Z

WebKit, 157