

The #1 WPF Book—Now Updated for WPF 4!

Adam Nathan

Full Color

Code samples
appear as they do
in Visual Studio!

WPF 4

UNLEASHED



SAMS

WPF 4 Unleashed

Copyright © 2010 by Pearson Education

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-33119-0

ISBN-10: 0-672-33119-5

Library of Congress Cataloging-in-Publication Data

Nathan, Adam.

WPF 4 unleashed / Adam Nathan.

p. cm.

Includes index.

ISBN 978-0-672-33119-0

1. Windows presentation foundation. 2. Application software. 3. Microsoft .NET Framework. I. Title.

QA76.76.A65N386 2010

006.7'882—dc22

2010017765

Printed in the United States of America

First Printing June 2010

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author(s) and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales

international@pearsoned.com

Editor-in-Chief

Karen Gettman

Executive Editor

Neil Rowe

Development Editor

Mark Renfrow

Managing Editor

Kristy Hart

Project Editor

Betsy Harris

Copy Editor

Kitty Wilson

Indexer

Erika Millen

Proofreader

Kathy Ruiz

Technical Editors

Dwayne Need

Robert Hogue

Joe Castro

Jordan Parker

Publishing Coordinator

Cindy Teeters

Book Designer

Gary Adair

Composition

Bronkella Publishing LLC

Introduction

Thank you for picking up *WPF 4 Unleashed!* Windows Presentation Foundation (WPF) is Microsoft's premier technology for creating Windows graphical user interfaces, whether they consist of plain forms, document-centric windows, animated cartoons, videos, immersive 3D environments, or all of the above. WPF is a technology that makes it easier than ever to create a broad range of applications. It's also the basis for Silverlight, which has extended WPF technology onto the Web and into devices such as Windows phones.

Ever since WPF was publicly announced in 2003 (with the code name "Avalon"), it has gotten considerable attention for the ways in which it revolutionizes the process of creating software—especially for Windows programmers used to Windows Forms and GDI. It's relatively easy to create fun, useful, and shareable WPF samples that demonstrate all kinds of techniques that are difficult to accomplish in other technologies. WPF 4, released in April 2010, improves on previous versions of WPF in just about every dimension.

WPF is quite a departure from previous technologies in terms of its programming model, underlying concepts, and basic terminology. Even viewing the source code for WPF (by cracking open its components with a tool such as .NET Reflector) is a confusing experience because the code you're looking for often doesn't reside where you'd expect to find it. When you combine all this with the fact that there are often several ways to accomplish any task in WPF, you arrive at a conclusion shared by many: *WPF has a very steep learning curve.*

That's where this book comes in. As WPF was developed, it was obvious that there would be no shortage of WPF books in the marketplace. But it wasn't clear to me that the books would have the right balance to guide people through the technology and its unique concepts while showing practical ways to exploit it. Therefore, I wrote the first edition of this book, *Windows Presentation Foundation Unleashed*, with the following goals in mind:

- ▶ To provide a solid grounding in the underlying concepts, in a practical and approachable fashion
- ▶ To answer the questions most people have when learning the technology and to show how commonly desired tasks are accomplished
- ▶ To be an authoritative source, thanks to input from members of the WPF team who designed, implemented, and tested the technology
- ▶ To be clear about where the technology falls short rather than selling the technology as the answer to all problems
- ▶ To be an easily navigated reference that you can constantly come back to

The first edition of this book was far more successful than I ever imagined it would be. Now, almost four years later, I believe that this second edition accomplishes all the same

goals but with even more depth. In addition to covering new features introduced in WPF 3.5, WPF 3.5 SP1, and WPF 4, it expands the coverage of the existing features from the first version of WPF. Whether you're new to WPF or a long-time WPF developer, I hope you find this book to exhibit all these attributes.

Who Should Read This Book?

This book is for software developers who are interested in creating user interfaces for Windows. Regardless of whether you're creating line-of-business applications, consumer-facing applications, or reusable controls, this book contains a lot of content that helps you get the most out of the platform. It's designed to be understandable even for folks who are new to the .NET Framework. And if you are already well versed in WPF, I'm confident that this book still has information for you. At the very least, it should be an invaluable reference for your bookshelf.

Because the technology and concepts behind WPF are the same ones behind Silverlight, reading this book can also make you a better developer for Windows Phone 7 and even a better web developer.

Although this book's content is not optimized for graphic designers, reading this book can be a great way to understand more of the "guts" behind a product like Microsoft Expression Blend.

To summarize, this book does the following:

- ▶ Covers everything you need to know about Extensible Application Markup Language (XAML), the XML-based language for creating declarative user interfaces that can be easily restyled
- ▶ Examines the WPF feature areas in incredible depth: controls, layout, resources, data binding, styling, graphics, animation, and more
- ▶ Highlights the latest features, such as multi-touch, text rendering improvements, new controls, XAML language enhancements, the Visual State Manager, easing functions, and much more
- ▶ Delves into topics that aren't covered by most books: 3D, speech, audio/video, documents, effects, and more
- ▶ Shows how to create popular user interface elements, such as galleries, ScreenTips, custom control layouts, and more
- ▶ Demonstrates how to create sophisticated user interface mechanisms, such as Visual Studio-like collapsible/dockable panes
- ▶ Explains how to develop and deploy all types of applications, including navigation-based applications, applications hosted in a web browser, and applications with great-looking nonrectangular windows
- ▶ Explains how to create first-class custom controls for WPF

- ▶ Demonstrates how to create hybrid WPF software that leverages Windows Forms, DirectX, ActiveX, or other non-WPF technologies
- ▶ Explains how to exploit new Windows 7 features in WPF applications, such as Jump Lists, and how to go beyond some of the limitations of WPF

This book doesn't cover every last bit of WPF. (In particular, XML Paper Specification [XPS] documents are given only a small bit of attention.) WPF's surface area is so large that I don't believe any single book can. But I think you'll be pleased with the breadth and depth achieved by this book.

Examples in this book appear in XAML and C#, plus C++/CLI for interoperability discussions. XAML is used heavily for a number of reasons: It's often the most concise way to express source code, it can often be pasted into lightweight tools to see instant results without any compilation, WPF-based tools generate XAML rather than procedural code, and XAML is applicable no matter what .NET language you use, such as Visual Basic instead of C#. Whenever the mapping between XAML and a language such as C# is not obvious, examples are shown in both representations.

Software Requirements

This book targets the final release of version 4.0 of Windows Presentation Foundation, the corresponding Windows SDK, and Visual Studio 2010.

The following software is required:

- ▶ A version of Windows that supports the .NET Framework 4.0. This can be Windows XP with Service Pack 2 (including Media Center, Tablet PC, and x64 editions), Windows Server 2003 with Service Pack 1 (including the R2 edition), Windows Vista, or later versions.
- ▶ The .NET Framework 4.0, which is installed by default starting with Windows Vista. For earlier versions of Windows, you can download the .NET Framework 4.0 for free from <http://msdn.com>.

In addition, the following software is recommended:

- ▶ The Windows Software Development Kit (SDK), specifically the .NET tools it includes. This is also a free download from <http://msdn.com>.
- ▶ Visual Studio 2010 or later, which can be a free Express edition downloaded from <http://msdn.com>.

If you want additional tool support for WPF-based graphic design, Microsoft Expression (specifically Expression Blend) can be extremely helpful.

A few examples are specific to Windows Vista, Windows 7, or a computer that supports multi-touch, but the rest of the book applies equally to all relevant versions of Windows.

Code Examples

The source code for examples in this book can be downloaded from <http://informat.com/title/9780672331190> or <http://adamnathan.net/wpf>.

How This Book Is Organized

This book is arranged into six main parts, representing the progression of feature areas that you typically need to understand to use WPF effectively. But if you're dying to jump ahead and learn about a topic such as 3D or animation, the book is set up to allow for nonlinear journeys as well. The following sections provide a summary of each part.

Part I: Background

This part includes the following chapters:

- ▶ Chapter 1: Why WPF, and What About Silverlight?
- ▶ Chapter 2: XAML Demystified
- ▶ Chapter 3: WPF Fundamentals

Chapter 1 introduces WPF by comparing it to alternative technologies and helping you make decisions about when WPF is appropriate for your needs. Chapter 2 explores XAML in great depth, giving you the foundation to understand the XAML you'll encounter in the rest of the book and in real life. Chapter 3 highlights the most unique pieces of WPF's programming model above and beyond what .NET programmers already understand.

Part II: Building a WPF Application

This part includes the following chapters:

- ▶ Chapter 4: Sizing, Positioning, and Transforming Elements
- ▶ Chapter 5: Layout with Panels
- ▶ Chapter 6: Input Events: Keyboard, Mouse, Stylus, and Multi-Touch
- ▶ Chapter 7: Structuring and Deploying an Application
- ▶ Chapter 8: Exploiting Windows 7

Part II equips you with the knowledge to assemble and deploy a traditional-looking application (although some fancier effects, such as transforms, nonrectangular windows, and Aero Glass, are also covered). Chapters 4 and 5 discuss arranging controls (and other elements) in a user interface. Chapter 6 covers input events, including new support for engaging multi-touch user interfaces. Chapter 7 examines several different ways to package and deploy WPF-based user interfaces to make complete applications. Chapter 8 ends this part by showing slick ways to exploit features in Windows 7 that can help make your application look modern.

Part III: Controls

This part includes the following chapters:

- ▶ Chapter 9: Content Controls
- ▶ Chapter 10: Items Controls
- ▶ Chapter 11: Images, Text, and Other Controls

Part III provides a tour of controls built into WPF. There are many that you'd expect to have available, plus several that you might not expect. Two categories of controls—content controls (Chapter 9) and items controls (Chapter 10)—are important and deep enough topics to merit their own chapters. The rest of the controls are examined in Chapter 11.

Part IV: Features for Professional Developers

This part includes the following chapters:

- ▶ Chapter 12: Resources
- ▶ Chapter 13: Data Binding
- ▶ Chapter 14: Styles, Templates, Skins, and Themes

The features covered in Part IV are not always necessary to use in WPF applications, but they can greatly enhance the development process. Therefore, they are indispensable for professional developers who are serious about creating maintainable and robust applications or components. These topics are less about the results visible to end users than they are about the best practices for accomplishing these results.

Part V: Rich Media

This part includes the following chapters:

- ▶ Chapter 15: 2D Graphics
- ▶ Chapter 16: 3D Graphics
- ▶ Chapter 17: Animation
- ▶ Chapter 18: Audio, Video, and Speech

This part of the book covers the features in WPF that typically get the most attention. The support for 2D and 3D graphics, animation, video, and more enable you to create a stunning experience. These features—and the way they are exposed—set WPF apart from previous systems. WPF lowers the barrier to incorporating such content in your software, so you might try some of these features that you never would have dared to try in the past!

Part VI: Advanced Topics

This part includes the following chapters:

- ▶ Chapter 19: Interoperability with Non-WPF Technologies
- ▶ Chapter 20: User Controls and Custom Controls
- ▶ Chapter 21: Layout with Custom Panels

The topics covered in Part VI are relevant for advanced application developers, or developers of WPF-based controls. The fact that existing WPF controls can be radically restyled greatly reduces the need for creating custom controls.

Conventions Used in This Book

Various typefaces in this book identify new terms and other special items. These typefaces include the following:

Typeface	Meaning
<i>Italic</i>	Italic is used for new terms or phrases when they are initially defined and occasionally for emphasis.
Monospace	Monospace is used for screen messages, code listings, and command samples, as well as filenames. In code listings, <i>italic monospace type</i> is used for placeholder text. Code listings are colorized similar to the way they are colorized in Visual Studio. Blue monospace type is used for XML elements and C#/C++ keywords, brown monospace type is used for XML element names and C#/C++ strings, green monospace type is used for comments, red monospace type is used for XML attributes, and teal monospace type is used for type names in C# and C++.

Throughout this book, you'll find a number of sidebar elements:

FAQ



What is a FAQ sidebar?

A FAQ sidebar presents a question readers might have regarding the subject matter in a particular spot in the book—and then provides a concise answer.

DIGGING DEEPER

Digging Deeper Sidebars

A Digging Deeper sidebar presents advanced or more detailed information on a subject than is provided in the surrounding text. Think of Digging Deeper material as stuff you can look into if you're curious but can ignore if you're not.

TIP

A tip is a bit of information that can help you in a real-world situation. Tips often offer shortcuts or alternative approaches to produce better results or to make a task easier or quicker.

WARNING

A warning alerts you to an action or a condition that can lead to an unexpected or unpredictable result—and then tells you how to avoid it.

CHAPTER 1

Why WPF, and What About Silverlight?

In movies and on TV, the main characters are typically an exaggeration of the people you encounter in real life. They're more attractive, they react more quickly, and they somehow always know exactly what to do. The same could be said about the software they use.

This first struck me back in 1994 when watching the movie *Disclosure*, starring Michael Douglas, Demi Moore, and an email program that looks nothing like Microsoft Outlook! Throughout the movie, we're treated to various visual features of the program: a spinning three-dimensional "e," messages that unfold when you open them and crumple when you delete them, hints of inking support, and slick animations when you print messages. (The email program isn't even the most unrealistic software in the movie. I'll just say "virtual reality database" and leave it at that.)

Usability issues aside, Hollywood has been telling us for a long time that software in the real world isn't as compelling as it should be. You can probably think of several examples on your own of TV shows and movies with comically unrealistic software. But lately, real-world software has been catching up to Hollywood's standards! You can already see it in traditional operating systems (yes, even in Windows), on the web, and in software for devices such as the iPhone, iPad, Zune, TiVo, Wii, Xbox, Windows phones, and many more. Users have increasing expectations for the experience of using software, and companies are spending a great deal of time and money on user interfaces that differentiate themselves from the competition. This isn't limited to consumer-facing software; even business applications and internal tools can greatly benefit from a polished user interface.

IN THIS CHAPTER

- ▶ A Look at the Past
- ▶ Enter WPF
- ▶ The Evolution of WPF
- ▶ What About Silverlight?

With higher demands placed on user interfaces, traditional software development processes and technologies often fall short. Modern software usually needs to support rapid iteration and major user interface changes throughout the process—whether such changes are driven by professional graphic designers, developers with a knack for designing user interfaces, or a boss who wants the product to be more “shiny” and animated. For this to be successful, you need technology and tools that make it natural to separate the user interface from the rest of the implementation as much as possible and to decouple visual behavior from the underlying program logic. Developers should be able to create a fully functional “ugly” application that designers can directly retheme without requiring developers to translate their artwork. The Win32 style of programming, in which controls directly contain code to paint and repaint themselves, makes rapid user interface iteration far too difficult for most projects.

In 2006, Microsoft released a technology to help people create 21st-century software that meets these high demands: Windows Presentation Foundation (WPF). With the release of WPF 4 in 2010, the technology is better than ever at delivering amazing results for just about any kind of software. Almost a decade after Tom Cruise helped popularize the idea of multi-touch computer input in the movie *Minority Report*, and after successful multi-touch implementations in a variety of devices (most notably the iPhone), WPF 4 and Windows 7 are bringing multi-touch to the masses. Hollywood better start coming up with some fresh ideas!

A Look at the Past

The primary technologies behind many Windows-based user interfaces—the graphics device interface (GDI) and USER subsystems—were introduced with Windows 1.0 in 1985. That’s almost prehistoric in the world of technology! In the early 1990s, OpenGL (created by Silicon Graphics) became a popular graphics library for doing advanced two-dimensional (2D) and three-dimensional (3D) graphics on both Windows and non-Windows systems. This was leveraged by people creating computer-aided design (CAD) programs, scientific visualization programs, and games. DirectX, a Microsoft technology introduced in 1995, provided a new high-performance alternative for 2D graphics, input, communication, sound, and eventually 3D (introduced with DirectX 2 in 1996).

Over the years, many enhancements have been made to both GDI and DirectX. GDI+, introduced in the Windows XP time frame, tried to improve upon GDI by adding support for features such as alpha blending and gradient brushes. It ended up being slower than GDI due to its complexity and lack of hardware acceleration. DirectX (which, by the way, is the technology behind Xbox) continually comes out with new versions that push the limits of what can be done with computer graphics. With the introduction of .NET and managed code in 2002, developers were treated to a highly productive model for creating Windows (and web) applications. In this world, Windows Forms (built on top of GDI+) became the primary way a C#, Visual Basic, and (to a lesser degree) C++ developer started to create new user interfaces on Windows. Windows Forms has been a successful and productive technology, but it still has all the fundamental limitations of GDI+ and USER.

Starting with DirectX 9, Microsoft shipped a DirectX framework for managed code (much like it shipped libraries specifically for Visual Basic in the past), which eventually was supplanted by the XNA Framework. Although this enables C# developers to use DirectX without most of the complications of .NET/COM interoperability, these managed frameworks aren't significantly easier to use than their unmanaged counterparts unless you're writing a game. (The XNA Framework makes writing a game easier because it includes new libraries specifically for game development and works with compelling tools such as the XNA Framework Content Pipeline and XNA Game Studio Express.)

So although you could have developed a Windows-based email program with the 3D effects seen in *Disclosure* ever since the mid-1990s with non-GDI technologies (actually, probably mixing DirectX or OpenGL with GDI), such technologies are rarely used in mainstream Windows applications even more than a decade later. There are several reasons for this: The hardware required to get a decent experience hasn't been ubiquitous until recently, it has been at least an order of magnitude harder to use alternative technologies, and GDI-based experiences have been considered "good enough."

Graphics hardware continues to get better and cheaper and consumer expectations continue to rise, but until WPF, the difficulty of creating modern user experiences had not been addressed. Some developers would take matters into their own hands to get cooler-looking applications and controls on Windows. A simple example of this is using bitmaps for buttons instead of using the standard button control. These types of customizations can not only be expensive to develop, but they also often produce a flakier experience. Such applications often aren't as accessible as they should be, don't handle high dots-per-inch (DPI) settings very well, and have other visual glitches.

Enter WPF

Microsoft recognized that something brand new was needed that escaped the limitations of GDI+ and USER yet provided the kind of productivity that people enjoy with frameworks like Windows Forms. And with the continual rise of cross-platform applications based on HTML and JavaScript, Windows desperately needed a technology that's as fun and easy to use as these, yet with the power to exploit the capabilities of the local computer. Windows Presentation Foundation (WPF) is the answer for software developers and graphic designers who want to create modern user experiences without having to master several difficult technologies. Although "Presentation" sounds like a lofty term for what I would simply call a user interface, it's probably more appropriate for describing the higher level of visual polish that's expected of today's applications and the wide range of functionality included in WPF!

The highlights of WPF include the following:

- ▶ **Broad integration**—Prior to WPF, a Windows developer who wanted to use 3D, video, speech, and rich document viewing in addition to normal 2D graphics and controls would have to learn several independent technologies with a number of inconsistencies and attempt to blend them together without much built-in support. But WPF covers all these areas with a consistent programming model as well as tight integration when each type of media gets composited and rendered. You can apply

the same kind of effects consistently across different media types, and many of the techniques you learn in one area apply to all the other areas.

- ▶ **Resolution independence**—Imagine a world in which moving to a higher resolution or DPI setting doesn't mean that everything gets smaller; instead, graphics and text simply get crisper! Envision user interfaces that look reasonable on a small netbook as well as on a 60-inch TV! WPF makes this easy and gives you the power to shrink or enlarge elements on the screen independently from the screen's resolution. A lot of this is possible because of WPF's emphasis on vector graphics.
- ▶ **Hardware acceleration**—WPF is built on Direct3D, so content in a WPF application—whether 2D or 3D, graphics, or text—is converted to 3D triangles, textures, and other Direct3D objects and then rendered by hardware. This means that WPF applications get the benefits of hardware acceleration for smoother graphics and all-around better performance (due to work being offloaded to graphics processing units [GPUs] instead of central processor units [CPUs]). It also ensures that all WPF applications (not just high-end games) receive benefits from new hardware and drivers, whose advances typically focus on 3D capabilities. But WPF doesn't *require* high-end graphics hardware; it has a software rendering pipeline as well. This enables features not yet supported by hardware, enables high-fidelity printing of any content on the screen, and is used as a fallback mechanism when encountering inadequate hardware resources (such as an outdated graphics card or even a high-end one that has simply run out of GPU resources such as video memory).
- ▶ **Declarative programming**—Declarative programming is not unique to WPF, as Win16/Win32 programs have used declarative resource scripts to define the layout of dialog boxes and menus for over 25 years. And .NET programs of all types often leverage declarative custom attributes plus configuration and resource files based on Extensible Markup Language (XML). But WPF takes declarative programming to the next level with Extensible *Application* Markup Language (XAML; pronounced “Zammel”). The combination of WPF and XAML is similar to using HTML to define a user interface—but with an incredible range of expressiveness. This expressiveness even extends beyond the bounds of user interfaces; WPF uses XAML as a document format, a representation of 3D models, and more. The result is that graphic designers are empowered to contribute directly to the look and feel of applications, as well as some behavior for which you'd typically expect to have to write code. The next chapter examines XAML in depth.
- ▶ **Rich composition and customization**—WPF controls can be composed in ways never before seen. You can create a ComboBox filled with animated Buttons or a Menu filled with live video clips! Although these particular customizations might sound horrible, it's important that you don't have to write a bunch of code (or any code!) to customize controls in ways that the control authors never imagined (unlike owner-draw in prior technologies). Along the same lines, WPF makes it quite easy to “skin” applications with radically different looks (covered in Chapter 14, “Styles, Templates, Skins, and Themes”).

In short, WPF aims to combine the best attributes of systems such as DirectX (3D and hardware acceleration), Windows Forms (developer productivity), Adobe Flash (powerful animation support), and HTML (declarative markup). With the help of this book, I think you'll find that WPF gives you more productivity, power, and fun than any other technology you've worked with in the past!

DIGGING DEEPER

GDI and Hardware Acceleration

GDI is actually hardware accelerated on Windows XP. The video driver model explicitly supported accelerating common GDI operations. Windows Vista introduced a new video driver model that does not hardware accelerate GDI primitives. Instead, it uses a “canonical display device” software implementation of the legacy video driver for GDI. However, Windows 7 reintroduced partial hardware acceleration for GDI primitives.

FAQ



Does WPF enable me to do something that I couldn't have previously done?

Technically, the answer is “No,” just like C# and the .NET Framework don't enable you to do something that you couldn't do in assembly code. It's just a question of how much work you want to do to get the desired results!

If you were to attempt to build a WPF-equivalent application from scratch without WPF, you'd not only have to worry about the drawing of pixels on the screen and interaction with input devices, you'd also need to do a ton of additional work to get the accessibility and localization support that's built in to WPF, and so on. WPF also provides the easiest way to take advantage of Windows 7 features, such as defining Jump List items with a small chunk of XAML (see Chapter 8, “Exploiting Windows 7”).

So I think most people would agree that the answer is “Yes” when you factor time and money into the equation!

FAQ



When should I use DirectX instead of WPF?

DirectX is more appropriate than WPF for advanced developers writing hard-core “twitch games” or applications with complex 3D models where you need maximum performance. That said, it's easy to write a naive DirectX application that performs far worse than a similar WPF application.

DirectX is a low-level interface to the graphics hardware that exposes all the quirks of whatever GPU a particular computer has. DirectX can be thought of as assembly language in the world of graphics: You can do anything the GPU supports, but it's up to you (the application author) to support all the hardware variations. This is onerous, but such low-level hardware access enables skilled developers to make their own tradeoffs between fine-grained quality and speed. In addition, DirectX exposes cutting-edge features of GPUs as they emerge more quickly than they appear in WPF.

Continued

In contrast, WPF provides a high-level abstraction that takes a description of a scene and figures out the best way to render it, given the hardware resources available. (It's a *retained mode* system rather than an *immediate mode* system.) 2D is the primary focus of WPF; its 3D support is focused on data visualization scenarios and integration with 2D rather than supporting the full power of DirectX.

The downside of choosing DirectX over WPF is a potentially astronomical increase in development cost. A large part of this cost is the requirement to test an application on each driver/GPU combination you intend to support. One of the major benefits of building on top of the WPF is that Microsoft has already done this testing for you! You can instead focus your testing on low-end hardware for measuring performance. The fact that WPF applications can even leverage the client GPU in a partial-trust environment is also a compelling differentiator.

Note that you are able to use both DirectX and WPF in the same application. Chapter 19, "Interoperability with Non-WPF Technologies," shows how this can be done.

The Evolution of WPF

Oddly enough, WPF 4 is the fourth major release of WPF. It's odd because the first release had the version number 3.0! The first release in November 2006 was called WPF 3.0 because it shipped as part of the .NET Framework 3.0. The second release—WPF 3.5—came almost exactly a year later (one day shy, in fact). The third release, once again, came almost a year later (in August 2008). This release was a part of Service Pack 1 (SP1) for .NET 3.5, but this was no ordinary service pack as far as WPF was concerned—it contained many new features and improvements.

In addition to these major releases, Microsoft introduced a "WPF Toolkit" in August 2008 at <http://wpf.codeplex.com> that, along with miscellaneous tools and samples, gets updated several times a year. The WPF Toolkit has been used as a way to ship features more quickly and in an experimental form (often with full source code). Features introduced in the WPF Toolkit often "graduate" to get included in a future release of WPF, based on customer feedback about their desirability and readiness.

When the first version of WPF was released, tool support was almost nonexistent. The following months brought primitive WPF extensions for Visual Studio 2005 and the first public preview release of Expression Blend. Now, Visual Studio 2010 not only has first-class support for WPF development but has been substantially rewritten to be a WPF application itself! Expression Blend, an application built 100% with WPF, has also gained a lot of functionality for designing and prototyping great user interfaces. And in the past several years, numerous WPF-based applications have been released from companies such as Autodesk, SAP, Disney, Blockbuster, Roxio, AMD, Hewlett Packard, Lenovo, and many more. Microsoft itself, of course, has a long list of software built with WPF (Visual Studio, Expression, Test and Lab Manager, Deep Zoom Composer, Songsmith, Surface, Semblio, Robotics Studio, LifeCam, Amalga, Games for Windows LIVE Marketplace, Office

TIP

To inspect the WPF elements used in any WPF-based application, you can use the Snoop tool available from <http://snoopwpf.codeplex.com>.

Communicator Attendant, Active Directory Administrative Center, Dynamics NAV, Pivot, PowerShell ISE, and many more).

Let's take a closer look at how WPF has changed over time.

Enhancements in WPF 3.5 and WPF 3.5 SP1

The following notable changes were made to WPF in versions 3.5 and 3.5 SP1:

- ▶ **Interactive 3D**—The worlds of 2D and 3D were woven together even more seamlessly with the `UIElement3D` base class, which gives 3D elements input, focus, and events; the odd-sounding `Viewport2DVisual3D` class, which can place any interactive 2D controls inside a 3D scene; and more. See Chapter 16, “3D Graphics.”
- ▶ **First-class interoperability with DirectX**—Previously, WPF applications could only interoperate with DirectX via the lowest common denominator of Win32. Now, WPF has functionality for interacting directly with Direct3D surfaces with the `D3DImage` class rather than being forced to interact with its host `HWND`. One benefit from this is the ability to place WPF content on top of DirectX content and vice versa. See Chapter 19.
- ▶ **Better data binding**—WPF gained support for XLINQ binding, better validation and debugging, and output string formatting in XAML that reduces the need for custom procedural code. See Chapter 13, “Data Binding.”
- ▶ **Better special effects**—The first version of WPF shipped with a handful of bitmap effects (blur, drop shadow, outer glow, emboss, and bevel) but with a warning to not use them because their performance was so poor! This has changed, with a new set of hardware-accelerated effects and a whole new architecture that allows you to plug in your own custom hardware-accelerated effects via pixel shaders. See Chapter 15, “2D Graphics.”
- ▶ **High-performance custom drawing**—WPF didn't previously have a good answer for custom drawings that involve thousands of points or shapes, as even the lowest-level drawing primitives have too much overhead to make such things perform well. The `WriteableBitmap` class was enhanced so you can now specify dirty regions when drawing on it rather than getting a whole new bitmap every frame! Because `WriteableBitmap` only lets you set pixels, it is a very primitive form of “drawing,” however.
- ▶ **Text improvements**—There's now better performance, better international support (improved input method editor [IME] support and improved Indic script support), and enhancements to `TextBox` and `RichTextBox`. See Chapter 11, “Images, Text, and Other Controls.”
- ▶ **Enhancements to partial-trust applications**—More functionality became available in the partial-trust sandbox for .NET applications, such as the ability to use Windows Communication Foundation (WCF) for web service calls (via `basicHttpBinding`) and the ability to read and write HTTP cookies. Also, support for XAML Browser Applications (XBAPs)—the primary mechanism for running partial-trust

WPF applications—was extended to the Firefox web browser instead of just Internet Explorer (In WPF, however, the add-on that enables this is no longer installed by default.)

- ▶ **Improved deployment for applications and the .NET Framework**—This arrived in many forms: a smaller and faster .NET Framework installation process thanks to the beginnings of a .NET Framework “client profile” that excludes server-only .NET pieces such as ASP.NET; a new “bootstrapper” component that handles all .NET Framework dependencies, installations, and upgrades for you as well as enabling setups with custom branding; and a variety of new ClickOnce features.
- ▶ **Improved performance**—WPF and the underlying common language runtime implemented several changes that significantly boosted the performance of WPF applications without any code changes needed. For example, the load time (especially first-time load) has been dramatically improved, animations (especially slow ones) are much smoother, data binding is faster in a number of scenarios, and layered windows (described in Chapter 8) are now hardware accelerated. Other performance improvements were made that you must opt into due to compatibility constraints, such as improved virtualization and deferred scrolling in items controls, described in Chapter 10, “Items Controls.”

Enhancements in WPF 4

WPF 4 brings the following changes, on top of the changes from previous versions:

- ▶ **Multi-touch support**—When running on computers that support multi-touch and run Windows 7 or later, WPF elements can get a variety of input events, from low-level data, to easy-to-consume manipulations (such as rotation and scaling), to high-level—including custom—gestures. The built-in WPF controls have also been updated to be multi-touch aware. The WPF team leveraged the work previously done by the Microsoft Surface team (whose software is built on WPF). As a result, multi-touch in WPF 4 is compatible with version 2 of the Surface SDK, which is great news for anyone considering developing for both Windows and Surface. See Chapter 6, “Input Events: Keyboard, Mouse, Stylus, and Multi-Touch.”
- ▶ **First-class support for other Windows 7 features**—Multi-touch is a cool new feature of Windows 7, but there are plenty of others that don’t require special hardware—so many more users will appreciate their inclusion. WPF provides the best way to integrate with new taskbar features such as Jump Lists and icon overlays, integrate with the latest common dialogs, and more. See Chapter 8.
- ▶ **New controls**—WPF 4 includes controls such as DataGrid, Calendar, and DatePicker, which originally debuted in the WPF Toolkit. See Chapter 11.
- ▶ **Easing animation functions**—Eleven new animation classes such as BounceEase, ElasticEase, and SineEase enable sophisticated animations with custom rates of acceleration and deceleration to be performed completely declaratively. These “easing functions” and their infrastructure were first introduced in Silverlight 3 before being adopted by WPF 4.

- ▶ **Enhanced styling with Visual State Manager**—The Visual State Manager, originally introduced in Silverlight 2, provides a new way to organize visuals and their interactivity into “visual states” and “state transitions.” This feature makes it easier for designers to work with controls in tools such as Expression Blend, but importantly also makes it easier to share templates between WPF and Silverlight.
- ▶ **Improved layout on pixel boundaries**—WPF straddles the line between being automatically DPI independent (which requires ignoring physical pixel boundaries) and having visual elements that look crisp (which, especially for small elements, requires being aligned on pixel boundaries). From the beginning, WPF has supported a property called `SnapsToDevicePixels` that forces “pixel snapping” on elements. But using `SnapsToDevicePixels` can be complex and doesn’t help in some scenarios. Silverlight went back to the drawing board and created a property called `UseLayoutRounding` that works more naturally. WPF 4 now has this property. Just set it to `true` on a root element, and the positions of that element plus all of children will be rounded up or down to lie on pixel boundaries. The result is user interfaces that can scale *and* can easily be crisp!
- ▶ **Non-blurry text**—WPF’s emphasis on DPI independence and a scalable user interface has been an issue for small text—the kind of text that occurs a lot in traditional user interfaces on 96-DPI screens. This has frustrated numerous users and developers. In fact, I’ve always claimed that I can spot a user interface created with WPF simply by looking at the blurriness of its text. WPF 4 has finally addressed this with an alternative way to render text that can make it look as crisp as GDI-based text yet with almost all the benefits that WPF brings. Visual Studio 2010, for example, uses this rendering mode for its text documents. Because there are some limitations to the new rendering approach, you must opt into it. See Chapter 11.
- ▶ **More deployment improvements**—The .NET Framework client profile can run side-by-side with the full .NET Framework, and it can be used in just about every scenario relevant for WPF applications. In fact, .NET 4.0 projects in Visual Studio 2010 target the smaller client profile by default.
- ▶ **More performance improvements**—In order to make vector graphics perform as well as possible, WPF can cache rendered results as bitmaps and reuse them. For advanced scenarios, you can control this behavior with the new `CacheMode` property. See Chapter 15. The heavy usage of WPF in Visual Studio 2010 drove a lot of miscellaneous performance improvements into WPF 4 across the board, but all WPF applications get to enjoy these improvements.

FAQ



What will be added to WPF after version 4?

Nothing has been announced at the time of writing, but I think it’s safe to say that performance and increased synergy with Silverlight will continue to be two major themes of WPF’s evolution. Plus, the WPF Toolkit provides some clues to future features that could be integrated into the core platform, such as chart controls, a `BreadcrumbBar` control, a `NumericUpDown` control, and more.

FAQ**Are there any differences with WPF, depending on the version of Windows?**

WPF exposes APIs that are relevant only for Windows 7 and later, such as multi-touch functionality and various features described in Chapter 8. Besides that, WPF has a few behavioral differences when running on Windows XP (the oldest version of Windows that WPF supports). For example, 3D objects do not get antialiased.

And, of course, WPF controls have different default themes to match their host operating system (Aero on Windows Vista and Windows 7 versus Luna on Windows XP).

Windows XP also has an older driver model that can negatively impact WPF applications. The driver model in later versions of Windows virtualizes and schedules GPU resources, making a system perform better when multiple GPU-intensive programs are running. Running multiple WPF or DirectX applications might bog down a Windows XP system but shouldn't cause performance issues on more recent versions of Windows.

What About Silverlight?

Silverlight is a small, lightweight version of the .NET Framework targeted at rich web scenarios (as an alternative to Adobe Flash and Flex, for example). Silverlight chose to follow WPF's approach to creating user interfaces rather than creating yet another distinct technology—and this approach has some great benefits. It was first released in 2007 and, like WPF, is already in its fourth major version. Silverlight 4 was released in April 2010, a few days after the release of WPF 4.

The relationship between WPF and Silverlight is a bit complex, and there is some confusion about when to use one technology versus the other. This is further exacerbated by the fact that WPF applications can run inside a web browser (as XBAPs) and be just as “web based” as Silverlight applications, and Silverlight applications can run outside a web browser, even in an offline mode.

Silverlight is mostly a subset of WPF plus the most fundamental classes in the .NET Framework (core data types, collection classes, and so on). Each new version of Silverlight includes more and more WPF functionality. Although compatibility with WPF and the full .NET Framework is a goal for Silverlight, its creators have taken some opportunities to learn from mistakes made in WPF and the .NET Framework. They have made some changes and begun to support new features that don't yet exist in the full .NET Framework. Some of these changes or additions have been later adopted by WPF and the full .NET Framework (such as the Visual State Manager and layout rounding), but others have not (such as video brushes and perspective transforms). There are parts of WPF and the .NET Framework that Silverlight will probably never support.

The bottom line is that the question to ask yourself isn't “Should I use WPF or Silverlight?” but rather, “Should I use the full .NET Framework or the small .NET Framework?” If you will require functionality that exists only in the full .NET Framework, then the choice is pretty simple. And WPF is the recommended user interface technology to use with the full .NET Framework. Similarly, if the ability to run on a Mac or devices

other than a standard PC is a requirement, then the choice is also clear. And Silverlight has only one user interface technology (although it interoperates with HTML nicely). Otherwise, the best choice depends greatly on the nature of the software and the target audience.

Ideally, you wouldn't have to make an up-front choice of which framework you want to target. Ideally, you could use the same codebase—even the same compiled binaries—and have an easy way to morph the application to exploit capabilities of the underlying device, whether your program is running on a mobile device, a full Windows PC, or a Mac. Maybe one day that will be true, but in the meantime, having a common codebase that can work for both WPF and Silverlight involves a bit of work. The most common approach has been to create a Silverlight-compatible codebase with `#ifdef` blocks for WPF-specific functionality, so you can compile separately for Silverlight versus WPF with minimal divergence in code.

It is my expectation (and hope) that the distinction between WPF and Silverlight will fade over time. While Silverlight is a much cooler name than Windows Presentation Foundation, the fact that these technologies have different names causes trouble and artificial distinctions. The way to think of Silverlight and WPF is as two implementations of the same basic technology. In fact, inside Microsoft, largely the same team works on both. Microsoft talks a lot about having a “client continuum” to target all platforms and devices with common skills (what you learn in this book), common tools (Visual Studio, Expression Blend, and others), and at least common *code* (a .NET language such as C# or VB along with XAML, for example) if not common *binaries*. While it would be overkill to call this book *WPF and Silverlight Unleashed*, it should be comforting to know that the knowledge you gain from this book can help you be an expert in both WPF and Silverlight.

Summary

As time passes, more software is delivering high-quality—sometimes *cinematic*—experiences, and software that doesn't risk looking old-fashioned. However, the effort involved in creating such user interfaces—especially ones that exploit Windows—has been far too difficult in the past.

WPF makes it easier than ever before to create all kinds of user interfaces, whether you want to create a traditional-looking Windows application or an immersive 3D experience worthy of a role in a summer blockbuster. Such a rich user interface can be evolved fairly independently from the rest of an application, allowing graphic designers to participate in the software development process much more effectively. But don't just take my word for it; read on to see for yourself how it's done!

Symbols/Numbers

\ (backslash), 34

{ } (curly braces), 33-34, 377

2D graphics

2D and 3D coordinate system transformation, 541, 590-591

explained, 596

Visual.TransformToAncestor method, 596-600

Visual3D.TransformToAncestor method, 600-605

Visual3D.TransformToDescendant method, 600-605

Brushes

BitmapCacheBrush class, 535

DrawingBrush class, 520-524

explained, 513

ImageBrush class, 524-525

LinearGradientBrush class, 515-518

as opacity masks, 527-529

RadialGradientBrush class, 519-520

SolidColorBrush class, 514

VisualBrush class, 525-527

drawings

clip art example, 491-492

Drawing class, 476

DrawingBrush class, 477

DrawingContext methods, 494

DrawingImage class, 477-479

DrawingVisual class, 477

GeometryDrawing class, 476-477

GlyphRunDrawing class, 476

ImageDrawing class, 476-478

Pen class, 489-491

VideoDrawing class, 476

Effects, 529-531

explained, 475-476

geometries

aggregate geometries, 483

Bézier curves, 480

CombinedGeometry class, 486-487

defined, 479

EllipseGeometry class, 479

GeometryGroup class, 484-486

LineGeometry class, 479

PathGeometry class, 479-483

RectangleGeometry class, 479

representing as strings, 487-489

StreamGeometry class, 483

house example, 538

Shapes

clip art based on Shapes, 512-513

Ellipse class, 508

explained, 505-506

how they work, 509

Line class, 509-510

overuse of, 507

Path class, 511-512

Polygon class, 511

Polyline class, 510

Rectangle class, 507-508

transforms. *See* transforms

Visuals

custom rendering, 499

displaying on screen, 496-498

DrawingContext methods, 494

DrawingVisuals, 493-496

explained, 493

visual hit testing, 499-505

WPF 3.5 enhancements, 15

3D graphics

2D and 3D coordinate system
transformation, 541, 590-591

explained, 596

Visual.TransformToAncestor method,
596-600

Visual3D.TransformToAncestor method,
600-605

Visual3D.TransformToDescendant
method, 600-605

3D hit testing, 592-593

Cameras

blind spots, 545

coordinate systems, 542-544

explained, 542

LookDirection property, 544-548

MatrixCamera, 553

OrthographicCamera versus
PerspectiveCamera, 551-553

Position property, 543-544

Transform property, 549

UpDirection property, 548-550

Z-fighting, 545

coordinate systems, 542-544

explained, 537-538

hardware acceleration

explained, 12

GDI and, 13

house example, 538-540

Lights, 542

Materials

AmbientMaterial, 575

combining, 578

DiffuseMaterial, 572-575

EmissiveMaterial, 576-578

explained, 571

- Model3Ds
 - explained, 563
 - GeometryModel3D, 571
 - Lights, 563-570
 - Model3DGroup class, 584-586
 - pixel boundaries, 17
 - resolution independence, 12
 - texture coordinates, 584
 - Transform3Ds
 - combining, 562
 - explained, 554-555
 - house drawing example, 555-556
 - MatrixTransform3D class, 554, 562
 - RotateTransform3D class, 554, 559-562
 - ScaleTransform3D class, 554, 557-559
 - Transform3DGroup class, 554
 - TranslateTransform3D class, 554-557
 - Viewport2DVisual3D class, 590-591
 - Viewport3D class, 593-596
 - Visual3Ds
 - explained, 586
 - ModelVisual3D class, 587-588
 - UIElement3D class, 588-590
 - WPF 3.5 enhancements, 15
 - 3D hit testing, 592-593**
- A**
- About dialog**
 - attached events, 165-167
 - with font properties moved to inner StackPanel, 90
 - with font properties set on root window, 85-86
 - Help command, 191-192
 - initial code listing, 75-76
 - routed events, 162-164
 - absolute sizing, 130**
 - accessing**
 - binary resources
 - embedded in another assembly, 348
 - from procedural code, 349-350
 - at site of origin, 348-349
 - from XAML, 345-348
 - logical resources, 360
 - Action property (QueryContinueDragEventArgs class), 173**
 - ActiveEditMode property (InkCanvas class), 317**
 - ActiveX controls, 714-718**
 - ActualHeight property (FrameworkElement class), 100**
 - ActualWidth property (FrameworkElement class), 100**
 - AddBackEntry method, 218**
 - AddHandler method, 160-161**
 - advantages of WPF, 13**
 - Aero Glass, 249-253**
 - aggregate geometries, 483**
 - AlternationCount property (ItemsControl class), 276**
 - AlternationIndex property (ItemsControl class), 276**
 - AmbientLight, 564, 569-570**
 - AmbientMaterial class, 575**
 - AnchoredBlock class, 326-327**
 - AND relationships (logical), 429-430**
 - Angle property (RotateTransform class), 108**
 - AngleX property (SkewTransform class), 112**
 - AngleY property (SkewTransform class), 112**

animation

animation classes

- AutoReverse property, 618
- BeginTime property, 616-617
- By property, 616
- DoubleAnimation, 611-612
- Duration property, 614
- EasingFunction property, 620
- explained, 609-610
- FillBehavior property, 621
- From property, 614-616
- IsAdditive property, 621
- IsCumulative property, 621
- lack of generics, 610-611
- linear interpolation, 612-613
- RepeatBehavior property, 618-619
- SpeedRatio property, 617
- To property, 614-616

data binding and, 632

easing functions, 16

- BackEase, 640
- BounceEase, 640
- CircleEase, 640
- EasingMode property, 637
- ElasticEase, 640
- ExponentialEase, 640
- power easing functions, 637-638
- SineEase, 640
- writing, 640-642

explained, 89, 607

frame-based animation, 609

keyframe animation

- discrete keyframes, 634-636
- easing keyframes, 636
- explained, 630
- linear keyframes, 631-633
- spline keyframes, 633-634

path-based animations, 637

reusing animations, 613

timer-based animation, 608-609

and Visual State Manager

- Button ControlTemplate with VisualStates, 643-646

transitions, 647-651

with XAML EventTriggers/Storyboards

explained, 621-622

starting animations from property triggers, 628-629

Storyboards as Timelines, 629-630

TargetName property, 625-626

TargetProperty property, 622-625

annotations, adding to flow documents, 331-334**AnnotationService class, 331****Application class**

- creating applications without, 204
- events, 202
- explained, 199-200
- Properties collection, 203
- Run method, 200-201
- single-instance applications, 204
- Windows collection, 202

ApplicationCommands class, 189**ApplicationPath property (JumpTask), 238****applications**

- associating Jump Lists with, 234
- embedding Win32 controls in WPF applications
 - explained, 677
 - keyboard navigation, 687-691
 - Webcam control, 678-687
- embedding Windows Forms controls in WPF applications
 - explained, 699-700
 - PropertyGrid, 700-703

- embedding WPF controls in Win32 applications
 - HwndSource class, 692-695
 - layout, 696-699
- embedding WPF controls in Windows Forms applications
 - converting between two representatives, 707-708
 - ElementHost class, 704-706
 - launching modal dialogs, 708
- gadget-style applications, 223-224
- loose XAML pages, 231-232
- multiple-document interface (MDI), 203
- navigation-based Windows applications
 - explained, 211-212
 - hyperlinks, 215-216
 - journal, 216-218
 - Navigate method, 214-215
 - navigation containers, 212-214
 - navigation events, 218-219
 - Page elements, 212-214
 - returning data from pages, 221-222
 - sending data to pages, 220-221
- standard Windows applications
 - Application class, 199-204
 - application state, 209-210
 - ClickOnce, 210-211
 - common dialogs, 206-207
 - custom dialogs, 207-208
 - explained, 195-196
 - multithreaded applications, 205
 - retrieving command-line arguments in, 202
 - single-instance applications, 204
 - splash screens, 205-206
 - Window class, 196-198
 - Windows Installer, 210
 - XAML Browser applications (XBAPs)
 - ClickOnce caching, 226
 - deployment, 229
 - explained, 224-226
 - full-trust XAML Browser applications, 228
 - integrated navigation, 228-229
 - limitations, 226-227
 - on-demand download, 230-231
 - security, 229
- Apply method, 245**
- arbitrary objects, content and, 263**
- ArcSegment class, 480**
- Arguments property (JumpTask), 238**
- ArrangeOverride method, overriding, 754-755**
- associating Jump Lists with applications, 234**
- asynchronous data binding, 401**
- attached events, 165-167**
- attached properties**
 - About dialog example, 90-91
 - as extensibility mechanism, 92-93
 - attached property providers, 92
 - defined, 89
- attached property providers, 92**
- attenuation, 566**
- attributes, setting, 25**
- audio support**
 - embedded resources, 663
 - explained, 653
 - MediaElement, 656-658
 - MediaPlayer, 655-656
 - MediaTimeline, 656-658
 - SoundPlayer, 654
 - SoundPlayerAction class, 654-655

- speech recognition
 - converting spoken words into text, 667-670
 - specifying grammar with GrammarBuilder, 671-672
 - specifying grammar with SRGS, 670-671
- speech synthesis
 - explained, 664
 - GetInstalledVoices method, 664
 - PromptBuilder class, 665-667
 - SelectVoice method, 664
 - SelectVoiceByHints method, 664
 - SetOutputToWaveFile method, 665
 - SpeakAsync method, 664
 - Speech Synthesis Markup Language (SSML), 665-667
 - SpeechSynthesizer, 664
 - SystemSounds class, 654

“Auto” length, 99**automation**

- automation IDs, 289
- UI Automation, supporting in custom controls, 749-750

AutoReverse property (animation classes), 618**autosizing, 128-130****AxisAngleRotation3D class, 559-560****AxMsTscAxNotSafeForScripting control, 716-717****B****BackEase function, 640****backslash (\), 34****BAML (Binary Application Markup Language)**

- decompiling back into XAML, 47-48
- defined, 45

Baml2006Reader class, 53**base values of dependency properties, calculating, 87-88****BaseValueSource enumeration, 88****BeginTime property (animation classes), 616-617****behavior**

- adding to custom controls
 - behavior, 737-739
 - code-behind file, 734
 - initial implementation, 733-737
 - resources, 734-735
- creating for user controls, 725-727

Bézier curves, 480**BezierSegment class, 480****Binary Application Markup Language (BAML)**

- decompiling back into XAML, 47-48
- defined, 45

binary resources

- accessing
 - embedded in another assembly, 348
 - from procedural code, 349-350
 - at site of origin, 348-349
 - from XAML, 345-348
- defining, 344-345
- explained, 343
- localizing
 - creating satellite assembly with LocBaml, 351
 - explained, 350
 - marking user interfaces with localization IDs, 351
 - preparing projects for multiple cultures, 350

Binding object. See also data binding

- binding
 - to .NET properties, 367-368
 - to collections, 370-373

- to entire objects, 369-370
- to UIElement, 370
- DoNothing values, 385
- ElementName property, 366
- IsAsync property, 401
- in procedural code, 363-365
- RelativeSource property, 367
- removing, 365
- sharing source with DataContext, 374-375
- StringFormat property, 375-376
- TargetNullValue property, 366
- UpdateSourceExceptionFilter property, 408
- UpdateSourceTrigger property, 404
- validation rules, 405-409
- ValidationRules property, 406
- in XAML, 365-367
- BindingMode enumeration, 403**
- BitmapCache class, 533-535**
- BitmapCacheBrush class, 535**
- BitmapEffect, 530**
- bitmaps**
 - nearest-neighbor bitmap scaling, 310
 - WriteableBitmap class, 15
- BitmapScalingMode property (RenderOptions), 306**
- BlackoutDates property (Calendar control), 337-338**
- blind spots (Cameras), 545**
- Block TextElements**
 - AnchoredBlock class, 326-327
 - BlockUIContainer, 321
 - List, 320
 - Paragraph, 320
 - sample code listing, 321-324
 - Section, 320
 - Table, 320

- BlockUIContainer Blocks, 321**
- BlurEffect, 529-530**
- BooleanToVisibilityConverter, 383-384**
- Bottom property (Canvas), 116**
- BounceEase function, 640**
- BrushConverter type converter, 32**
- Brushes**
 - applying without logical resources, 352-353
 - BitmapCacheBrush class, 535
 - consolidating with logical resources, 353-355
 - explained, 513
 - ImageBrush class, 524-525
 - LinearGradientBrush class, 515-518
 - as opacity masks, 527-529
 - RadialGradientBrush class, 519-524
 - SolidColorBrush class, 514
 - VisualBrush class, 525-527
- bubbling, 161**
- BuildWindowCore class, 684**
- built-in commands, 189-192**
- Button class, 81, 264-265**
- ButtonAutomationPeer class, 265**
- ButtonBase class, 263-264**
- buttons**
 - Button class, 81, 264-265
 - Button ControlTemplate with VisualStates, 643-646
 - ButtonBase class, 263-264
 - CheckBox class, 266
 - defined, 263
 - RadioButton class, 266-268
 - RepeatButton class, 265
 - styling with built-in animations, 626-628
 - ToggleButton class, 265-266
- By property (animation classes), 616**

C**C++/CLI, 681-682****cached composition**

- BitmapCache class, 533-535
- BitmapCacheBrush class, 535
- Viewport2DVisual3D support for, 591

caching, ClickOnce, 226**Calendar control, 336-338****calendar controls**

- Calendar, 336-338
- DatePicker, 338-339

Cameras

- blind spots, 545
- coordinate systems, 542-544
- explained, 542
- LookDirection property, 544-548
- MatrixCamera, 553
- OrthographicCamera versus PerspectiveCamera, 551-553
- Position property, 543-544
- Transform property, 549
- UpDirection property, 548-550
- Z-fighting, 545

CAML (Compiled Application Markup Language), 46**CanUserDeleteRows property (DataGrid), 298****cancel buttons, 264****Cancel method, 185****CanExecute method, 189****CanExecuteChanged method, 189****CanUserAddRows property (DataGrid), 298****Canvas, 116-118. See also SimpleCanvas**

- mimicking with Grid, 136

capturing mouse events, 173-174**cells (DataGrid), selecting, 295****Center property (RadialGradientBrush), 519****CenterX property**

- RotateTransform class, 108-110
- SkewTransform class, 112

CenterY property

- RotateTransform class, 108-110
- SkewTransform class, 112

change notification (dependency properties), 83-84**CheckBox class, 266****child object elements**

- content property, 35-36
- dictionaries, 37-38
- lists, 36-37
- processing rules, 40
- values type-converted to object elements, 38

/clr compiler option, 686**CircleEase function, 640****class hierarchy, 73-75****Class keyword, 44****classes. See specific classes****ClearAllBindings method, 365****ClearBinding method, 365****ClearHighlightsCommand, 331****clearing**

- bindings, 365
- local values, 88

ClearValue method, 88**CLI (Common Language Infrastructure), 681****Click event, 263-264****clickable cube example, 588-590****ClickCount property (MouseButtonEventArgs), 172****ClickMode property (ButtonBase class), 263****ClickOnce, 210-211**

- ClickOnce caching, 226
- with unmanaged code, 211

clients, pure-XAML Twitter client, 412-413

clip art example, 491-492

- clip art based on Shapes, 512-513
- drawing-based implementation, 491-492
- DrawingContext-based implementation, 495-496
- WindowHostingVisual.cs file, 497

clipboard interaction (DataGrid), 296

ClipboardCopyMode property (DataGrid), 296

clipping, 139-141

ClipToBounds property (panels), 140

clr-namespace directive, 39

code-behind files, 44, 734

CoerceValueCallback delegate, 89

cold start time, 205

Collapsed value (Visibility enumeration), 102

collections

- binding to, 370-373
- customizing collection views
 - creating new views, 394-396
 - explained, 386
 - filtering, 392
 - grouping, 388-391
 - navigating, 392-393
 - sorting, 386-388
- dictionaries, 37-38
- ItemsSource, 297
- lists, 36-37
- Properties, 203
- SortDescriptions, 387
- Triggers, 85
- Windows, 202

CollectionViewSource class, 394

color brushes

- applying without logical resources, 352-353
- consolidating with logical resources, 353-355

LinearGradientBrush class, 515-518

RadialGradientBrush class, 519-520

SolidColorBrush class, 514

color space profiles, 515

Color structure, 514

columns (Grid)

- auto-generated columns, 294-295
- column types, 293-294
- freezing, 297
- sharing row/column sizes, 134-136
- sizing
 - absolute sizing, 130
 - autosizing, 130
 - GridLength structures, 131-132
 - interactive sizing with GridSplitter, 132-133
 - percentage sizing, 131
 - proportional sizing, 130

CombinedGeometry class, 486-487

combining

- Materials, 578
- Transform3Ds, 562
- transforms, 113-114

ComboBox control

- ComboBoxItem objects, 286-287
- customizing selection box, 282-285
- events, 282
- explained, 282
- IsEditable property, 282
- IsReadOnly property, 282
- SelectionChanged event, 285-286

ComboBoxItem objects, 286-287

ComCtl32.dll, 255-256

command-line arguments, retrieving, 202

commands. See also specific commands

- built-in commands, 189-192
- controls with built-in command bindings, 193-194
- executing with input gestures, 192-193
- explained, 188-189
- implementing with custom controls, 745

commas in geometry strings, 489**common dialogs, 206-207****Common Language Infrastructure (CLI), 681****Compiled Application Markup Language (CAML), 46****compiling XAML, 43-45****Complete method, 185****CompleteQuadraticEase class, 642****ComponentCommands class, 190****CompositeCollection class, 410****CompositionTarget_Rendering event handler, 713****conflicting triggers, 429****consolidating routed event handlers, 167-168****ConstantAttenuation property (PointLights), 566****containers**

- Expander class, 273-274
- Frame class, 271-272
- GroupBox class, 273
- Label class, 268
- navigation containers, 212-214
- ToolTip class, 269-271

ContainerUIElement3D class, 590**Content build action, 344****content controls**

- and arbitrary objects, 263
- buttons
 - Button class, 264-265
 - ButtonBase class, 263-264
 - CheckBox class, 266

defined, 263

RadioButton class, 266-268

RepeatButton class, 265

ToggleButton class, 265-266

containers

Expander class, 273-274

Frame class, 271-272

GroupBox class, 273

Label class, 268

ToolTip class, 269-271

ContentControl class, 262

defined, 262

content overflow, handling

clipping, 139-141

explained, 139

scaling, 143-147

scrolling, 141-143

Content property, 35-36

ContentControl class, 435-437

Frame class, 272

ContentControl class, 262, 435-437**ContentElement class, 74****ContextMenu control, 301-302****ContextMenuService class, 302****Control class, 75****control parts, 744-745****control states, 745-749****control templates**

ControlTemplate with triggers, 432-434

editing, 457-458

explained, 430-431

mixing with styles, 456-457

named elements, 434

reusability of, 438-440

simple control template, 431-432

target type, restricting, 434-435

- TargetType property, 434-435
- templated parent properties, respecting
 - Content property (ContentControl class), 435-437
 - hijacking existing properties for new purposes, 441
 - other properties, 438-440
- triggers, 432-434
- visual states
 - respecting with triggers, 442-446
 - respecting with VSM (Visual State Manager), 447-455
- controls**
 - ActiveX controls, 714-718
 - buttons
 - Button class, 264-265
 - ButtonBase class, 263-264
 - CheckBox class, 266
 - defined, 263
 - RadioButton class, 266-268
 - RepeatButton class, 265
 - ToggleButton class, 265-266
 - Calendar, 336-338
 - ComboBox
 - ComboBoxItem objects, 286-287
 - customizing selection box, 282-285
 - events, 282
 - explained, 282
 - IsEditable property, 282
 - IsReadOnly property, 282
 - SelectionChanged event, 285-286
 - containers
 - Expander class, 273-274
 - Frame class, 271-272
 - GroupBox class, 273
 - Label class, 268
 - ToolTip class, 269-271
 - ContextMenu, 301-302
 - control parts, 447-449
 - control states, 449-455
 - controls with built-in command bindings, 193-194
 - custom controls, creating
 - behavior, 733-739
 - code-behind file, 734
 - commands, 745
 - control parts, 744-745
 - control states, 745-749
 - explained, 12, 721-722
 - generic resources, 741-742
 - resources, 734-735
 - UI Automation, 749-750
 - user controls versus custom controls, 722
 - user interfaces, 739-740, 742
 - DataGrid, 293
 - auto-generated columns, 294-295
 - CanUserAddRows property, 298
 - CanUserDeleteRows property, 298
 - clipboard interaction, 296
 - ClipboardCopyMode property, 296
 - column types, 293-294
 - displaying row details, 296-297
 - editing data, 297-298
 - EnableColumnVirtualization property, 296
 - EnableRowVirtualization property, 296
 - example, 292-293
 - freezing columns, 297
 - FrozenColumnCount property, 297
 - RowDetailsVisibilityMode property, 297
 - selecting rows/cells, 295
 - SelectionMode property, 295

- SelectionUnit property, 295
 - virtualization, 296
- DatePicker, 338-339
- explained, 261-263
- GridView, 290-291
- InkCanvas, 316-318
- ItemsControl class, 275-276
 - AlternationCount property, 276
 - AlternationIndex property, 276
 - DisplayMemberPath property, 276-277
 - HasItems property, 276
 - IsGrouping property, 276
 - IsTextSearchCaseSensitive property, 285
 - IsTextSearchEnabled property, 285
 - Items property, 275
 - ItemsPanel property, 276-280
 - ItemsSource property, 276
 - scrolling behavior, controlling, 280-281
- ListBox
 - automation IDs, 289
 - example, 287-288
 - scrolling, 289
 - SelectionMode property, 288
 - sorting items in, 289
 - support for multiple selections, 288
- ListView, 290-291
- Menu, 298-301
- PasswordBox, 316
- ProgressBar, 335
- RichTextBox, 316
- ScrollViewer, 141-143
- Selector class, 281
- Slider, 335-336
- states, 745-749
- StatusBar, 307-308
- TabControl, 291-292
- TextBlock
 - explained, 313
 - explicit versus implicit runs, 314
 - properties, 313
 - support for multiple lines of text, 315
 - whitespace, 314
- TextBox, 315
- ToolBar, 304-306
- TreeView, 302-304
- user controls, creating
 - behavior, 725-727
 - dependency properties, 728-731
 - explained, 721-722
 - protecting controls from accidental usage, 727-728
 - routed events, 731-732
 - user controls versus custom controls, 722
 - user interfaces, 723-725
- ControlTemplate class. See control templates**
- Convert method, 382**
- converting spoken words into text, 667-670**
- ConvertXmlStringToObjectGraph method, 65**
- coordinate systems, 542-544**
- CountToBackgroundConverter class, 382-384**
- CreateBitmapSourceFromHBitmap method, 708**
- CreateHighlightCommand, 331**
- CreateInkStickyNoteCommand, 331**
- CreateTextStickyNoteCommand, 331**
- CreateWindow method, 685**
- Cube example**
 - clickable cube, 588-590
 - cube and TextBlocks, 600-604
 - cube button style, 594-595
 - cube of buttons and small purple cube, 597-599
 - initial code listing, 585-586

cultures, preparing projects for multiple cultures, 350

curly braces ({}), 33-34

CurrentItem property (ICollectionView), 392

curves, Bézier, 480

CustomCategory property (JumpTask), 239-240

customization

advantages/disadvantages, 416

collection views

creating new views, 394-396

explained, 386

filtering, 392

grouping, 388-391

navigating, 392-393

sorting, 386-388

color space profiles, 515

custom controls, creating

behavior, 733-739

commands, 745

control parts, 744-745

control states, 745-749

explained, 721-722

generic resources, 741-742

UI Automation, 749-750

user controls versus custom controls, 722

user interfaces, 739-742

custom rendering, 499

custom sorting, 388

data display, 385

data flow, 403-405

dialogs, 207-208

JumpTask behavior, 237-240

keyboard navigation, 306

panels

communication between parents and children, 752-755

explained, 751-752

selection boxes (ComboBox control), 282-285

taskbar

explained, 245-246

taskbar item progress bars, 246

taskbar overlays, 247

thumb buttons, 248-249

thumbnail content, 247

D

D3DImage class, 708-714

DashStyle class, 490-491

DashStyle property (Pen class), 490

data binding, 15

animation and, 632

asynchronous data binding, 401

Binding object, 363

binding to .NET properties, 367-368

binding to collections, 370-373

binding to entire objects, 369-370

binding to UIElement, 370

ElementName property, 366

IsAsync property, 401

in procedural code, 363-365

RelativeSource property, 367

removing, 365

sharing source with DataContext, 374-375

StringFormat property, 375-376

TargetNullValue property, 366

UpdateSourceExceptionFilter property, 408

UpdateSourceTrigger property, 404

validation rules, 405-409

- ValidationRules property, 406
 - in XAML, 365-367
- canceling temporarily, 385
- CompositeCollection class, 410
- controlling rendering
 - data templates, 378-380
 - explained, 375
 - string formatting, 375-377
 - value converters, 381-386
- customizing collection views
 - creating new views, 394-396
 - explained, 386
 - filtering, 392
 - grouping, 388-391
 - navigating, 392-393
 - sorting, 386-388
- customizing data flow, 403-405
- data providers
 - explained, 396
 - ObjectDataProvider class, 401-403
 - XmlDataProvider class, 397-401
- defined, 363
- Language Integrated Query (LINQ), 396
- to methods, 402-403
- MultiBinding class, 410-411
- PriorityBinding class, 411
- pure-XAML Twitter client, 412-413
- troubleshooting, 384
- data flow, customizing, 403-405**
- Data property (DragEventArgs class), 172**
- data providers**
 - explained, 396
 - ObjectDataProvider class, 401-403
 - XmlDataProvider class, 397-401
- data templates, 378-380**
 - HierarchicalDataTemplate, 399-400
 - template selectors, 381

data triggers, 84, 427-428**data types**

- bridging incompatible data types, 381-384
- in XAML 2009, 50

DataContext property, 374-375**DataGrid control**

- CanUserAddRows property, 298
- CanUserDeleteRows property, 298
- clipboard interaction, 296
- ClipboardCopyMode property, 296
- column types, 293-295
- displaying row details, 296-297
- editing data, 297-298
- EnableColumnVirtualization property, 296
- EnableRowVirtualization property, 296
- example, 292-293
- freezing columns, 297
- FrozenColumnCount property, 297
- RowDetailsVisibilityMode property, 297
- selecting rows/cells, 295
- SelectionMode property, 295
- SelectionUnit property, 295
- virtualization, 296

DataGridCheckBoxColumn, 294**DataGridComboBoxColumn, 294****DataGridHyperlinkColumn, 293****DataGridTemplateColumn, 294****DataGridTextColumn, 293****DataTrigger class, 427-428****DatePicker control, 338-339****DateValidationError event, 339****DayOfWeek enumeration, 338****DeadCharProcessedKey property (KeyEventArgs class), 168****debugger (Visual C++), 695****declaration context, 375****declarative programming, 12**

decorators, 144

default buttons, 264

default styles, 88

defining

binary resources, 344-345

object elements, 25

properties, 53

delegates

CoerceValueCallback, 89

delegate contravariance, 168

ValidateValueCallback, 89

DeleteStickyNotesCommand, 331

dependency properties, 419-420

adding to user controls, 728-731

attached properties

About dialog example, 90-91

attached property providers, 92

defined, 89

as extensibility mechanism, 92-93

attached property providers, 92

change notification, 83-84

explained, 80-81

hijacking, 441

implementation, 81-83

property triggers, 83-85

property value inheritance, 85-86

support for multiple providers

applying animations, 89

coercion, 89

determining base values, 87-88

evaluating, 89

explained, 87

validation, 89

DependencyObject class, 74, 82

DependencyPropertyHelper class, 88

deployment

ClickOnce, 210-211

Windows Installer, 210

WPF 3.5 enhancements, 16

WPF 4 enhancements, 17

XAML Browser applications, 229

DesiredSize property (FrameworkElement class), 99

DestroyWindowCore class, 684

device-independent pixels, 102

DialogFunction method, 694

dialogs

About dialog

with font properties moved to inner StackPanel, 90

with font properties set on root window, 85-86

initial code listing, 75-76

common dialogs, 206-207

custom dialogs, 207-208

dialog results, 208

modal dialogs

launching from Win32 applications, 699

launching from Windows Forms applications, 708

launching from WPF applications, 692, 703

modeless dialogs, 196

TaskDialogs, 253-256

dictionaries, 37-38, 50

DiffuseMaterial, 572-575

direct routing, 161

Direct3D, 12

Direction property

DirectionalLight, 564

PointLights, 568

DirectionalLight, 564-565

directives. See specific directives

DirectX

- development of, 10-11
- versus WPF, 13-14
- when to use, 13-14
- WPF interoperability, 15, 708-714

discrete keyframes, 634-636**DispatcherObject class, 74****DispatcherPriority enumeration, 205****DispatcherTimer class, 608-609****DisplayDateEnd property (Calendar control), 337****DisplayDateStart property (Calendar control), 337****displaying**

- flow documents, 329-331
- Visuals on screen, 496-498

DisplayMemberPath property, 276-277, 371**Dock property (DockPanel), 122****DockPanel**

- examples, 122-125
- explained, 122
- interaction with child layout properties, 125
- mimicking with Grid, 136
- properties, 122

documents, flow

- annotations, 331-334

Blocks

- AnchoredBlock class, 326-327
- BlockUIContainer, 321
- List, 320
- Paragraph, 320
- sample code listing, 321-324
- Section, 320
- Table, 320

creating, 318-319

defined, 318

displaying, 329-331

Inlines

- AnchoredBlock, 326-327
- defined, 324-325
- InlineUIContainer, 329
- LineBreak, 327
- Span, 325-326

DoNothing value (Binding), 385**DoubleAnimation class, 611-612****download groups, 230****DownloadFileGroupAsync method, 231****drag-and-drop events, 172-173****DragEventArgs class, 172****Drawing class, 476****DrawingBrush class, 477, 520-524****DrawingContext class**

- clip art example, 495-496
- methods, 494

DrawingImage class, 477-479**drawings**

- clip art example, 491-492
- Drawing class, 476
- DrawingBrush class, 477
- DrawingContext methods, 494
- DrawingImage class, 477-479
- DrawingVisual class, 477
- geometries. *See* geometries
- GeometryDrawing class, 476-477
- GlyphRunDrawing class, 476
- ImageDrawing class, 476-478
- Pen class, 489-491
- VideoDrawing class, 476
- WPF 3.5 enhancements, 15

DrawingVisuals

- explained, 477, 493
- filling with content, 493-496

DropDownOpened event, 282**DropShadowEffect, 529-530**

duration of animations, controlling, 614
Duration property (animation classes), 614
DwmExtendFrameIntoClientArea method, 249-252
 dynamic versus static resources, 355-357
DynamicResource markup extension, 356-357

E

Ease method, 640
EaseIn method, 642-643
EaseInOut method, 642-643
 easing functions, 16
 easing keyframes, 636
EasingFunction property (animation classes), 620
EasingFunctionBase class, 641
EasingMode property (easing functions), 637
 editing

- control templates, 457-458
- DataGrid** data, 297-298

EditingCommands class, 190
EditingMode property (**InkCanvas** class), 317
EditingModeInverted property (**InkCanvas** class), 317
Effects, 529-531
ElasticEase function, 640
 element trees. *See* trees
ElementHost class, 704-706
ElementName property (**Binding** object), 366
 elements. *See* object elements; property elements
EllipseGeometry class, 479
 embedded resources, 663
EmbeddedResource build action, 345

embedding
 ActiveX controls in WPF applications, 714-718
 Win32 controls in WPF applications

- explained, 677
- keyboard navigation, 687-691
- Webcam control, 678-687

 Windows Forms controls in WPF applications

- explained, 699-700
- PropertyGrid**, 700-703

 WPF controls in Win32 applications

- HwndSource** class, 692-695
- layout, 696-699

 WPF controls in Windows Forms applications

- converting between two representatives, 707-708
- ElementHost** class, 704-706
- launching modal dialogs, 708

EmissiveMaterial class, 576-578
EnableClearType property (**BitmapCache** class), 534
EnableColumnVirtualization property (**DataGrid**), 296
EnableRowVirtualization property (**DataGrid**), 296
EnableVisualStyles method, 703
EndLineCap property (**Pen** class), 489
EndMember value (**NodeType** property), 57
EndObject value (**NodeType** property), 57
EndPoint property (**LinearGradientBrush**), 516

enumerations
 BaseValueSource, 88
 BindingMode, 403
 DayOfWeek, 338
 DispatcherPriority, 205
 GeometryCombineMode, 486

- GradientSpreadMethod, 517
- JumpltemRejectionReason, 244
- Key, 168-169
- MouseButtonState, 171
- PixelFormats, 311
- RoutingStrategy, 161
- ShutdownMode, 202
- Stretch, 144
- StretchDirection, 144
- TileMode, 523
- UpdateSourceTrigger, 404-405
- Visibility, 102-103
- error handling, 407-409**
- Error ProgressState, 246**
- EscapePressed property**
(QueryContinueDragEventArgs class), 173
- Euler angles, 560**
- EvenOdd fill (FillRule property), 482**
- event handlers, 52**
- event wrappers, 160**
- events**
 - attributes, 25
 - Click, 263-264
 - DateValidationError, 339
 - DropDownOpened, 282
 - event wrappers, 160
 - JumpltemsRejected, 244
 - JumpltemsRemovedByUser, 244
 - keyboard events, 168-170
 - mouse events
 - capturing, 173-174
 - drag-and-drop events, 172-173
 - explained, 170-171
 - MouseButtonEventArgs, 171
 - MouseEventArgs, 171-172
 - MouseEventArgs, 171
 - transparent and null regions, 171
 - multi-touch events
 - basic touch events, 177-180
 - explained, 176
 - manipulation events, 180-188
 - navigation events, 218-219
 - order of processing, 26
 - Rendering, 609
 - routed events
 - About dialog example, 162-164
 - adding to user controls, 731-732
 - attached events, 165-167
 - consolidating routed event handlers, 167-168
 - defined, 159
 - explained, 159-160
 - implementation, 160-161
 - RoutedEventArgs class, 162
 - routing strategies, 161-162
 - stopping, 165
 - SelectedDatesChanged, 339
 - SelectionChanged, 281, 285-286
 - stylus events, 174-176
- EventTriggers, 84, 621-622**
- ExceptionValidationRule object, 407**
- Execute method, 189**
- executing commands with input gestures, 192-193**
- Expander class, 273-274**
- Expansion property (ManipulationDelta class), 181**
- explicit sizes, avoiding, 99**
- explicit versus implicit runs, 314**
- ExponentialEase function, 640**
- Expression Blend, 14**
- expressions, 89**

ExtendGlassFrame method, 252
 extensibility mechanisms, attached properties as, 92-93
 extensibility of XAML, 39
Extensible Application Markup Language.
 See XAML

F

factoring XAML, 357
FanCanvas, 768-772
FileInputBox control
 behavior, 725-727
 dependency properties, 728-731
 protecting from accidental usage, 727-728
 routed events, 731-732
 user interface, 723-725
files. See *also specific files*
 code-behind files, 44
 MainWindow.xaml.cs, 178-179, 186-187
 raw project files, opening in Visual Studio, 350
 VisualStudioLikePanels.xaml, 151-153
 VisualStudioLikePanels.xaml.cs, 153-157
FillBehavior property (animation classes), 621
FillRule property (PathGeometry class), 482-483
Filter property (ICollectionView), 392
 filtering, 392
 finding type converters, 32
FindResource method, 359
FirstDayOfWeek property (Calendar control), 338
Flat line cap (Pen), 490

flow documents

annotations, 331-334
 Blocks
 AnchoredBlock class, 326-327
 BlockUIContainer, 321
 List, 320
 Paragraph, 320
 sample code listing, 321-324
 Section, 320
 Table, 320
 creating, 318-319
 defined, 318
 displaying, 329-331
 Inlines
 AnchoredBlock, 326-327
 defined, 324-325
 InlineUIContainer, 329
 LineBreak, 327
 Span, 325-326
FlowDirection property (FrameworkElement class), 105-106
FlowDocument element, 318
FlowDocumentPageViewer control, 329
FlowDocumentReader control, 329-333
FlowDocumentScrollViewer control, 329
FontSizeConverter type converter, 32
Form1.cs file, 704, 707
FormatConvertedBitmap class, 310
 formatting strings, 375-377
Frame class, 212-214, 271-272
 frame-based animation, 609
FrameworkContentElement class, 75, 80, 318
FrameworkElement class
 ActualHeight property, 100
 ActualWidth property, 100

- DesiredSize property, 99
- explained, 75, 80
- FlowDirection property, 105-106
- Height property, 98-100
- HorizontalAlignment property, 103-104
- HorizontalContentAlignment property, 104-106
- LayoutTransform property, 106
- Margin property, 100-102
- Padding property, 100-102
- RenderSize property, 99
- RenderTransform property, 106
- Triggers property, 85
- VerticalAlignment property, 103-105
- Visibility property, 102-103
- Width property, 98-100

FrameworkPropertyMetadata, 731

Freezable class, 74

freezing columns, 297

From property (animation classes), 614-616

FromArgb method, 707

FrozenColumnCount property (DataGrid), 297

full-trust XAML Browser applications, 228

functions. See *specific functions*

G

gadget-style applications, 223-224

GDI (graphics device interface), 10

- GDI+, 10

- hardware acceleration and, 13

generated source code, 46

generic dictionaries, 467, 741-742

generics support (XAML2009), 49

geometries

- aggregate geometries, 483

- Bézier curves, 480

- CombinedGeometry class, 486-487

- defined, 479

- EllipseGeometry class, 479

- Geometry3D class, 578

- GeometryGroup class, 484-486

- LineGeometry class, 479

- MeshGeometry3D class, 578-579

- Normals property, 581-583

- Positions property, 579

- TextureCoordinates property, 583

- TriangleIndices property, 580-581

- PathGeometry class

- ArcSegment, 480

- BezierSegment, 480

- example, 480-482

- explained, 479

- FillRule property, 482-483

- LineSegment, 480

- PolyBezierSegment, 480

- PolyLineSegment, 480

- PolyQuadraticBezierSegment, 480

- QuadraticBezierSegment, 480

- RectangleGeometry class, 479

- representing as strings, 487-489

- StreamGeometry class, 483

Geometry3D class, 578

GeometryCombineMode enumeration, 486

GeometryDrawing class, 476-477

GeometryGroup class, 484-486

GeometryModel3D

- defined, 563

- explained, 571

- Geometry3D class, 578

- Materials
 - AmbientMaterial, 575
 - combining, 578
 - DiffuseMaterial, 572-575
 - EmissiveMaterial, 576-578
 - explained, 571
- MeshGeometry3D class, 578-579
 - Normals property, 581-583
 - Positions property, 579
 - TextureCoordinates property, 583
 - TriangleIndices property, 580-581
- GetCommandLineArgs method, 202**
- GetExceptionForHR method, 51**
- GetGeometry method, 479**
- GetHbitmap function, 708**
- GetInstalledVoices method, 664**
- GetIntermediateTouchPoints method, 177**
- GetObject value (NodeType property), 57**
- GetPosition method, 171-175**
- GetTouchPoint method, 177**
- GetValueSource method, 88**
- GetVisualChild method, 497-498**
- GlyphRunDrawing class, 476**
- GradientOrigin property (RadialGradientBrush), 519**
- gradients**
 - GradientSpreadMethod enumeration, 517
 - GradientStop objects, 515
 - LinearGradientBrush class, 515-518
 - RadialGradientBrush class, 519-520
 - transparent colors, 520
- GradientSpreadMethod enumeration, 517**
- GradientStop objects, 515**
- GrammarBuilder class, 671-672**
- grammars**
 - GrammarBuilder class, 671-672
 - Speech Recognition Grammar Specification (SRGS), 670-671
- graphics device interface (GDI), 10**
- graphics hardware, advances in, 11**
- graphics. See 2D graphics; 3D graphics**
- Grid**
 - cell properties, 128
 - compared to other panels, 136
 - explained, 125
 - interaction with child layout properties, 137
 - interactive sizing with GridSplitter, 132-133
 - mimicking Canvas with, 136
 - mimicking DockPanel with, 136
 - mimicking StackPanel with, 136
 - sharing row/column sizes, 134-136
 - ShowGridLines property, 129
 - sizing rows/columns
 - absolute sizing, 130
 - autosizing, 130
 - GridLength structures, 131-132
 - percentage sizing, 131
 - proportional sizing, 130
 - start page with Grid, 126-129
- GridLength structures, 131-132**
- GridLengthConverter, 131**
- GridSplitter class, 132-133**
- GridView control, 290-291**
- GridViewColumn object, 290**
- GroupBox control, 273**
- GroupDescriptions property (ICollectionView), 388**
- grouping, 388-391**
- GroupName property (RadioButton class), 267**

H

- Handled property (RoutedEventArgs class), 162
 - HandleRef, 684
 - hardware acceleration, 12-13
 - HasContent property (ContentControl class), 262
 - HasItems property (ItemsControl class), 276
 - Header property (ToolBar), 306
 - headered items controls, 299
 - HeaderedItemsControl class, 299
 - headers, containers with headers
 - Expander class, 273-274
 - GroupBox class, 273
 - Height property (FrameworkElement class), 98-100
 - Help command, 191-192
 - Hidden value (Visibility enumeration), 102
 - HierarchicalDataTemplate class, 380, 399-400
 - hijacking dependency properties, 441
 - Hillberg, Mike, 384
 - hit testing
 - 3D hit testing, 592-593
 - input hit testing
 - explained, 499
 - InputHitTest method, 513
 - visual hit testing
 - callback methods, 505
 - explained, 499
 - with multiple Visuals, 500-503
 - with overlapping Visuals, 503-505
 - simple hit testing, 499-500
 - HitTest method, 502-505
 - HitTestCore method, 505
 - HitTestFilterCallback delegate, 504
 - HitTestResultCallback delegates, 503
 - HorizontalAlignment property (FrameworkElement class), 103-104
 - HorizontalContentAlignment property (FrameworkElement class), 104-105
 - HostingWin32.cpp file, 685
 - HostingWPF.cpp file, 693-697
 - house drawing, 538-539
 - 2D drawing, 538
 - 3D drawing, 539-540
 - Transform3Ds, 555-556
 - HwndHost class, 685
 - HwndSource class, 692-695
 - HwndSource variable, 697-698
 - hyperlinks, 215-216
-
- ICC (International Color Consortium), 515
 - ICommand interface, 189
 - Icon property (MenuItem class), 299
 - IconResourceIndex property (JumpTask), 238
 - IconResourcePath property (JumpTask), 238
 - ICustomTypeDescriptor interface, 368
 - IEasingFunction interface, 640
 - IList interface, 36
 - Image control, 309-311
 - ImageBrush class, 524-525
 - ImageDrawing class, 476, 478
 - images. *See* 2D graphics; 3D graphics
 - ImageSource class, 310
 - ImageSourceConverter type converter, 309
 - ImeProcessedKey property (KeyEventArgs class), 168
 - immediate-mode graphics systems, 14, 475
 - implicit .NET namespaces, 27
 - implicit styles, creating, 421-422
 - implicit versus explicit runs, 314
 - InAir property (StylusDevice class), 174

- Indeterminate ProgressState, 246**
- inertia, enabling, 183-188**
- Ingebretsen, Robby, 23**
- inheritance**
 - class hierarchy, 73-75
 - property value inheritance, 85-86
 - styles, 418
- InitializeComponent method, 46-48, 198**
- InitialShowDelay property (ToolTip class), 270**
- InkCanvas class, 316-318**
- Inline elements**
 - AnchoredBlock, 326-327
 - defined, 324-325
 - InlineUIContainer, 329
 - LineBreak, 327
 - Span, 325-326
- Inlines property (TextBlock control), 314**
- InlineUIContainer class, 329**
- InnerConeAngle property (PointLights), 568**
- input gestures, executing commands with, 192-193**
- input hit testing**
 - explained, 499
 - InputHitTest method, 513
- InputGestureText property (MenuItem class), 300**
- InputHitTest method, 513**
- inspecting WPF elements, 14**
- instantiating objects**
 - with factory methods, 51-52
 - with non-default constructors, 51
- integration of WPF, 11**
- IntelliSense, 71**
- intensity of lights, 565**
- interfaces. See *specific interfaces***
- International Color Consortium (ICC), 515**
- interoperability (WPF)**
 - ActiveX content, 714-718
 - C++/CLI, 681
 - DirectX content, 15, 708-714
 - explained, 675-677
 - overlapping content, 677
 - Win32 controls
 - explained, 677
 - HwndSource class, 692-695
 - keyboard navigation, 687-691
 - launching modal dialogs, 692, 699
 - layout, 696-699
 - Webcam control, 678-687
 - Windows Forms controls
 - converting between two representatives, 707-708
 - ElementHost class, 704-706
 - explained, 699-700
 - launching modal dialogs, 703, 708
 - PropertyGrid, 700-703
- InvalidItem value (JumpItemRejectionReason enumeration), 244**
- Inverted property (StylusDevice class), 174**
- IsAdditive property (animation classes), 621**
- IsAsync property (Binding object), 401**
- IsCheckable property (MenuItem class), 299**
- IsChecked property (ToggleButton class), 265**
- IsCumulative property (animation classes), 621**
- IsDefault property (Button class), 81, 264**
- IsDefaulted property (Button class), 264**
- IsDown property (KeyEventArgs class), 168**
- IsEditable property (ComboBox), 282**
- IsFrontBufferAvailableChanged event handler, 712**
- IsGrouping property (ItemsControl class), 276**
- IsIndeterminate property (ProgressBar control), 335**

- IsKeyboardFocused** property (UIElement class), 170
- IsKeyDown** method, 169
- IsMouseDirectlyOver** property (UIElement class), 171
- IsNetworkDeployed** method, 231
- isolated storage, 209-210
- IsolatedStorage** namespace, 210
- IsolatedStorageFile** class, 210
- IsolatedStorageFileStream** class, 210
- IsPressed** property (ButtonBase class), 263
- IsReadOnly** property (ComboBox), 282
- IsRepeat** property (KeyEventArgs class), 168
- IsSelected** property (Selector class), 281
- IsSelectionActive** property (Selector class), 281
- IsSynchronizedWithCurrentItem** method, 373
- IsSynchronizedWithCurrentItem** property (Selector), 373
- IsTextSearchCaseSensitive** property (ItemsControl class), 285
- IsTextSearchEnabled** property (ItemsControl class), 285
- IsThreeState** property (ToggleButton class), 265
- IsToggled** property (KeyEventArgs class), 168
- IsUp** property (KeyEventArgs class), 168
- ItemHeight** property (WrapPanel), 120
- items controls**
 - ComboBox
 - ComboBoxItem objects, 286-287
 - customizing selection box, 282-285
 - events, 282
 - explained, 282
 - IsEditable property, 282
 - IsReadOnly property, 282
 - SelectionChanged event, 285-286
 - ContextMenu, 301-302

- DataGrid**
 - auto-generated columns, 294-295
 - CanUserAddRows property, 298
 - CanUserDeleteRows property, 298
 - clipboard interaction, 296
 - ClipboardCopyMode property, 296
 - column types, 293-294
 - displaying row details, 296-297
 - editing data, 297-298
 - EnableColumnVirtualization property, 296
 - EnableRowVirtualization property, 296
 - example, 292-293
 - freezing columns, 297
 - FrozenColumnCount property, 297
 - RowDetailsVisibilityMode property, 297
 - selecting rows/cells, 295
 - SelectionMode property, 295
 - SelectionUnit property, 295
 - virtualization, 296
- GridView**, 290-291
- ItemsControl** class
 - AlternationCount property, 276
 - AlternationIndex property, 276
 - DisplayMemberPath property, 276-277
 - HasItems property, 276
 - IsGrouping property, 276
 - IsTextSearchCaseSensitive property, 285
 - IsTextSearchEnabled property, 285
 - Items property, 275
 - ItemsPanel property, 276-280
 - ItemsSource property, 276
- ListBox**
 - automation IDs, 289
 - example, 287-288
 - scrolling, 289

- SelectionMode property, 288
- sorting items in, 289
- support for multiple selections, 288
- ListView, 290-291
- Menu, 298-301
- scrolling behavior, controlling, 280-281
- Selector class, 281
- StatusBar, 307-308
- TabControl, 291-292
- ToolBar, 304-306
- TreeView, 302-304
- items panels, 278**
- Items property (ItemsControl class), 275, 373**
- ItemsCollection object, 289**
- ItemsControl class**
 - AlternationCount property, 276
 - AlternationIndex property, 276
 - DisplayMemberPath property, 276-277
 - HasItems property, 276
 - IsGrouping property, 276
 - IsTextSearchCaseSensitive property, 285
 - IsTextSearchEnabled property, 285
 - Items property, 275
 - ItemsPanel property, 276-280
 - ItemsSource property, 276
 - scrolling behavior, controlling, 280-281
- ItemsPanel property (ItemsControl class), 276-280**
- ItemsSource collection, 297**
- ItemsSource property (ItemsControl class), 276, 373**
- ItemWidth property (WrapPanel), 120**
- IValueConverter interface, 382-383**
- IXamlLineInfo interface, 58**

J

journal, 216-218

JournalOwnership property (Frame class), 216-217

Jump Lists

- associating with applications, 234

- explained, 233-234

JumpPaths

- adding, 242-243

- explained, 241

- recent and frequent JumpPaths, 243-244

- responding to rejected or removed items, 244

JumpTasks

- customizing behavior of, 237-240

- example, 235

- explained, 234

- and Visual Studio debugger, 236

JumpItemRejectionReason enumeration, 244

JumpItemsRejected event, 244

JumpItemsRemovedByUser event, 244

JumpPaths

- adding, 242-243

- explained, 241

- recent and frequent JumpPaths, 243-244

- responding to rejected or removed items, 244

JumpTasks

- customizing behavior of, 237-240

- example, 235

- explained, 234

K**Kaxaml, 22-23****Key enumeration, 168-169****Key property (KeyEventArgs class), 168****keyboard events, 168-170****keyboard navigation**

customizing, 306

supporting in Win32 controls, 687-688

access keys, 691

tabbing into Win32 content, 688-689

tabbing out of Win32 content, 689-690

KeyboardDevice property (KeyEventArgs class), 168**KeyboardNavigation class, 306****KeyDown event, 168****KeyEventArgs class, 168****keyframe animation**

discrete keyframes, 634-636

easing keyframes, 636

explained, 630

linear keyframes, 631-633

spline keyframes, 633-634

keyless resources, 422-423**KeyStates property**

KeyEventArgs class, 168

QueryContinueDragEventArgs class, 173

KeyUp event, 168**keywords. See specific keywords****L****Label class, 268****Language Integrated Query (LINQ), 396****LastChildFill property (DockPanel), 122****launching modal dialogs**

from Win32 applications, 699

from Windows Forms applications, 708

from WPF applications, 692, 703

layout

content overflow, handling

clipping, 139-141

explained, 139

scaling, 143-147

scrolling, 141-143

custom panels

communication between parents and children, 752-755

explained, 751-752

FanCanvas, 768-772

OverlapPanel, 763-768

SimpleCanvas, 755-760

SimpleStackPanel, 760-763

explained, 97-98

panels

Canvas, 116-118

DockPanel, 122-125

explained, 115-116

Grid. *See* Grid

SelectiveScrollingGrid, 138-139

StackPanel, 118-119

TabPanel, 137

ToolBarOverflowPanel, 138

ToolBarPanel, 138

ToolBarTray, 138

UniformGrid, 138

WrapPanel, 120-122

positioning elements

content alignment, 104-105

explained, 103

flow direction, 105-106

- horizontal and vertical alignment, 103-104
- stretch alignment, 104
- sizing elements
 - explained, 98
 - explicit sizes, avoiding, 99
 - height and width, 98-100
 - margin and padding, 100-102
 - visibility, 102-103
- transforms
 - applying, 106-107
 - combining, 113-114
 - explained, 106
 - MatrixTransform, 112-113
 - RotateTransform, 108-109
 - ScaleTransform, 109-111
 - SkewTransform, 112
 - support for, 114
 - TranslateTransform, 112
- Visual Studio-like panes, creating
 - sequential states of user interface, 147-151
 - VisualStudioLikePanes.xaml, 151-153
 - VisualStudioLikePanes.xaml.cs, 153-157
- LayoutTransform property (FrameworkElement class), 106**
- Left property (Canvas), 116**
- LengthConverter type converter, 102**
- Light and Fluffy skin example, 463-464**
- Light objects**
 - AmbientLight, 564, 569-570
 - defined, 563
 - DirectionalLight, 564-565
 - explained, 542, 563
 - intensity of, 565
 - PointLight, 564-566
 - SpotLight, 564, 566-568
- Line class, 509-510**
- linear interpolation, 612-613**
- linear keyframes, 631-633**
- LinearAttenuation property (PointLights), 566**
- LinearGradientBrush class, 515-518**
- LineBreak class, 327**
- LineGeometry class, 479**
- LineJoin property (Pen class), 490**
- LineSegment class, 480**
- LINQ (Language Integrated Query), 396**
- ListBox control**
 - arranging items horizontally, 279
 - automation IDs, 289
 - example, 287-288
 - placing PlayingCards custom control into, 742
 - scrolling, 289
 - SelectionMode property, 288
 - sorting items in, 289
 - support for multiple selections, 288
- lists, 36-37**
 - Jump Lists
 - associating with applications, 234
 - explained, 233-234
 - JumpPaths, 241-244
 - JumpTasks, 234-240
 - and Visual Studio debugger, 236
 - ListBox control
 - arranging items horizontally, 279
 - automation IDs, 289
 - example, 287-288
 - placing PlayingCards custom control into, 742
 - scrolling, 289
 - SelectionMode property, 288
 - sorting items in, 289
 - support for multiple selections, 288
 - ListView control, 290-291

ListView control, 290-291

- live objects, writing to, 61-63
- Load method, 40-41, 64
- LoadAsync method, 41
- LoadComponent method, 47
- loading XAML at runtime, 40-42
- Lobo, Lester, 23
- local values, clearing, 88
- localization IDs, marking user interfaces with, 351
- localizing binary resources
 - creating satellite assembly with LocBaml, 351
 - explained, 350
 - marking user interfaces with localization IDs, 351
 - preparing projects for multiple cultures, 350
- LocBaml, creating satellite assembly with, 351**
- locking D3DImage, 713**
- logical AND relationships, 429-430**
- logical OR relationships, 429**
- logical resources**
 - accessing directly, 360
 - consolidating color brushes with, 353-355
 - defining and applying in procedural code, 359-360
 - explained, 351-352
 - interaction with system resources, 360-361
 - resource lookup, 355
 - resources without sharing, 358
 - static versus dynamic resources, 355-357
- logical trees, 75-80**
- LogicalChildren property, 80**
- LogicalTreeHelper class, 77**
- LookDirection property (Cameras), 544-548**
- lookup, resource lookup, 355
- loose XAML pages, 231-232

M

- mage.exe command-line tool, 210
- mageUI.exe graphical tool, 210
- Main method, 199-201
- MainWindow class, 197-198
- MainWindow.xaml file, 710
- MainWindow.xaml.cs file, 178-179, 186-187, 710-712
- malicious skins, preventing, 464-465
- managed code, mixing with unmanaged code, 682
- manipulation events**
 - adding inertia with, 183-188
 - enabling panning/rotating/zooming with, 182-183
 - explained, 180-181
 - ManipulationCompleted, 181
 - ManipulationDelta, 181
 - ManipulationStarted, 181
 - ManipulationStarting, 181
- ManipulationBoundaryFeedback event, 185**
- ManipulationCompleted event, 181**
- ManipulationDelta event, 181-183**
- ManipulationDeltaEventArgs instance, 181**
- ManipulationInertiaStarting event, 183, 187**
- ManipulationStarted event, 181**
- ManipulationStarting event, 181**
- Margin property (FrameworkElement class), 100-102**
- marking user interfaces with localization IDs, 351
- markup compatibility, 61
- markup extensions**
 - explained, 32-35
 - parameters, 33
 - in procedural code, 35

Materials

- AmbientMaterial, 575
- combining, 578
- DiffuseMaterial, 572-575
- EmissiveMaterial, 576-578
- explained, 571

MatrixCamera class, 553

MatrixTransform, 112-113

MatrixTransform3D class, 562

MDI (multiple-document interface), 203

MeasureOverride method, overriding, 752-754

MediaCommands class, 190

MediaElement class

- playing audio, 656-658
- playing video, 658-660

MediaPlayer class, 655-656

MediaTimeline class

- playing audio, 656-658
- playing video, 661-662

Menu control, 298-301

MenuItem class, 299

menus

- ContextMenu control, 301-302
- Menu control, 298-301
- MenuItem class, 299

MergedDictionaries property (ResourceDictionary class), 357

MeshGeometry3D class, 578-579

- Normals property, 581-583
- Positions property, 579
- TextureCoordinates property, 583
- TriangleIndices property, 580-581

methods, binding to, 402-403. See also specific methods

Microsoft Anna, 664

missing styles, troubleshooting, 461

mnemonics, 691

modal dialogs, launching

- from Win32 applications, 699
- from Windows Forms applications, 708
- from WPF applications, 692, 703

Model3DGroup class, 563, 584-586

Model3Ds

- explained, 563
- GeometryModel3D
 - defined, 563
 - explained, 571
 - Geometry3D class, 578
 - Materials, 571-578
 - MeshGeometry3D class, 578-583

Lights

- AmbientLight, 564, 569-570
- DirectionalLight, 564-565
- explained, 563
- intensity of, 565
- PointLight, 564-566
- SpotLight, 564-568
- Model3DGroup class, 563, 584-586

modeless dialogs, 196

ModelUIElement3D class, 588-590

ModelVisual3D class, 587-588

Modifiers property (KeyboardDevice), 169

Mouse class, 173

mouse events

- capturing, 173-174
- drag-and-drop events, 172-173
- explained, 170-171
- MouseButtonEventArgs, 171
- MouseEventArgs, 171-172
- MouseWheelEventArgs, 171
- transparent and null regions, 171

MouseButtonEventArgs class, 171

MouseButtonState enumeration, 171

MouseEventArgs class, 171-172

MouseWheelEventArgs class, 171

multi-touch events

basic touch events, 177-180

explained, 176

manipulation events

adding inertia with, 183-188

enabling panning/rotating/zooming with, 182-183

explained, 180-181

ManipulationCompleted, 181

ManipulationDelta, 181

ManipulationStarted, 181

ManipulationStarting, 181

multi-touch support, 16

MultiBinding class, 410-411

multiple providers, support for

applying animations, 89

coercion, 89

determining base values, 87-88

evaluating, 89

explained, 87

validation, 89

multiple Visuals, hit testing with, 500-503

multiple-document interface (MDI), 203

MultiPoint Mouse SDK, 176

multithreaded applications, 205

MyHwndHost class, 684-686

N

Name keyword, 42

named elements, 434

named styles, 421-422

NamespaceDeclaration value (NodeType property), 57

namespaces

explained, 26-28

implicit .NET namespaces, 27

mapping, 26

naming elements, 42-43

Navigate method, 214-215

navigation

keyboard navigation, supporting in Win32 controls, 687-691

views, 392-393

XAML Browser applications, 228-229

navigation-based Windows applications

explained, 211-212

hyperlinks, 215-216

journal, 216-218

Navigate method, 214-215

navigation containers, 212-214

navigation events, 218-219

Page elements, 212-214

returning data from pages, 221-222

sending data to pages, 220-221

NavigationCommands class, 190

NavigationProgress event, 219

NavigationStopped event, 219

NavigationWindow class, 212-214

nearest-neighbor bitmap scaling, 310

.NET properties, binding to, 367-368

NodeType property (XAML), 57-58

None ProgressState, 246

None value (NodeType property), 58

nonprinciple axis, scaling about, 559

NonZero fill (FillRule property), 482

NoRegisteredHandler value (JumpItemRejectionReason enumeration), 244

Normal ProgressState, 246

normals, 581

Normals property (MeshGeometry3D class), 581-583

null regions, 171



Object class, 73

object elements

- attributes, 25
- content property, 35-36
- declaring, 25
- dictionaries, 37-38
- explained, 24-26
- lists, 36-37
- naming, 42-43
- positioning
 - content alignment, 104-105
 - explained, 103
 - flow direction, 105-106
 - horizontal and vertical alignment, 103-104
 - stretch alignment, 104
- processing child elements, 40
- sizing
 - explained, 98
 - explicit sizes, avoiding, 99
 - height and width, 98-100
 - margin and padding, 100-102
 - visibility, 102-103
- transforms
 - applying, 106-107
 - combining, 113-114
 - explained, 106
 - MatrixTransform, 112-113
 - RotateTransform, 108-109

ScaleTransform, 109-111

SkewTransform, 112

support for, 114

TranslateTransform, 112

values type-converted to object elements, 38

ObjectDataProvider class, 401-403

objects

- binding to, 369-370
- instantiating via factory methods, 51-52
- instantiating with non-default constructors, 51
- live objects, writing to, 61-63
- logical trees, 75-76
- Object class, 73
- visual trees, 76-80

on-demand download (XAML Browser applications), 230-231

OneTime binding, 403

OneWay binding, 403

OneWayToSource binding, 403-404

OnMnemonic method, 691

OnNoMoreTabStops method, 690

opacity masks, brushes as, 527-529

Opacity property (brushes), 527

OpacityMask property (brushes), 527-529

OpenGL, 10

opening project files in Visual Studio, 350

OR relationships (logical), 429

order of property and event processing, 26

Orientation property

ProgressBar control, 335

StackPanel, 118

WrapPanel, 120

OriginalSource property (RoutedEventArgs class), 162

OrthographicCamera class

- blind spots, 545
- compared to PerspectiveCamera class, 551-553
- LookDirection property, 544-548
- Position property, 543-544
- UpDirection property, 548-550
- Z-fighting, 545

OuterConeAngle property (PointLights), 568

OverlapPanel, 763-768

overlapping content, 677

overlapping Visuals, hit testing with, 503-505

Overlay property (TaskbarItemInfo), 247

overlays, adding to taskbar items, 247

overriding

- ArrangeOverride method, 754-755
- MeasureOverride method, 752-754

P

packageURI, 349

Padding property (FrameworkElement class), 100-102

Page elements, 212-214

PageFunction class, 221-222

pages

- loose XAML pages, 231-232
- Page elements, 212-214
- refreshing, 217
- returning data from, 221-222
- sending data to, 220-221
- stopping loading, 217

panels

- Canvas, 116-118, 136
- content overflow, handling
 - clipping, 139-141
 - explained, 139

scaling, 143-147

scrolling, 141-143

custom panels

- communication between parents and children, 752-755
- explained, 751-752
- FanCanvas, 768-772
- OverlapPanel, 763-768
- SimpleCanvas, 755-760
- SimpleStackPanel, 760-763

DockPanel

- examples, 122-125
- explained, 122
- interaction with child layout properties, 125
- mimicking with Grid, 136
- properties, 122

explained, 115-116

Grid

- cell properties, 128
- compared to other panels, 136
- explained, 125
- interaction with child layout properties, 137
- interactive sizing with GridSplitter, 132-133
- mimicking Canvas with, 136
- mimicking DockPanel with, 136
- mimicking StackPanel with, 136
- sharing row/column sizes, 134-136
- ShowGridLines property, 129
- sizing rows/columns, 130-132
- start page with Grid, 126-129

SelectiveScrollingGrid, 138-139

- StackPanel
 - explained, 118
 - interaction with child layout properties, 119
 - mimicking with Grid, 136
- TabPanel, 137
- ToolBarOverflowPanel, 138
- ToolBarPanel, 138
- ToolBarTray, 138
- UniformGrid, 138
- Visual Studio-like panes, creating
 - sequential states of user interface, 147-151
 - VisualStudioLikePanes.xaml, 151-153
 - VisualStudioLikePanes.xaml.cs, 153-157
- WrapPanel
 - examples, 121
 - explained, 120
 - interaction with child layout properties, 121-122
 - properties, 120
 - and right-to-left environments, 121
- panning**
 - enabling with multi-touch events, 182-183
 - with inertia, 184-185
- Paragraph Blocks, 320**
- Parse method, 64**
- parsing XAML at runtime, 40-42**
- partial keyword, 44**
- partial-trust applications, 15**
- parts (control), 447-449**
- PasswordBox control, 316**
- path-based animations, 637**
- Path class, 511-512**
- PathGeometry class**
 - ArcSegment, 480
 - BezierSegment, 480
 - example, 480-482
 - explained, 479
 - FillRule property, 482-483
 - LineSegment, 480
 - PolyBezierSegment, 480
 - PolyLineSegment, 480
 - PolyQuadraticBezierSegment, 480
 - QuadraticBezierSegment, 480
- Paused ProgressState, 246**
- Pen class, 489-491**
- percentage sizing, 131**
- performance**
 - cached composition
 - BitmapCache class, 533-535
 - BitmapCacheBrush class, 535
 - Viewport2DVisual3D support for, 591
 - improving rendering performance
 - BitmapCache class, 533-535
 - BitmapCacheBrush class, 535
 - RenderTargetBitmap class, 532-533
 - XAML, 71
 - WPF 3.5 enhancements, 16
 - WPF 4 enhancements, 17
- persisting application state, 209-210**
- PerspectiveCamera class**
 - blind spots, 545
 - compared to OrthographicCamera class, 551-553
 - LookDirection property, 544-548
 - Position property, 544
 - UpDirection property, 548-550
 - Z-fighting, 545
- Petzold, Charles, 23**
- PIvoke, 251**
- PixelFormats enumeration, 311**

pixels

- device-independent pixels, 102

- pixel boundaries, 17

- pixel shaders, 531

Play method, 654**PlayingCard control**

- behavior

- code-behind file, 734

- final implementation, 737-739

- initial implementation, 733-737

- resources, 734-735

- generic resources, 741-742

- placing into ListBox, 742

- user interface, 739-742

PointLight, 564-566**PolyBezierSegment class, 480****Polygon class, 511****Polyline class, 510****PolyLineSegment class, 480****PolyQuadraticBezierSegment class, 480****Position property (Cameras), 543-544****positioning elements**

- content alignment, 104-105

- explained, 103

- flow direction, 105-106

- horizontal and vertical alignment, 103-104

- stretch alignment, 104

Positions property (MeshGeometry3D class), 579**power easing functions, 637-638****PressureFactor property (StylusPoint object), 175****PreviewKeyDown event, 168****PreviewKeyUp event, 168****printing logical/visual trees, 78-79****PrintLogicalTree method, 79****PrintVisualTree method, 78****PriorityBinding class, 411****procedural code**

- accessing binary resources from, 349-350

- animation classes

- AutoReverse property, 618

- BeginTime property, 616-617

- DoubleAnimation, 611-612

- Duration property, 614

- EasingFunction property, 620

- explained, 608-610

- FillBehavior property, 621

- From property, 614-616

- IsAdditive property, 621

- IsCumulative property, 621

- lack of generics, 610-611

- linear interpolation, 612-613

- RepeatBehavior property, 618-619

- reusing animations, 613

- SpeedRatio property, 617

- To property, 614-616

- Binding object in, 363-365

- compared to XAML, 24

- defining and applying resources in, 359-360

- embedding PropertyGrid with, 700-702

- frame-based animation, 609

- markup extensions in, 35

- mixing XAML with

- BAML (Binary Application Markup Language), 45-48

- CAML (Compiled Application Markup Language), 46

- compiling XAML, 43-45

- generated source code, 46

- loading and parsing XAML at runtime, 40-42

- naming XAML elements, 42-43

- procedural code inside XAML, 47

- skins, 462
- timer-based animation, 608
- type converters in, 31
- inside XAML, 47
- procedural code timer-based animation, 609**
- ProgressBar, 335**
 - adding to taskbars, 246
 - pie chart control template, 442-444, 453-455
- ProgressState property (TaskbarItemInfo), 246**
- ProgressValue property (TaskbarItemInfo), 246**
- project files, opening in Visual Studio, 350**
- PromptBuilder class, 665-667**
- properties. See also specific properties**
 - dependency properties
 - attached properties, 89-93
 - attached property providers, 92
 - change notification, 83-84
 - explained, 80-81
 - implementation, 81-83
 - property value inheritance, 85-86
 - support for multiple providers, 87-89
 - .NET properties, binding to, 367-368
 - order of processing, 26
 - Properties collection, 203
- Properties collection, 203**
- property attributes, 25**
- property elements, 29-30**
- property paths, 277**
- property triggers, 83-85, 424-427, 628-629**
- property value inheritance, 85-86**
- property wrappers, 82**
- PropertyGrid**
 - embedding with procedural code, 700-702
 - embedding with XAML, 702-703
- PropertyGroupDescription class, 390**

- proportional sizing, 130**
- protecting controls from accidental usage, 727-728**
- pure-XAML Twitter client, 412-413**

Q

- QuadraticAttenuation property (PointLights), 566**
- QuadraticBezierSegment class, 480**
- QuaternionRotation3D class, 559**
- QueryContinueDragEventArgs class, 173**

R

- RadialGradientBrush class, 519-520**
- RadioButton class, 266-268**
- RadiusX property**
 - RadialGradientBrush, 519
 - Rectangle class, 507
- RadiusY property**
 - RadialGradientBrush, 519
 - Rectangle class, 507
- range controls**
 - explained, 334
 - ProgressBar, 335
 - Slider, 335-336
- Range property (PointLights), 566**
- raw project files, opening in Visual Studio, 350**
- readers (XAML)**
 - explained, 53-54
 - markup compatibility, 61
 - node loops, 56-57
 - NodeType property, 57-58

- sample XAML content, 58-59
- XAML node stream, 59-61
- XamlServices class, 64-67
- recent and frequent JumpPaths, 243-244**
- Rectangle class, 507-508**
- RectangleGeometry class, 479**
- Refresh method, 217**
- refreshing pages, 217**
- Register method, 82**
- rejected items, reponding to, 244**
- RelativeSource property (Binding object), 367**
- releases of WPF**
 - future releases, 17
 - WPF 3.0, 14
 - WPF 3.5, 14-16
 - WPF 3.5 SP1, 15-16
 - WPF 4, 14, 16-17
 - WPF Toolkit, 14
- removed items, reponding to, 244**
- RemovedByUser value (JumpItemRejectionReason enumeration), 244**
- RemoveHandler method, 160-161**
- removing Binding objects, 365**
- RenderAtScale property (BitmapCache class), 533**
- rendering**
 - custom rendering, 499
 - improving rendering performance
 - BitmapCache class, 533-535
 - BitmapCacheBrush class, 535
 - RenderTargetBitmap class, 532-533
 - text, 17
 - TextOptions class, 312
 - WPF 4 enhancements, 311-312
- Rendering event, 609**
- rendering, controlling**
 - data templates, 378-380
 - explained, 375
 - string formatting, 375-377
 - value converters
 - Binding.DoNothing values, 385
 - bridging incompatible data types, 381-384
 - customizing data display, 385
 - explained, 381
- RenderSize property (FrameworkElement class), 99**
- RenderTargetBitmap class, 532-533**
- RenderTransform property (FrameworkElement class), 106**
- RenderTransformOrigin property (UIElement class), 107**
- RepeatBehavior property (animation classes), 618-619**
- RepeatButton class, 265**
- ResizeBehavior property (GridSplitter), 133**
- ResizeDirection property (GridSplitter), 133**
- resolution independence, 12**
- Resource build action, 344-345**
- ResourceDictionary class, 357**
- ResourceDictionaryLocation parameter, 467**
- resources**
 - binary resources
 - accessing, 345-350
 - defining, 344-345
 - explained, 343
 - localizing, 350-351
 - defined, 343
 - keyless resources, 422-423
 - logical resources
 - accessing directly, 360
 - consolidating color brushes with, 353-355

- defining and applying in procedural code, 359-360
 - explained, 351-352
 - interaction with system resources, 360-361
 - resource lookup, 355
 - resources without sharing, 358
 - static versus dynamic resources, 355-357
 - for PlayingCard custom control, 734-735
 - responding to rejected or removed items, 244**
 - restoring application state, 209-210**
 - restricting style usage, 420-421**
 - results, dialog results, 208**
 - retained-mode graphics systems, 14, 475-476**
 - returning data from pages, 221-222**
 - reusing animations, 613**
 - RichTextBox control, 316**
 - Right property (Canvas), 116**
 - right-hand rule, 543, 580**
 - right-handed coordinate systems, 543-544**
 - RotateTransform, 108-109**
 - RotateTransform3D class, 559-562**
 - rotation**
 - enabling with multi-touch events, 182-183
 - with inertia, 184-185
 - RotateTransform3D class, 559-562
 - Rotation property (ManipulationDelta class), 181**
 - routed events**
 - About dialog example, 162-164
 - adding to user controls, 731-732
 - attached events, 165-167
 - consolidating routed event handlers, 167-168
 - defined, 159
 - explained, 159-160
 - implementation, 160-161
 - RoutedEventArgs class, 162
 - routing strategies, 161-162
 - stopping, 165
 - RoutedEvent property (RoutedEventArgs class), 162**
 - RoutedEventArgs class, 162**
 - RoutedUICommand objects, 190**
 - routing strategies, 161-162**
 - RoutingStrategy enumeration, 161**
 - RowDetailsVisibilityMode property (DataGrid), 297**
 - rows (Grid)**
 - displaying row details, 296-297
 - selecting, 295
 - sharing row/column sizes, 134-136
 - sizing
 - absolute sizing, 130
 - autosizing, 130
 - GridLength structures, 131-132
 - interactive sizing with GridSplitter, 132-133
 - percentage sizing, 131
 - proportional sizing, 130
 - rules, 405-409**
 - Run method, 200-201**
 - running XAML examples, 22**
 - runtime, loading and parsing XAML at, 40-42**
- S**
- satellite assemblies, creating with LocBaml, 351**
 - Save method, 64**
 - Scale property (ManipulationDelta class), 181**
 - ScaleTransform, 109-111, 144**

ScaleTransform3D class, 557-559

ScaleX property (RotateTransform class), 109

ScaleY property (RotateTransform class), 109

scaling, 143-147

nearest-neighbor bitmap scaling, 310

about nonprinciple axis, 559

ScaleTransform3D class, 557-559

scope of typed styles, 421

scRGB color space, 514

ScrollBars, 142-143

scrolling behavior, 141-143

controlling in items controls, 280-281

ListBox control, 289

ScrollViewer control, 141-143

Section Blocks, 320

security, XAML Browser applications, 229

SelectedDatesChanged event, 339

SelectedIndex property (Selector class), 281

SelectedItem property (Selector class), 281

SelectedValue property (Selector class), 281

selecting rows/cells, 295

**selection boxes (ComboBox control),
customizing, 282-285**

SelectionChanged event, 281, 285-286

SelectionMode property

Calendar control, 337

DataGrid, 295

ListBox, 288

SelectionUnit property (DataGrid), 295

SelectiveScrollingGrid, 138-139

Selector class, 281

selectors, data template selectors, 381

SelectVoice method, 664

SelectVoiceByHints method, 664

sending data to pages, 220-221

Separator control, 299

SetBinding method, 365

SetCurrentValue method, 89

SetOutputToDefaultAudioDevice method, 665

SetOutputToWaveFile method, 665

SetResourceReference method, 359

Setters, 419-420

Settings class, 210

ShaderEffect, 530-531

Shapes

clip art based on Shapes, 512-513

Ellipse class, 508

explained, 505-506

how they work, 509

Line class, 509-510

overuse of, 507

Path class, 511-512

Polygon class, 511

Polyline class, 510

Rectangle class, 507-508

sharing

data source with DataContext, 374-375

Grid row/column sizes, 134-136

resources without sharing, 358

styles, 418-420

ShowDialog method, 208-209

ShowDuration property (ToolTip class), 270

**ShowFrequentCategory property (JumpList
class), 243**

ShowGridLines property (Grid), 129

ShowOnDisabled property

ContextMenuService class, 302

ToolTipService class, 271

**ShowRecentCategory property (JumpList
class), 243**

ShutdownMode enumeration, 202

Silicon Graphics OpenGL, 10

Silverlight, 18-19, 180

**Silverlight XAML Vocabulary Specification 2008
(MS-SLXV), 24**

- SimpleCanvas, 755-760**
- SimpleQuadraticEase class, 641**
- SimpleStackPanel, 760-763**
- SineEase function, 640**
- single-instance applications, 204**
- single-threaded apartment (STA), 199**
- sizing**
 - Grid rows/columns
 - absolute sizing, 130
 - autosizing, 130
 - GridLength structures, 131-132
 - interactive sizing with GridSplitter, 132-133
 - percentage sizing, 131
 - proportional sizing, 130
 - sharing row/column sizes, 134-136
 - elements
 - explained, 98
 - explicit sizes, avoiding, 99
 - height and width, 98-100
 - margin and padding, 100-102
 - visibility, 102-103
- SkewTransform, 112**
- skins**
 - defined, 415
 - examples, 459-461
 - explained, 458-459, 462
 - Light and Fluffy skin example, 463-464
 - malicious skins, preventing, 464-465
 - missing styles, troubleshooting, 461
 - procedural code, 462
- Skip method, 63**
- Slider control, 335-336**
- snapshots of individual video frames, taking, 660**
- SnapsToDevicePixels property, 17, 534**
- Snoop, 14**
- SolidColorBrush class, 514**
- SortDescription class, 395**
- SortDescriptions collection, 387**
- SortDescriptions property**
 - ICollectionView class, 386
 - ItemsCollection object, 289
- sorting, 289, 386-388**
- SoundPlayer class, 654**
- SoundPlayerAction class, 654-655**
- Source property**
 - MediaElement class, 656
 - RoutedEventArgs class, 162
- SourceName property (Trigger class), 433**
- spaces in geometry strings, 489**
- spans, 325-326**
- SpeakAsync method, 664**
- SpeakAsyncCancelAll method, 664**
- speech recognition**
 - converting spoken words into text, 667-670
 - specifying grammar with GrammarBuilder, 671-672
 - specifying grammar with SRGS, 670-671
- Speech Recognition Grammar Specification (SRGS), 670-671**
- speech synthesis**
 - explained, 664
 - GetInstalledVoices method, 664
 - PromptBuilder class, 665-667
 - SelectVoice method, 664
 - SelectVoiceByHints method, 664
 - SetOutputToWaveFile method, 665
 - SpeakAsync method, 664
 - Speech Synthesis Markup Language (SSML), 665-667
 - SpeechSynthesizer, 664
- Speech Synthesis Markup Language (SSML), 665-667**

SpeechRecognitionEngine class, 669-670

SpeechSynthesizer, 664

SpeedRatio property (animation classes), 617

spell checking, 315

Spinning Prize Wheel, 186-187

splash screens, 205-206

spline keyframes, 633-634

SpotLight, 564-568

**SpreadMethod property
(LinearGradientBrush), 517**

Square line cap (Pen), 490

sRGB color space, 514

**SRGS (Speech Recognition Grammar
Specification), 670-671**

**SSML (Speech Synthesis Markup Language),
665-667**

STA (single-threaded apartment), 199

StackPanel. See also SimpleStackPanel

explained, 118

interaction with child layout properties, 119

mimicking with Grid, 136

setting font properties on, 90-91

with Menu control, 300

standard Windows applications

Application class

creating applications without, 204

events, 202

explained, 199-200

Properties collection, 203

Run method, 200-201

Windows collection, 202

application state, 209-210

ClickOnce, 210-211

common dialogs, 206-207

custom dialogs, 207-208

explained, 195-196

multiple-document interface (MDI), 203

multithreaded applications, 205

retrieving command-line arguments in, 202

single-instance applications, 204

splash screens, 205-206

Window class, 196-198

Windows Installer, 210

start pages, building with Grid, 126-129

**starting animations from property triggers,
628-629**

StartLineCap property (Pen class), 489

StartMember value (NodeType property), 57

StartObject value (NodeType property), 57

StartPoint property (LinearGradientBrush), 516

StartupUri property (Application class), 200-201

states

control states, 449-455, 745-749

persisting and restoring, 209-210

visual states

respecting with triggers, 442-446

respecting with VSM (Visual State
Manager), 447-455

STAThreadAttribute, 695

static versus dynamic resources, 355-357

StaticResource markup extension, 355-357

StatusBar control, 307-308

StopLoading method, 217

stopping

page loading, 217

routed events, 165

Storyboards

EventTriggers containing Storyboards,
621-622

Storyboards as Timelines, 629-630

TargetName property, 625-626

TargetProperty property, 622-625

StreamGeometry class, 483

Stretch alignment, 104

Stretch enumeration, 144**Stretch property**

- DrawingBrush class, 521
- MediaElement class, 658

StretchDirection enumeration, 144**StretchDirection property (MediaElement class), 658****StringFormat property (Binding object), 375-376****strings**

- formatting, 375-377
- representing geometries as, 487-489

Stroke objects, 317**structures, ValueSource, 88****styles**

- consolidating property assignments in, 417
- default styles, 88
- defined, 415
- explained, 416-418
- implicit styles, creating, 421-422
- inheritance, 418
- keyless resources, 422-423
- missing styles, troubleshooting, 461
- mixing with control templates, 456-457
- named styles, 421-422
- per-theme styles and templates, 466-469
- restricting usage of, 420-421
- Setter behavior, 419-420
- sharing, 418-420
- theme styles, 88
- triggers
 - conflicting triggers, 429
 - data triggers, 427-428
 - explained, 423-424
 - expressing logic with, 428-430
 - property triggers, 424-427
 - respecting visual states with, 442-446
- typed styles, 421-422

stylus events, 174-176**StylusButtonEventArgs instance, 176****StylusButtons property (StylusDevice class), 175****StylusDevice class, 174-175****StylusDownEventArgs instance, 176****StylusEventArgs class, 176****StylusPoint objects, 175****StylusSystemGestureEventArgs instance, 176****Surface Toolkit for Windows Touch, 188****system resources, interaction with logical resources, 360-361****SystemKey property (KeyEventArgs class), 168****SystemSounds class, 654****T****TabControl control, 291-292****TabInto method, 688****Table Blocks, 320****TabletDevice property (StylusDevice class), 175****TabPanel, 137****TargetName property (Storyboards), 625-626****TargetNullValue property (Binding object), 366****TargetProperty property (Storyboards), 622-625****TargetType property**

- ControlTemplate class, 434-435
- Style class, 420-421

taskbar, customizing

- explained, 245-246
- taskbar item overlays, 247
- taskbar item progress bars, 246
- thumb buttons, 248-249
- thumbnail content, 247

TaskDialogs, 253-256

tasks, JumpTasks

- customizing behavior of, 237-240
- example, 235
- explained, 234

TemplateBindingExtension class, 435-437**templated parent properties, respecting, 435-439**

- Content property (ContentControl class), 435-437
- hijacking existing properties for new purposes, 441
- other properties, 440

templates

- control templates
 - editing, 457-458
 - mixing with styles, 456-457
 - named elements, 434
 - resuability of, 438-440
 - simple control template, 431-432
 - target type, restricting, 434-435
 - templated parent properties, respecting, 435-441
 - other properties, 438-439
 - triggers, 432-434
 - visual states, respecting with triggers, 442-446
 - visual states, respecting with VSM (Visual State Manager), 447-455
- DataTemplates, 378-380
- defined, 415
- explained, 430-431
- HierarchicalDataTemplate, 399-400
- per-theme styles and templates, 466-469
- template selectors, 381
- Windows themes, 470

temporarily canceling data binding, 385**testing**

- 3D hit testing, 592-593
- input hit testing
 - explained, 499
 - InputHitTest method, 513
- visual hit testing
 - callback methods, 505
 - explained, 499
 - simple hit testing, 499-500
 - with multiple Visuals, 500-503
 - with overlapping Visuals, 503-505

text

- converting spoken words into, 667-670
- InkCanvas class, 316-318
- PasswordBox control, 316
- rendering, 17, 311-312
- RichTextBox control, 316
- text-to-speech
 - explained, 664
 - GetInstalledVoices method, 664
 - PromptBuilder class, 665-667
 - SelectVoice method, 664
 - SelectVoiceByHints method, 664
 - SetOutputToWaveFile method, 665
 - SpeakAsync method, 664
 - Speech Synthesis Markup Language (SSML), 665-667
 - SpeechSynthesizer, 664
- TextBlock control
 - explained, 313-314
 - explicit versus implicit runs, 314
 - properties, 313
 - support for multiple lines of text, 315
 - whitespace, 314
- TextBox control, 315
- TextOptions class, 312

text-to-speech

explained, 664

GetInstalledVoices method, 664

PromptBuilder class, 665-667

SelectVoice method, 664

SelectVoiceByHints method, 664

SetOutputToWaveFile method, 665

SpeakAsync method, 664

Speech Synthesis Markup Language (SSML), 665-667

SpeechSynthesizer, 664

TextBlock control

explained, 313-314

explicit versus implicit runs, 314

properties, 313

support for multiple lines of text, 315

whitespace, 314

TextBox control, 315**TextElement class, 319-320****Blocks**

AnchoredBlock class, 326-327

BlockUIContainer, 321

List, 320

Paragraph, 320

sample code listing, 321-324

Section, 320

Table, 320

Inlines

AnchoredBlock, 326-327

defined, 324-325

InlineUIContainer, 329

LineBreak, 327

Span, 325-326

TextFormattingMode property (TextOptions), 312**TextHintingMode property (TextOptions), 312****TextOptions class, 312****TextRenderingMode property (TextOptions), 312****texture coordinates, 584****TextureCoordinates property (MeshGeometry3D class), 583****theme dictionaries, 466****theme styles, 88****ThemeDictionaryExtension, 468****ThemeInfoAttribute, 467-468****themes**

defined, 415, 465

generic dictionaries, 467

per-theme styles and templates, 466-469

system colors, fonts, and parameters, 465-466

theme dictionaries, 466

Thickness class, 100-102**ThicknessConverter type converter, 102****thumb buttons (taskbar), adding, 248-249****ThumbButtonInfo property (TaskbarItemInfo), 248-249****thumbnail content (taskbar), customizing, 247****ThumbnailClipMargin property (TaskbarItemInfo), 247****tile brushes**

DrawingBrush class, 520-524

ImageBrush class, 524-525

VisualBrush class, 525-527

TileMode enumeration, 523**TileMode property (DrawingBrush class), 521-523****Timelines, 629-630****timer-based animation, 608-609****To property (animation classes), 614-616****ToggleButton class, 265-266****ToolBar control, 304-306****ToolBarOverflowPanel, 138****ToolBarPanel, 138****ToolBarTray class, 138, 305**

ToolTip class, 269-271**ToolTipService class, 271****Top property (Canvas), 116****touch events, 177-180****TouchEvent property (TouchEventArgs class), 177****TouchDown event, 178-180****TouchEventArgs class, 177****TouchMove event, 178-180****TouchUp event, 178-180****TraceSource object, 384****Transform method, 65****Transform property (Cameras), 549****Transform3Ds**

combining, 562

explained, 554-555

house drawing example, 555-556

MatrixTransform3D class, 554, 562

RotateTransform3D class, 554, 559-562

ScaleTransform3D class, 554, 557-559

Transform3DGroup class, 554

TranslateTransform3D class, 554-557

TransformConverter type converter, 113**transforms**

applying, 106-107

clipping and, 141

combining, 113-114

explained, 106

MatrixTransform, 112-113

RotateTransform, 108-109

ScaleTransform, 109-111

SkewTransform, 112

support for, 114

Transform3Ds

combining, 562

explained, 554-555

house drawing example, 555-556

MatrixTransform3D class, 554, 562

RotateTransform3D class, 554, 559-562

ScaleTransform3D class, 554, 557-559

Transform3DGroup class, 554

TranslateTransform3D class, 554-557

TranslateTransform, 112

TransformToAncestor method, 596-605**TransformToDescendant method, 600-605****transitions (animation), 647-651****Transitions property (VisualStateGroup class), 455****TranslateAccelerator method, 689-691****TranslateTransform, 112****TranslateTransform3D class, 556-557****Translation property (ManipulationDelta class), 181****transparent colors, 520****transparent regions and mouse events, 171****trees**

logical trees, 75-76

visual trees, 76-80

TreeView control, 302-304**TreeViewItem class, 303-304****TriangleIndices property (MeshGeometry3D class), 580-581****Trigger class. See triggers****TriggerBase class, 85****triggers**

conflicting triggers, 429

data triggers, 84, 427-428

event triggers, 84

explained, 423-427

expressing logic with, 428

logical AND, 429-430

logical OR, 429

in control templates, 432-434

property triggers, 83-85, 424-427
 respecting visual states with, 442-446

Triggers collection, 85

Triggers property (FrameworkElement class), 85

troubleshooting

data binding, 384
 missing styles, 461

TryFindResource method, 359

tunneling, 161

turning off type conversion, 50

Twitter, pure-XAML Twitter client, 412-413

TwoWay binding, 403

type converters

BrushConverter, 32
 explained, 30-31
 finding, 32
 FontSizeConverter, 32
 GridLengthConverter, 131
 ImageSourceConverter, 309
 LengthConverter, 102
 in procedural code, 31
 ThicknessConverter, 102
 TransformConverter, 113
 turning off type conversion, 50
 values type-converted to object
 elements, 38

typed styles, 421-422

U

**UI Automation, supporting in custom controls,
 749-750**

UICulture element, 350

Uid directive, 351

UIElement class

binding to, 370
 explained, 74
 IsKeyboardFocused property, 170
 IsMouseDownDirectlyOver property, 171
 RenderTransformOrigin property, 107

UIElement3D class, 15, 588

ContainerUIElement3D, 590
 explained, 74
 ModelUIElement3D, 588-590

uniform scale, 557

UniformGrid, 138

**unmanaged code, mixing with managed
 code, 682**

UpdateLayout method, 100

**UpdateSourceExceptionFilter property (Binding
 object), 408**

UpdateSourceTrigger enumeration, 404-405

**UpdateSourceTrigger property (Binding
 object), 404**

UpDirection property (Cameras), 548-550

URIs

packageURI, 349
 URIs for accessing binary resources,
 346-347

usage context, 375

UseLayoutRounding property, 17

user controls, creating

behavior, 725-727
 dependency properties, 728-731
 explained, 721-722
 protecting controls from accidental usage,
 727-728
 routed events, 731-732
 user controls versus custom controls, 722
 user interfaces, 723-725

user interfaces

- creating for PlayingCard custom control, 739-742
- creating for user controls, 723-725
- marking with localization IDs, 351

USER subsystems, 10**V****ValidateValueCallback delegate, 89****validation rules, 405-409****ValidationRules property (Binding object), 406****value converters**

- Binding.DoNothing values, 385
- bridging incompatible data types, 381-384
- customizing data display, 385
- explained, 381
- temporarily canceling data binding, 385
- ValueMinMaxToLargeArcConverter, 445-446
- ValueMinMaxToPointConverter, 445-446

Value value (NodeType property), 57**ValueMinMaxToLargeArcConverter, 445-446****ValueMinMaxToPointConverter, 445-446****ValueSource structure, 88****variables, HwndSource, 697-698****verbosity of XAML, 71****versions of WPF**

- future releases, 17
- WPF 3.0, 14
- WPF 3.5, 14-16
- WPF 3.5 SP1, 15-16
- WPF 4, 14, 16-17
- WPF Toolkit, 14

VerticalAlignment property (FrameworkElement class), 103-105**video support**

- controlling underlying media, 661-662
- embedded resources, 663

explained, 658

MediaElement, 658-660

taking snapshots of individual video frames, 660

Windows Media Player, 658

VideoDrawing class, 476**Viewbox class, 144-147****Viewbox property (DrawingBrush class), 523-524****Viewport2DVisual3D class, 15, 590-591****Viewport3D class, 593-596****Viewport3DVisual class, 596****views**

- customizing collection views
 - creating new views, 394-396
 - explained, 386
 - filtering, 392
 - grouping, 388-391
 - navigating, 392-393
 - sorting, 386-388
- TreeView control, 302-304

viewSource_Filter method, 395**virtualization, 289, 296****VirtualizingPanel class, 120****VirtualizingStackPanel, 120, 279****Visibility property (FrameworkElement class), 102-103****Visible value (Visibility enumeration), 102****Visual C++, 681, 695****Visual class, 80**

explained, 74

TransformToAncestor method, 596-600

visual effects, 529-531**visual hit testing**

- callback methods, 505
- explained, 499
- simple hit testing, 499-500
- with multiple Visuals, 500-503
- with overlapping Visuals, 503-505

Visual State Manager (VSM), 17

- animations and
 - Button ControlTemplate with VisualStates, 643-646
 - transitions, 647-651
- respecting visual states with
 - control parts, 447-449
 - control states, 449-455

visual states

- respecting with triggers, 442-446
- respecting with VSM (Visual State Manager)
 - control parts, 447-449
 - control states, 449-455

Visual Studio debugger, 236**Visual Studio-like panes, creating**

- sequential states of user interface, 147-151
- VisualStudioLikePanes.xaml, 151-153
- VisualStudioLikePanes.xaml.cs, 153-157

Visual3Ds

- explained, 74, 586
- ModelVisual3D class, 587-588
- TransformToAncestor method, 600-605
- TransformToDescendant method, 600-605
- UIElement3D class, 588
 - ContainerUIElement3D, 590
 - ModelUIElement3D, 588-590

VisualBrush class, 525-527**VisualChildrenCount method, 497-498****Visuals**

- custom rendering, 499
- displaying on screen, 496-498
- DrawingContext methods, 494
- DrawingVisuals
 - explained, 493
 - filling with content, 493-496
- explained, 493

visual hit testing

- callback methods, 505
- explained, 499
- simple hit testing, 499-500
- with multiple Visuals, 500-503
- with overlapping Visuals, 503-505

VisualStateGroup class, 455**VisualStateManager. See Visual State Manager****VisualStudioLikePanes.xaml file, 151-153****VisualStudioLikePanes.xaml.cs file, 153-157****VisualTransition objects, 647-651****VisualTreeHelper class, 77****vshost32.exe, 236****VSM (Visual State Manager), 17**

- animations and
 - Button ControlTemplate with VisualStates, 643-646
 - transitions, 647-651
- respecting visual states with
 - control parts, 447-449
 - control states, 449-455

W**Webcam control (Win32)**

- HostingWin32.cpp file, 685-687
- Webcam.cpp file, 678-681
- Webcam.h file, 678
- Window1.h file, 683-684

Webcam.cpp file, 679-681**Webcam.h file, 678****whitespace, TextBlock control, 314****Width property (FrameworkElement class), 98-100****Win32 controls, WPF interoperability**

- explained, 677
- HwndSource class, 692-695

- keyboard navigation, 687-691
- launching modal dialogs, 692, 699
- layout, 696-699
- Webcam control, 678-687
- winding order (mesh), 579-580**
- Window class, 196-198**
- Window1.h file, 683**
- Window1.xaml file, 717**
- Window1.xaml.cs file, 716**
- WindowHostingVisual.cs file, 495-497**
- WindowInteropHelper class, 708**
- Windows 7 user interface features**
 - Aero Glass, 249-253
 - Jump Lists
 - and Visual Studio debugger, 236
 - associating with applications, 234
 - explained, 233-234
 - JumpPaths, 241-244
 - JumpTasks, 234-240
 - taskbar item customizations
 - explained, 245-246
 - taskbar item overlays, 247
 - taskbar item progress bars, 246
 - thumb buttons, 248-249
 - thumbnail content, 247
 - TaskDialogs, 253-256
 - WPF 4 support for, 16
- Windows applications**
 - multiple-document interface (MDI), 203
 - navigation-based Windows applications
 - explained, 211-212
 - hyperlinks, 215-216
 - journal, 216-218
 - Navigate method, 214-215
 - navigation containers, 212-214
 - navigation events, 218-219
 - Page elements, 212-214
 - returning data from pages, 221-222
 - sending data to pages, 220-221
 - single-instance applications, 204
 - standard Windows applications
 - Application class, 199-204
 - application state, 209-210
 - ClickOnce, 210-211
 - common dialogs, 206-207
 - custom dialogs, 207-208
 - explained, 195-196
 - multithreaded applications, 205
 - retrieving command-line arguments in, 202
 - splash screens, 205-206
 - Window class, 196-198
 - Windows Installer, 210
- Windows collection, 202**
- Windows Forms controls, WPF interoperability, 10**
 - converting between two representatives, 707-708
 - ElementHost class, 704-706
 - explained, 699-700
 - launching modal dialogs, 703, 708
 - PropertyGrid, 700-703
- Windows Installer, 210**
- Windows Media Player, 658**
- Windows themes, 470**
- Windows XP, WPF differences on, 18**
- WindowsFormsHost class, 702**
- WorkingDirectory property (JumpTask), 238**
- WPF 3.0, 14**
- WPF 3.5, 14-16**
- WPF 3.5 SP1, 15-16**
- WPF 4, 14, 16-17**
- WPF Toolkit, 14**
- WPF XAML Vocabulary Specification 2006 (MS-WPFV), 24**
- WrapPanel**
 - examples, 121
 - explained, 120

interaction with child layout properties,
121-122
properties, 120
and right-to-left environments, 121

WriteableBitmap class, 15

writers (XAML)

explained, 53-54
node loops, 56-57
writing to live objects, 61-63
writing to XML, 63-64
XamlServices class, 64-67

writing

easing functions, 640-642
validation rules, 406-407

X

X property

StylusPoint object, 175
TranslateTransform class, 112

x:Arguments keyword, 51, 67

x:Array keyword, 70

x:AsyncRecords keyword, 67

x:Boolean keyword, 67

x:Byte keyword, 67

x:Char keyword, 67

x:Class keyword, 45, 67

x:ClassAttributes keyword, 68

x:ClassModifier keyword, 68

x:Code keyword, 68

x:ConnectionId keyword, 68

x:Decimal keyword, 68

x:Double keyword, 68

x:FactoryMethod keyword, 51-52, 68

x:FieldModifier keyword, 68

x:Int16 keyword, 68

x:Int32 keyword, 68

x:Int64 keyword, 68

x:Key keyword, 68

x:Members keyword, 53, 68

x:Name keyword, 42, 68, 434

x:Null keyword, 70

x:Object keyword, 68

x:Property keyword, 53, 68

x:Reference keyword, 70, 703

x:Shared keyword, 69, 358

x:Single keyword, 69

x:Static keyword, 70

x:String keyword, 69

x:Subclass keyword, 69

x:SynchronousMode keyword, 69

x:TimeSpan keyword, 69

x:Type keyword, 70

x:TypeArguments keyword, 69

x:Uid keyword, 69

x:Uri keyword, 69

x:XData keyword, 69

XAML (Extensible Application Markup Language)

{ } escape sequence, 377

accessing binary resources from, 345-348

advantages of, 22-24

animation with EventTriggers/Storyboards

explained, 621-622

starting animations from property triggers, 628-629

Storyboards as Timelines, 629-630

TargetName property, 625-626

TargetProperty property, 622-625

BAML (Binary Application Markup Language)

decompiling back into XAML, 47-48

defined, 45

Binding object in, 365-367

- CAML (Compiled Application Markup Language), 46
- common complaints about, 70-71
- compiling, 43-45
- defined, 23-24
- embedding PropertyGrid with, 702-703
- explained, 12, 21-22
- extensibility, 39
- factoring, 357
- generated source code, 46
- keywords, 67-70
- loading and parsing at runtime, 40-42
- loose XAML pages, 231-232
- markup extensions
 - explained, 32-35
 - in procedural code, 35
 - parameters, 33
- namespaces
 - explained, 26-28
 - implicit .NET namespaces, 27
 - mapping, 26
- object elements
 - attributes, 25
 - content property, 35-36
 - declaring, 25
 - dictionaries, 37-38
 - explained, 24-26
 - lists, 36-37
 - naming, 42-43
 - processing child elements, 40
 - values type-converted to object elements, 38
- order of property and event processing, 26
- procedural code inside, 47
- property elements, 29-30
- pure-XAML Twitter client, 412-413
- readers
 - explained, 53-54
 - markup compatibility, 61
 - node loops, 56-57
 - NodeType property, 57-58
 - sample XAML content, 58-59
 - XAML node stream, 59-61
 - XamlServices class, 64-67
- running XAML examples, 22
- specifications, 24
- type converters
 - BrushConverter, 32
 - explained, 30-31
 - finding, 32
 - FontSizeConverter, 32
 - in procedural code, 31
 - values type-converted to object elements, 38
- writers
 - explained, 53-54
 - node loops, 56-57
 - writing to live objects, 61-63
 - writing to XML, 63-64
 - XamlServices class, 64-67
- XAML Browser Applications (XBAPs), 15
 - ClickOnce caching, 226
 - deployment, 229
 - explained, 224-226
 - full-trust XAML Browser applications, 228
 - integrated navigation, 228-229
 - limitations, 226-227
 - on-demand download, 230-231
 - security, 229
- XAML2009
 - built-in data types, 50
 - dictionary keys, 50
 - event handler flexibility, 52
 - explained, 48-49
 - full generics support, 49
 - object instantiation via factory methods, 51-52

object instantiation with non-default constructors, 51

properties, defining, 53

XAML Browser Applications (XBAPs)

ClickOnce caching, 226

deployment, 229

explained, 224-226

full-trust XAML Browser applications, 228

integrated navigation, 228-229

limitations, 226-227

on-demand download, 230-231

security, 229

XAML Cruncher, 23

XAML Object Mapping Specification 2006 (MS-XAML), 24

XAML2009

built-in data types, 50

dictionary keys, 50

event handler flexibility, 52

explained, 48-49

full generics support, 49

object instantiation via factory methods, 51-52

object instantiation with non-default constructors, 51

properties, defining, 53

XamlBackgroundReader class, 53

XamlMember class, 58

XamlObjectReader class, 53

XamlObjectWriter class, 54

XamlObjectWriterSettings.

PreferUnconvertedDictionaryKeys property, 50

XamlPad, 23

XAML PAD2009, 22-23

XamlPadX, 23, 77

XamlReader class

explained, 53-54

Load method, 40-41

LoadAsync method, 41

XamlServices class, 64-67

XamlType class, 58

XamlWriter class, 48, 53-54

XamlXmlReader class, 53-56

markup compatibility, 61

sample XAML content, 58-59

XAML node stream, 59-61

XamlXmlWriter class, 54

XBAPs. See XAML Browser Applications

XML, writing to, 63-64

XML Paper Specification (XPS), 319

XML Path Language (XPath), 397

xml:lang attribute, 67

xml:space attribute, 67

XmlDataProvider class, 397-401

XNA Framework, 11

XPath (XML Path Language), 397

XPS (XML Paper Specification), 319

Y-Z

Y property

StylusPoint object, 175

TranslateTransform class, 112

Z order, 117-118

Z-fighting, 545

zooming

enabling with multi-touch events, 182-183

with inertia, 184-185