Brennon Williams

Foreword by Steve White
Program Manager, Microsoft Expression Blend

Microsoft® Expression Blend™

UNLEASHED

SAMS

Microsoft® Expression Blend™ Unleashed

Copyright © 2008 by Pearson Education, Inc.

ISBN-13: 978-0-672-32931-9

ISBN-10: 0-672-32931-X

Library of Congress Cataloging-in-Publication Data available upon request.

Printed in the United States of America

First Printing June 2008

Trademarks

Warning and Disclaimer

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@pearson.com

Editor-in-Chief
Karen Gettman

Executive Editor
Neil Rowe

Development Editor
Mark Renfrow

Technical Editors
Amir Khella
Chad Carter

Managing Editor
Kristy Hart

Project Editor
Anne Goebel

Copy Editors
Harrison Ridge Services
Mike Henry

Indexer
Publishing Works Inc.

Proofreader
Linda Seifert

Publishing Coordinator
Cindy Teeters

Cover Designer
Gary Adair

Composition
Jake McFarland

Foreword

You're a user of things—we all are. You have an experience with every *thing* you interact with. And, it better be a *good* experience, too; otherwise, you'll change the thing, avoid interacting with it, or suffer.

It could be that basking in the color of your bedroom walls is a therapeutic experience. Conversely, you might have a frustrating experience with a hammer whose head keeps falling off (or which, to keep you and those around you safe, asks you to confirm every blow). On the other hand, an object may be perfectly well-designed, but not necessarily designed for you. Even if you could fit in it, a child's chair would be too short for you. As anyone who has seen the movie *The Forbidden Planet* will know, Krell-sized doorways would be a needlessly extravagant use of space in a human home! In each of the previous examples, the experience depends, to some degree, on the person doing the experiencing.

Let's talk about software—the kind of software *you* create. That's why you have this book in your hands, right? Good software is concerned with the emotions of the person using it. It can excite the user from time to time, but it should never frustrate the consumer. When it's working effectively, good software's unobtrusive usability brings only a faint smile of satisfaction to its user's lips, not grumbles. The value of software is the value of the experience that flows from it.

This book is for anyone interested in designing usable and beautiful software. It's about a framework for helping you do just that. It's about a platform and toolset that do the heavy lifting so you can focus on the art, the usability, the experience. You'll learn how it's possible to separate the tasks done by the designer and developer roles so that you can work in an independent, yet complementary way. You'll see how you can avoid a potentially lousy translation step from comp to user interface. You'll see how to customize controls, bind to data, create artwork with vectors and brushes, and represent user interface and data in declarative markup that is kept separate from the application logic.

WPF, Silverlight, and Expression are instruments: Artists create with them. The tools have a user experience of their own—and that experience will improve as they mature. Expression Blend, for instance, is suited to designers with esthetic talent, to be sure—but those with a taste for a little technology will find a lot of additional possibilities. Soon, Blend will adapt to an even greater diversity among designers. Making that happen is the daily preoccupation of the development, test, program management, technical writing, product design, product development, support, and evangelism members of the Blend product group.

Jimi Hendrix asked: “Are you experienced? “A Hendrix album or concert *was* an experience that went beyond the everyday to provide astonishing sounds, virtuosity, charisma, psychedelia and theatrics. What makes *your* consumers enthusiastic about—and return to—your software and design is the quality and nature of their experience. For the most part the experience you provide is dependent on your talent and imagination. However, *some* of your success is due to the tools you use and how well you know them.

I hope you enjoy using WPF and Expression. I hope they give you what you need and make you successful. You can communicate with the Blend product group at http://expression.microsoft.com.

In the meantime, there’s a lot for you to learn. I’ll let Brennon tell you the rest!

Steve White
Program Manager, Microsoft Expression Blend

CHAPTER 1

Introduction to Expression Blend

IN THIS CHAPTER

- The Next Generation User Experience
- Windows Presentation Foundation (WPF)
- Expression Blend Versus Visual Studio
- An Introduction to Expression Blend and XAML
- The Benefits of Expression Blend

Taking aside the technical aspects of learning how to use Microsoft Expression Blend, there are many areas that are often disregarded within discussions about how Blend is—and should be—used in a real-world project sense. You may be a single person business or employee that needs to fill all the roles that Blend is best used for; or, you may be part of an enterprise size team.

Knowing, why a tool should be used in a certain way sometimes makes it easier to apply the options provided. This chapter talks about areas within project teams, their roles, and their perspective.

The Next Generation User Experience

One of my favorite pastimes is watching movies. I don't care how old they are or if I have already watched them 10 times. I find that watching movies relaxes me; it allows me to escape my own thoughts for an hour or two, and most of the time they inspire and motivate me in some way.

I especially enjoy watching movies in which computers are used in the plot in some way. I guess that is the geek coming out in me, but there is something special about the user interfaces, and in some spectacular cases the visual and interactive design, that are also explored in such movies.

Minority Report, with Tom Cruise, is a great example of what we want to be moving toward in providing an outstanding user experience and interface. That interface (and indeed the entire process that encompasses Tom's user experience) is custom made for its application in assisting the user to scrub video feeds and visualize complex data. Without having actually used such an interface, we can only assume that it is ideal because of the efficiency it gave Tom's character in performing his job's functional requirements.

The applications functional playback of simultaneous multimedia streams gave Tom's character the ability to visualize and analyze complex data, but it is the user interface and "light glove" device that provided the cohesive, relevant, and intuitive movements; allowing the application to work with him, flow with him, and ultimately deliver for him. The input hardware is not quite ready yet, but by the time we collectively perfect our interfaces, we should start to see additional hardware flow into the market place.

Next Generation Hardware Is Coming!

Already hardware changes are being made to accommodate Vista-specific features like Windows Sideshow. This technology allows next generation laptops (code named Newport) to show a second, smaller screen on the laptop lid. This system runs even while the main laptop power has ceased, providing email, calendar and a whole host of other features. However, it appears that the implementation has been slow to gather momentum in this particular case.

Will Vista deliver this? Or more accurately, will Vista allow us, as software designers and developers, to deliver perfect user interfaces? Gartner Inc. analyst Michael Silver believes that the year 2009 should see a noted shift in worldwide Vista installations, already predicting it to eclipse Windows XP. According to Microsoft (Securities and Exchange Commission Microsoft Corporation Quarterly report #0-14278), an estimated 80% of all new PC-based computers sold will be pre-installed with Vista and a steady upgrading through business will occur as the security models improve and business requirements for security of personal data increases.

Microsoft predicted (or just told anyone who would listen): Vista will be the quickest adopted operating system in history. I'm not so sure that this adoption wave is the reality at the moment; however, designing and developing better applications through improved user experience is the one thing that could push users into upgrading.

All the predictions are based on the maturity of the Internet as an information delivery mechanism. If you observed the Internet and the changes that occurred when dynamic rich content first began showing itself, you would have seen how companies were able to expand their business offerings to online customers in areas that were previously thought to be too complicated for the average nontechnical user to follow. Suddenly, holiday booking engines and online flight ordering businesses were able to provide live pricing data that enabled potential customers to shop for the best deal and a myriad of other business sectors could engage their current and potential customers by explaining their processes more clearly (visually as opposed to verbally by a sales person on a telephone) and in a nontechnical way.

From a user experience point of view, given the choice between two websites, even the most nontechnical users will opt for the more professional looking website, because subconsciously they believe that if a company has put great efforts into making things easier to view, the company must be more professional in their approach and will almost certainly treat personal information and security details with the same high quality of care. Of course, this is not always the case, and is a reason phishing attacks are so successful—but that's another story.

This does, indeed, prove the case that the best product with the most features does not always guarantee success for a business. You need a door opener, a way of first attracting the customer. Then you can educate them on the features and benefits of a given product or service. The main thing is that you got them interested in your offerings first, before that of your competition.

So, why would people continue to use desktop applications over web-based applications? The features and abilities of any given web or desktop application are directly linked to the code accessibility permissions of the client. The web will continue to be a delivery mechanism for partially trusted applications, whereas the desktop will continue to provide unrestricted high speed features, unbound by prohibitively expensive bandwidth limitations that still exist in countries like Australia and Ireland, to name two.

Where Is Your Stuff?

The level of trust the proponents of online applications place in the companies providing such services is interesting. Take any online application, such as a word editor, that also saves your work online. Do you really know where your material is? What happens if the company goes broke? Even the biggest companies go broke—anyone remember Enron? Don't be fooled into thinking it can't happen again!

How This Book Will Help You

The use of Blend in real-world environments has been a topic that not many people have wanted to discuss. Maybe that's not true, but it is difficult to find others that think about it in depth. On the surface, the discussions have been all about the nice flashy demonstrations, the eye candy side of the promise that Blend brings—the Kool-Aid—as one of my U.S. associates refers to it. The promise I am talking about is designer/developer collaboration.

It's one thing for people, even Microsoft, to talk about designer/developer collaboration, but actually making it work in the real world can be quite difficult. I thought that, aside from helping people (both designers and developers) to use Blend, I might also be able to provide some instruction in using the tool(s) so that others don't get the headaches I did when implementing Expression Blend in projects for the first time. You will still get some headaches—that's just a given!

The Tools

When I talk about the tools, I am talking about the classification of tools that I see as being necessary to implement the collaboration promise for creating User Interfaces.

- **Expression Design** or other graphic design packages that are capable of exporting Extensible Application Markup Language (XAML, pronounced "zamel")
- **Visual Studio** for solution architecture and .NET code requirements
- **Expression Blend** for the interactive designers and XAML architects

These three tools (along with the right workflow), not just one tool by itself, allow this promise to be delivered. Of course, people have their own preferences in their choice of tools, so feel free to insert your favorites. This tool package, however, is the focus of this book.

The best advice is to embrace as many designer/developer tools as you can, and then decide which ones are good and which ones are not. As a result of trying each tool, you will know which are best suited for a particular purpose when it comes time to deliver.

Expression Blend is a designer's tool because it has no true development language within it (it has a script but not a language). This does not mean that only designers should use it. Yes, you use .NET code in the form of C# and VB.NET to create the backend functionality for your controls and objects, but inside Blend it's all about XAML. The Blend UI serves as a visual aid in creating XAML scripts.

Before going any further, it is important to clarify the term *designer.* It is just too broad to be used often when discussing Expression Blend. There are many types of designers: interactive, graphic, and industrial to name a few. These individuals may see the term in relation to what they know of the product and think that Blend really isn't aimed at them at all. Blend can be used by most, if not all, classes of designers. However, it is most suited to the interactive designer—the designer who creates the workflow and user processes that are intended to create an experience.

Blend certainly provides several excellent services in real-world projects:

- **Prototyping**—By creating a functional prototype using Expression Blend, properly constructed UI solutions are available to the production cycle immediately, providing a massive productivity gain and reducing development lifecycles. Prototyping also removes an additional layer of interpretation that is traditionally performed by developers that can radically change the end user's experience if incorrectly implemented.

- **UI process development**—User interface processes can mean the difference between high and low application productivity. Involving potential end users in the design and development process is always advisable. Being able to modify the interface to suit the users' requirements with Blend is certainly a benefit to the end solution. Previously a designer may have needed to report end user feedback and changes that were needed to a developer. At that point, the whole "interpretation" issue was raised again.

- **Animation production**—Blend provides a simple, easy-to-use storyboarding mechanism for animating object properties that allows designers to control motion and workflow while, at the same time, giving developers a simple reference to execute when the specific animation is required.

- **Visual asset creation**—While not a primary role for Expression Blend, the application is stocked with enough tools to allow graphic designers to create exceptional visual assets. Using the built-in resource management system also allows those assets to be shared and reused with minimal work, increasing productivity and ensuring visual continuity within the solution.
- **Resource file management**—Expression Blend allows you to create, modify, and utilize templates and resources stored in the working files, the application, and/or a XAML file called a *resource dictionary*. This means that one designer could create a certain style or a specific functionality of a certain object, a button for example, and then provide that same group of styles and functionality to everyone else working on the solution. This is perhaps one of the most powerful features of Expression Blend.
- **XAML architecting**—This is how Blend is used in a real-world project—a role to which Blend is particularly well-suited. XAML architecting is about taking all the assets from both designers and developers and putting them together. Visual Studio plays a big part in this process as well; but, without Blend combining assets would be a very difficult job indeed.

Why the Architect Title?

By definition, an architect is someone who creates a solution. That is a pretty broad description. In software terms, object structure and communication are not the only parts of a solution that need to be architected; interaction and workflow should also be designed. Because there was no term describing the role that constructs those parts of the solution, I created one—XAML architect.

Of all the roles previously mentioned, the XAML architect (XA) is the role which embodies the prescribed collaborative process—the best in a real-world scenario.

The day-to-day process surrounding the XA is asset designers mocking up and creating beautiful pieces of art and coders implementing required functionality that brings said art to life. If these designers don't understand .NET development and the coders have no idea about animation, then without the XAML architect to bring their contributions together in a meaningful way, no collaboration can exist. Neither the designer nor the developer can understand or appreciate the requirements of the solution as a whole.

It is the job of the XAML architect (XA) to own and control the design and development of all XAML-based assets. The XA understands the creative vision of the graphic designer, the user experience and workflow concepts of the interactive designers, and how to implement functionality in .NET code to ensure animations, binding, and other pieces of the functionality work with the user interface (UI) specification and the user experience (UE) expectation.

This is where the power of Blend can be exploited to its capacity. The name says it all, really. As an XA, you blend the code and visual assets to produce the UI and UE.

This blending of design and development is the most important concept you should take away from this book.

The great scientist Louis Pasteur once said, “Chance favors the prepared mind.” That means: If you expand your knowledge and gain experience that covers all the angles, then there is no such thing as luck—just opportunities. While a Zen Monk may disagree with this interpretation, no one could argue that keeping an open mind in today’s world isn’t a good approach.

Now is the time to gather experience, understand the potential, and prepare for the opportunities. This book is not going to give you all the answers, but it should make sure your head is screwed on the right way before you start your journey. I have tried to change the dialog you see in most technical books, because I want you to think of the two of us sitting in a pub somewhere having a chat over something cold, may be a beer, whatever suits your fancy!

The chapters are in the order in which a non-developer (but still technically astute) designer should be learning the skills necessary to use both Expression Blend and Visual Studio. There is no escape from Visual Studio if you want to use Expression Blend for anything more complex than creating XAML-based visual assets. Another reason for the chapter sequence is that there were moments during my initial experiences with Expression Blend and XAML that I wish I knew some things earlier. This chapter sequence addresses the concerns I had.

Unlike the rest of the book, this first chapter is one big rant. If you continue to read it, you will understand why it is important for you to grasp the new technology and how you might begin to change your working vision to become one of the new breed of XAML architects that is required to make the technology work successfully in enterprise level development. Of course, you may just want to continue to be a designer of a certain discipline or a developer—that’s fine. I have tried to cater to diverse needs.

If you are just itching to get into the code, please be patient. Shortly you will be up to your neck in code and markup, but you will understand why you are doing certain things, and more importantly, what you are doing. You will understand the mindset of others in your working arena and should then be ready to climb this very steep learning curve.

Above all, remember: We are dealing with interfaces, user experiences, coding concepts, architecture, and how to bring this all together to capture, amaze, and empower your end users.

The Business Mindset of the End User

Think back to *Minority Report* for a moment. Visually speaking, those UI’s are amazing. When you think about it, however, we are not too far away from that right now.

In my experience over the last 15 years working in several countries on projects of all sizes and budgets, the single biggest issue facing the development of these UIs in mainstream applications is a time/cost ratio that rarely has enough time or money assigned to the commercial development lifecycle. The one exception to this rule is the game/entertainment industry. They understand the importance of visual perception.

Visual perception is the awesome use of a designer's imagination, where he or she envisions user interfaces that are tailor-made to provide an immersive, state of the art navigation system that engages users so that they are excited about what they are doing. Although these navigation menus often don't provide any more functionality than could be achieved with the good old File menu ribbon system that is available in simple applications like Notepad.exe, the users' visual perceptions are that the interface is more engaging. These UI's actually draw upon the users emotional pool.

Most of the bigger companies are now starting to pick up on the fact that if you engage your user, make him or her feel good about what they are doing, the user's productivity goes through the roof. Regardless of whether another product has more features, higher productivity outweighs the lost feature benefits. High productivity is like a virus. If you walk into a room and everyone is smiling, you will naturally feel better. Similarly, in an office environment, if some staff are productive and excited, more people tend to get onboard and feel productive and excited. This is a huge benefit to business.

Companies like Microsoft have always tended to place particular importance on the standardization of interfaces. Using certain user interface items such as the *Ribbon* in Office products even have strict guidelines that must be followed to comply with the end user licensing. For most there is a Vista User Experience Guideline document that provides for requirements on such things as menus, control layouts, accessibility, and more. The logic here was that people would be more inclined to want to learn and continue to use an application if it had a familiar feel. That logic is still partly true today.

Familiarity is a strange sense. Just the other day I was helping a relative discover the joys of modern day computing. He happened to mention to me that the staff in his office use Classic settings in Windows XP. He had no idea what this meant. It was just something he heard them say once—it is their preference, or simply what they are comfortable using and feel most familiar with.

As an experiment, I showed him the standard Windows XP interface and explained where things were. Then I switched the view to Classic view. His response was that it looked dull, didn't really give him much in the way of the Start Menu, and that he would find it much easier to remember the pictures (he meant icons) as opposed to just the names of items in the control panel application, for example.

In the 21st century, the general population in most developed and technically advanced countries is becoming more tech savvy than ever before. People do not need to be coddled and can understand that moving a mouse and clicking on a button has repercussions. They don't need a paper clip (see Figure 1.1) telling them they have just clicked the button!

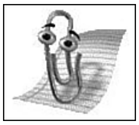


FIGURE 1.1 Just in case therapy helped you to forget!

People are going to ask the question “Why should I upgrade to Vista if it doesn’t give me anything new?” I respond: It is not about getting anything new. Vista and the software designed for it help users work smarter, faster, and more efficiently by providing a framework of *smart* common controls and a more visually intuitive user interface. Together these elements ensure that all applications provide a familiarity while at the same time making the responses to choices that the end user makes much clearer.

It is certainly fair to say that while technology has continued to improve and advance, the opportunity to improve user experiences in software has not kept pace until now. Compared to the Internet, which has seen a huge rise in back-end technology as well as user interface design tools and standards in the last decade alone, application user experiences are far behind. Think of all the new languages that have come along to facilitate web applications: the more frequent use of Ajax to improve user experience and perhaps the biggest advance—the introduction of Adobe Flash and the shockwave format (.swf).

This improvement in technology is now affecting desktop applications, through user interface design. The power of the desktop application will always surpass that of the web application, primarily due to security; but, until now all of the desktop applications have been a tad boring and deployment has been nothing short of a nightmare. In the coming years ever more importance will be placed on the user experience, which is directly linked to the quality of the user interfaces that we will be building for our end users.

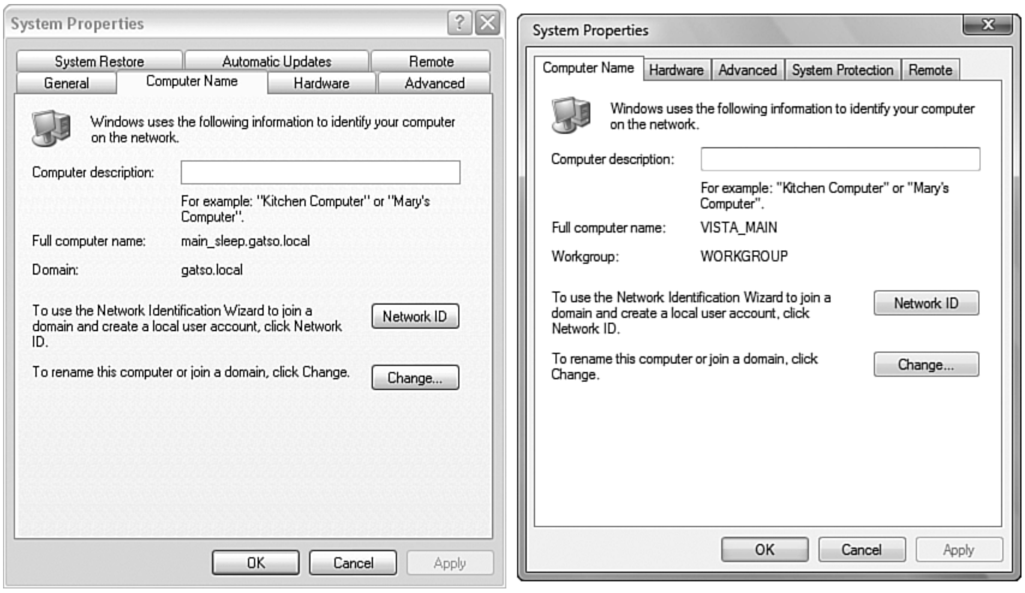
Vista and .NET Framework 3.0/3.5 (Formerly WinFX)

“Where is my *Minority Report*?” I hear you say. Well like Tom, you do not have a *Minority Report*. You can be 100% sure of what your future holds (hopefully not murder!) but it will contain Vista, high definition (HD) and high fidelity content, glass and gel interface objects, Expression tools and .NET programming languages. Microsoft is betting the house on it, so to speak, and they usually get what they want—eventually.

There are other companies out there producing products for designers and developers that are XAML based, like Aurora by Mobiform Software Inc. Regardless of what advanced features they have implemented, I can tell you that Microsoft are working a few years ahead so Expression Blend and their other products should ultimately work more efficiently with other integrating technologies that will become fulfilled with Vista, and even further ahead with Windows 7 and beyond.

If you have used Microsoft Vista, you will have noticed how Microsoft appears to have finally gotten an outstanding balance with their product with respect to familiarity, visual appearance, and general feel. As you go deeper into Vista, you will see that some of the dialogs have much the same content and layout as they did in Windows 2000 and XP. Figure 1.2 illustrates the similarity. There are slight differences in the user workflow in some cases, but the visual continuity is always the same throughout the operating system.

Vista also brings an outstanding addition to the operating system control base, the use of Bread Crumb controls. These controls have several features that make the task of navigating more flexible and more intuitive. Let’s take a look at the file explorer application window (see Figure 1.3).



System Properties

System Restore | Automatic Updates | Remote
General | Computer Name | Hardware | Advanced

Windows uses the following information to identify your computer on the network.

Computer description:

For example: "Kitchen Computer" or "Mary's Computer".

Full computer name: main_sleep.gatso.local

Domain: gatso.local

To use the Network Identification Wizard to join a domain and create a local user account, click Network ID. [Network ID]

To rename this computer or join a domain, click Change. [Change...]

OK | Cancel | Apply

System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

Windows uses the following information to identify your computer on the network.

Computer description:

For example: "Kitchen Computer" or "Mary's Computer".

Full computer name: VISTA_MAIN

Workgroup: WORKGROUP

To use the Network Identification Wizard to join a domain and create a local user account, click Network ID. [Network ID]

To rename this computer or join a domain, click Change. [Change...]

OK | Cancel | Apply

FIGURE 1.2 The Windows XP and Vista Computer Properties applications.

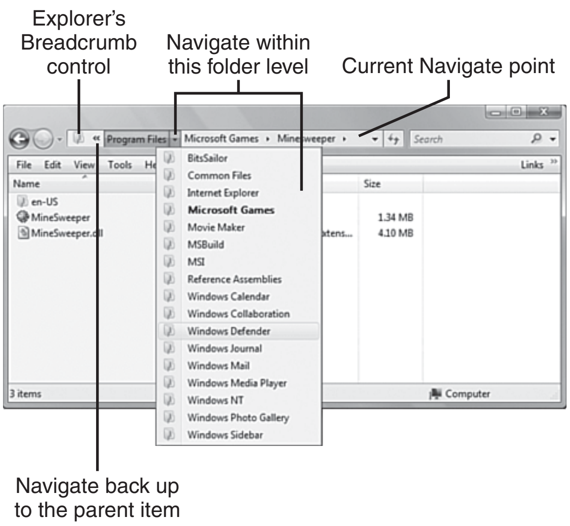


FIGURE 1.3 The Vista File Explorer with Bread Crumb control.

You can now navigate from within any hierarchical level along your path instead of getting repetitive strain injury (RSI) hitting the back button way back on the first level. You can enter a path directly as you have always been able to do, complete with history. You will also get visual feedback as to the status of your navigation from the control being used as a progress bar in the background. That is most handy when navigating large file sets or across really slow networks.

There are lots of controls like Bread Crumbs and Navigation Services that have been added to Vista. The important thing to realize is that better use of both graphics and functionality has improved the application and user experience tenfold. The Vista navigation elements are shown in Figure 1.4.

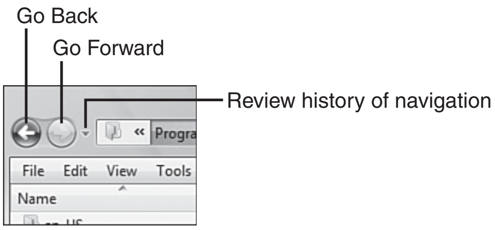


FIGURE 1.4 Vista application Navigation buttons.

The Windows Presentation Foundation (WPF), part of the .NET Framework 3.0/3.5, provides us with hardware-accelerated graphics power for standard applications that have previously been available only to DirectX and OpenGL applications. Integrating DirectX into a standard desktop application certainly has some great benefits, if not interesting results, but the development time is enormous—not to mention the performance issues designers and developers face when dealing with such a huge variation of end user machine specifications.

Increased visual performance through hardware-accelerated graphic pipelines includes more than just getting glass window borders in your applications. It includes the ability to apply transforms and animations to objects and use automated layout and true scaling with vector images that don't decay with size. By shifting all this graphic processing to the GPU, a huge load is removed from the CPU which, in itself, gives rise to massive performance increases.

A Developer's Note: Vector Graphic Versus Bitmaps

The difference between vector and bitmap graphics is the formats they comprise. A bitmap graphic contains thousands, sometimes millions of pixels, which represent the actual image. When scaling, the number of pixels is either decreased for reducing or increased for enlarging the image. The problems come into play when enlarging an image beyond its original size. The computer essentially has to guess what pixels should contain what color because when an image is increased in size, the pixels do not enlarge; more are merely added to the image to fill the required space.

Vector graphics, on the other hand, are instructions on how lines, points, and curves should be drawn to form complex shapes, rather than what these shapes should look like. The advantages are that the image can be increased in size or scaled without loss of quality. The disadvantage, however, is that while vector shapes can be complex, they are no match when it comes to producing an image with high-quality photo realism, which is best suited to the bitmap.

The key to all this interactive magic is XAML (the Markup script-language supported in Expression Blend) and its integration with the development language base of the .NET Framework 3.0 and 3.5. Visual Studio 2005 with extensions from the product code named Cider was a preview of Visual Studio 2008, which is now available. This tool is home to developers, allowing them to modify XAML as well as UI elements within a design surface.

XAML is the only reason designers and developers—and you as an XA—will be able to work together as a team . The XAML markup is the common component that binds all parties together, the glue, if you will.

So, why use Expression Blend if Visual Studio contains a design environment? The design environment in Visual Studio is simple at best at this stage. It contains no methods for applying animations through a timeline interface, nor does it have a control template editor. In other words, it is a limited design interface made for developers and not for animators or designers—and certainly not for an XA who needs a clear understanding of the end user workflow. Conversely, Blend is a designer's environment that switches between both XAML markup and .NET code in Visual Studio to allow you to apply CLR coding requirements. Both Visual Studio (or the .NET Framework to be more accurate) and Expression Blend share common UIElements. That is why we want to look at Expression Blend as an XA's tool. Understanding UIElements visually is far easier than looking at a heap of XAML markup and trying to picture it in your head, although in time you will be able to do this as well.

What I am trying to say is that with these new tools and pseudo languages you will be able to quickly create a "Minority Report" type of interface for Vista or XP. You, most likely, already have the majority of the core skills required. I am going to show you how to use those skills to give you a head start for getting into the new technology.

Getting your hands dirty is the only real way of understanding and perfecting your use of XAML and integrating it with the .NET languages. We will use only C# in this book because it is the language used most commonly for this purpose at the time of writing.

Does It Really Matter What You Use?

There is always a lot of conjecture about which language is best: C#, VB.NET, C++, or any of the other .NET-compliant languages. The fact is that all these languages still compile to almost identical MSIL code. Performance differences between them are negligible. Learn, at the very least, both C# and VB.NET to a professional level. This will increase your value as well as increase the number of jobs you could be qualified for when time comes to move on.

You do not have to be a coding guru to get great results from WPF—but it always helps. I am going to assume that you are a beginner programmer or designer. It would be helpful if you understand some of the core functionality provided by the .NET Framework within client application development terms. Don't worry if you haven't opened Visual Studio before, though. Chapter 10, "Visual Studio: C# Primer," provides a designer's view of coding to help you on your way.

If you are a developer, note that I am not going to try to convert you into a designer (which is more of a natural skill). Conversely, I am not trying to magically turn designers into developers. Instead, this chapter is intended to show you a little of how the other half lives, so to speak, so you understand and can liaise with teammates with other skill sets. The first half of the book is more oriented toward designers, focusing on Blend and XAML. The second half is more focused on developers, bringing in the concepts of the .NET coding requirements that will greatly extend the functionality of your applications.

Windows Presentation Foundation (WPF)

Windows Presentation Foundation (formerly code named Avalon) is a collection of display technologies—or a display level subsystem—that allows developers to take advantage of the latest graphic card hardware acceleration features. WPF is the father of XAML; it is what allows the XAML language to be used in a declarative way. The term declarative means “to describe” so XAML allows you to describe your applications UI functionality and components.

Through learning Expression Blend and a little .NET code, you will have the ability to compile your solutions into either a Silverlight or desktop deliverable. Technically, there are slight differences to some of the methods you use, but for the most part you will understand how both Silverlight and executable-based WPF applications are created.

Silverlight enables your applications to be viewed in a browser-hosted environment like Internet Explorer on any operating system that has the Silverlight plug-in installed. This gives the user the same experience they would get if they were using your application from a compiled .exe file running on a Windows desktop—or so the theory goes.
There is some reduced functionality in Silverlight, such as the ability to create hardware-accelerated 3D visuals for example, so care must be taken to understand the construct you are ultimately deploying to and the users you are working for.

Microsoft describes WPF as *“a unified API, allowing developers to present high definition media content from within their application constructs, as well as providing extensibility to the .NET Framework for Vista specific technologies.”* This, in reality, means that the best of both WINForms applications and web applications are available to you in WPF applications running on either Windows Vista or Windows XP (with Service Pack 2 and the .NET Framework 3.0/3.5 installed).

It has long been a nightmare for the UI Developer to ensure that content and visual assets were being managed in terms of layout. Before a generic entry level set of graphic cards (8Mb) came along, changing screen resolutions and color settings all pointed to early retirement for a vast number of us. Things have been getting better, certainly since .NET came along. Now WPF has once and for all sorted out such issues; never before has it been so easy to create an application that can adapt its content layout entirely based on the runtime environment.

I am writing this book on a laptop with a 17" widescreen display running a resolution of 1920 × 1200. Even though it is one of the most powerful laptops on the market, some of the biggest companies in the world have managed to produce software that does not layout correctly on it. One of the biggest joys I have found in creating applications with Blend is that the layouts work—perfectly every time.

It may take a little while to get your head around using Blend because most of the time you no longer need to specify a control's width and height. You are now working with dynamic flow and elements layout and positioning that is based on relevance to its parent and the construct it (the parent) provides. You can still specify width and height, but in most cases this will be a minimum width and a minimum height value. See Chapter 5, "UIElement: Control Embedding" for detailed information.

Silverlight

Microsoft unveiled the plans for Silverlight in the second quarter of 2007. Along with Silverlight comes the promise of a cut down, cross platform CLR plug-in that would make creating and distributing XAML-based applications across differing operating systems and devices simple. At the time of writing, Microsoft has just released a preview version of Expression Blend 2.5, which also showed the intended functionality that Blend will be providing in future for this new technology.

Silverlight delivers a subset of the WPF core and will allow loose XAML-based applications to run as pseudo web-based application or WBA. You can write the functionality for your application in your choice of .NET-compliant languages and/or JavaScript. So, Silverlight applications differ completely from another type of Internet-delivered XAML application called an XBAP.

XBAP applications run as a compiled application in a browser that supports additional features such as 3D, where Silverlight applications don't. There are many more differences that will change before the final incarnation of both these technologies are ratified.

In either scenario, though, you must consider that when your application (Silverlight or XBAP) runs in a browser type environment, it is running in what is known as a partial trust sandbox. In geek speak, that means the application is not installed on the end user's machine (although the plugins and other delivery mechanisms are).

Partial trust has some rules that you must follow. It is essential that you be aware of your application requirements before deciding to try and deploy your application in this manner, in either format. The rules are simple rules that govern whether an application is able to run based on the user authentication level and assigned roles or code privilege that is required by certain tasks, controls, and objects.

The essence of the partial trust sandbox is that it has been developed using the same security model that is available to all .NET developers using Code Access Security (CAS). This means that varying levels of an "application code" can demand and must obtain the correct security privileges before that code can execute.

In the XBAP deployment scenario, you will find tight controls on File read and write permissions and registry actions, as well as the inability to call and execute most areas of unmanaged code. From an application-specific point of view, you can't launch new windows from your application. You can't have application-defined dialogs present; you can't even apply BitmapEffects within your application. However, at the time of writing, the biggest concern is that you can't access all the features of Windows Communication Foundation Web Services. In other words, you are still restricted to only using data from within the same executing domain. Undoubtedly, Microsoft is working very hard to change these restrictions, and you should always seek the latest information from Microsoft with regards to such restrictions.

Exceptions to the Rules

There are exceptions for some forms of data, like video, for example. Your application will be able to display a video file hosted in another domain, but your application will not be able to access the raw data from the file that is playing.

Silverlight allows you to deliver streaming video content very efficiently through the browser. Expression Media Encoder assists you in packaging your video for streaming and creating Silverlight templates that are ready to go.

What Are BitmapEffects?

BitmapEffects allow you to apply effects to resources used in WPF applications such as Shadows or Ripples, similar to the types of effects you get with graphic applications like Adobe Photoshop. There are several effects that come as standard; and you can even write your own, although this is very convoluted at present. Companies such as Atalasoft are already producing libraries that you will be able to import and use. Technically though, at present BitmapEffects use software rendering in Windows; so they don't use the power of the graphics card, which will inhibit the performance of your application. The greatest concern, though, is the performance hit they place on Expression Blend in the design environment. It's your choice to use BitmapEffects or not.

Microsoft has indicated that hardware accelerated BitmapEffects will be available for the release of .NET Framework 4.0 possibly in late 2008 or early 2009.

I have described what has been slated at the time of this writing. Be sure to check with Microsoft to see if any of the restrictions have been lifted, or if more have been added. It's all about security, so it is important to understand it.

There are still many areas within an application that can run without issue, so don't be put off by the restrictions. Instead, look at it as a greater challenge to your initiative. After all, you can still incorporate 2D, 3D, and full animations in XBAPs, 2D in Silverlight, and image and audio features, flow documents and much more in XAML-based offerings—not to mention that your controls are no longer the clunky old CLR based controls.

Silverlight is exciting and it's fast. By learning Expression Blend and a little .NET development you will be able to create rich Internet applications (RIAs) that will become a standard in future development scenarios.

Expression Blend Versus Visual Studio

Expression Blend and Visual Studio are not really in competition with each other; instead, they are being developed to work with each other. Think of Blend being the intermediary between a designer-specific application like Expression Design or Adobe Illustrator and Visual Studio.

There is much rhetoric as to the potential for Blend being slowly merged into the Visual Studio environment. I would see this integration as a blow to the workflow that will exist between designers and developers if you start forcing designers to try and navigate the Visual Studio IDE. If you implement the role of a XAML architect, there is no need for a designer to ever worry about Visual Studio, and a developer needs only to concentrate on data structures and logic.

In the end, I believe Microsoft will see the value of the two products as they stand alone but in support of each other—and as the only tools fit for the purpose of the XA's role. Only time will tell, but for now, using both products simultaneously is workable, although a little clunky.

Is the XA a New Role?

At the time of this writing, the XA role was an area very few people understood, even those at Microsoft. I did find a few people that understood and had come to this conclusion themselves. Two such people are Darren McCormick and Jon Harris of Microsoft, both User Experience Evangelists. Both could see a clear need for the role to be defined and promoted.

It's not really a new concept. Most large design agencies have, for quite some time now, implemented such a role with projects involving Flash, in which designers were not always competent in Action Scripting.

There are indeed pros and cons to both environments. As it stands, I, personally, could not spend any great length of time trying to design an application experience in Visual Studio. At the end of this book you will make your own decision on how you like it.

Visual Studio does not provide for some of the functionality that Blend provides, such as an easy to use storyboarding tool to create animations and define triggers. Blend also handles data binding elegantly, which is a very important area of most applications. One of the biggest pros for Blend is that it will always give you an accurate depiction of the XAML (and code to some degree) live in the design-time environment, something that Visual Studio (in WPF solutions) continues to struggle with.

An Introduction to Expression Blend and XAML

If you believe that Expression Blend is an application best suited for a role like an XA, then you also understand and appreciate that its goals are to allow the rapid creation of user interfaces (UI's). Perhaps the area that you may have heard about is XAML.

Terminology

Throughout the book, XAML code is referred to as *markup*. This is common with most XML-derived languages.

The term *code* is used when referring to .NET developer language code like C# or VB.NET.

Expression Blend is itself written with XAML and WPF application development technologies such as Cider extensions for Visual Studio. Although there are marked improvements on the performance of the Blend application compared to the very first preview versions, the overall concept is starting to be accepted by the wider development/designer communities as a whole. They understand that performance will improve as the technology matures, which has been proven now with the release of Visual Studio 2008 and the .NET Framework 3.5, both improving the ability to develop for the platform.

You will find out pretty soon, if you like working with XAML. The Blend design interface may take a little longer to get into, simply because it is a different tool with a similar ideology as other timeframe-based design environments (or so it may appear).

When I began using the Expression Interactive Designer Community Technology Previews, I was excited at first. This was finally justifying everything I had been saying to development managers and others who couldn't grasp the importance of a great user interface—and more importantly, a great user experience—for years.

When I had been using it for a few days, I got angry with certain elements of it. It just wasn't working like Adobe Flash does. I couldn't understand why designers would *want* to use it, considering the complexities it would bring to their lives and how their confusion with .NET code and the application features would eventually turn into hostility toward the lowly developer who just wanted to provide the data for a list box and not have to worry about how it looked. It all came together for me a few days later (well at about three in the morning actually) during one of those rare moments of absolute clarity.

The combination of (Designer, Blend, and Visual Studio) is not about how a control looks, it's about how collections of visual and logical assets look, function, and ultimately perform as a singular unit to provide the overall user experience. Blend is not just for designers (or dare I say developers), but it is the head of a toolset aimed at bringing both parties together with a technology that can facilitate stunning designs with awesome .NET application performance.

I can't reiterate enough how important the glue—XAML—is! Graphic designers indirectly use it to make visual assets; interactive designers mold it with Blend; and together they create UIElements that integrate and use specific functionality that adds to an overall visual perception. Developers can use it to implement the designed functionality with additional feature sets.

An example of how XAML allows applications to be created better is when a designer produces a storyboard that shows a gorgeous list box that has glass highlights and rounded borders. The next storyboard shows that when a user clicks on an item in the list box, the selected item flies across the screen. There are very few instances where a conventional WINForms developer would take the time to try and produce this result as per the storyboards in a WINForms environment. But, by working together as a team, the XA makes this entirely possible with Blend and Visual Studio, using the XAML visual assets created by graphic designers. It is also extremely quick to build applications compared to the development time on other existing platforms. The following equation says it all:

(developer + designer) / time = speed = pub^2

The remainder of this chapter is an overview of XAML, complete with explanations of some primary levels of the XAML structures, how they relate to each other, and how it relates to code classes produced in .NET. In Chapter 4, "XAML for Beginners," you will study XAML in a far greater depth.

Before you look at some XAML, there is another issue that struck me when using Blend in the early stages. There is no Source Safe integration. Enterprise developers instantly shy away from products that don't have such integration. But then again, it could make you think a little more about the architecture. In a strict sense, your development environment should always contain some sort of source protection system. That is when the use of multiple client-side application layers came to me. By using this architecture, you can still implement a source protection system that will not affect the UI development.

Layered Understanding

From now on, when you think about the front end of your application, you need to think about layered development, similar to multi-tier or N-Tier development in which you have specific application layers that perform specific tasks for the entire application to function and perform in the manner in which it was designed. Think about your front end having, at the very least, two layers which I describe in detail for clarity.

The Graphical UI Layer

As the name suggests, this layer is where your user interface objects function within their own scope. You, as the XA, now take XAML markup provided by a designer or the design team and make it into a button or other required object in Blend. Previously, if you wanted this button control to have a rollover effect when the mouse moved over it, you would need to provide all the programming that not only created the button in the first place, but also the code to animate it or make it interact with other objects.

The designer creates the visual asset in a design tool like Expression Design that exports XAML. You, as the XA, put all the pieces together; in fact, as you will see, XAML is merely a representation of a .NET code class. As such, it has the power to set properties and apply resources and data binding to-from objects, UIElements, plus much, much more. Putting the design XAML together with the functional XAML is simple.

Let's say, for example, you have made a simple calculator application. When the user clicks the = button on your form, the buttons click handler method goes about taking the inputted values from other controls and or members in the form and then displays the output directly on a textbox control. There is nothing wrong with this; but to get my point across, now we are not going to provide this method of application logic to your simple calculator application.

In your new Blend applications, you or your developer team would go off and write a logic layer that receives inputs and provides outputs back to your graphical layer (see Figure 1.5).

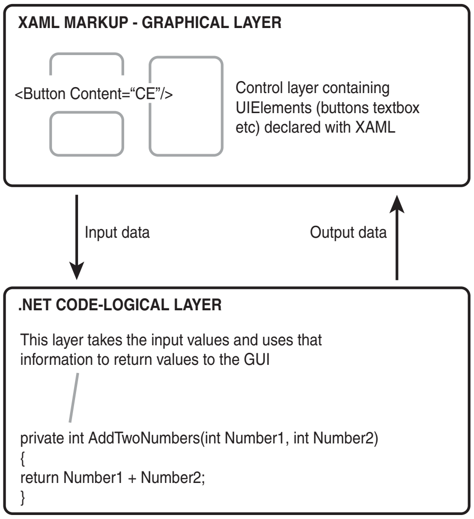

XAML MARKUP - GRAPHICAL LAYER
<Button Content="CE"/>
Control layer containing UIElements (buttons textbox etc) declared with XAML
Input data
Output data
.NET CODE-LOGICAL LAYER
This layer takes the input values and uses that information to return values to the GUI
private int AddTwoNumbers(int Number1, int Number2)
{
return Number1 + Number2;
}

FIGURE 1.5 Data flow between the layers.

The Logic Layer (Class Library)

This layer is where developers author the code to interact and sometimes control the UIElements that will appear in the scenes, by way of events and property change notification. So, to continue with the previous example of a simple calculator, the logic layer (I prefer engine or wrapper) provides public methods with which each click of a button simply adds numbers and symbols for the underlying engine to deal with. The UI layer is free to carry on with any animations or UI-derived actions without the sequential hassles of also trying to deal with the application logic.

Now when the user clicks the = button, the engine is called by the UI graphics layer and goes to work with all the various inputs it has received. It then fires a result out through an event or property change notification where the graphics layer (which has subscribed to this event and or notification) then does what it needs to do in order to display the result.

You may choose to have the graphical assets flash red or go semi-transparent, who knows? The point is that the two layers remain completely separated, which allows developers to easily add unit testing with NUint or a similar tool, as well as gives you a complete set of functionality that will be the same regardless of the front end UI or OS on which you are running the code. The biggest advantage to this method, it has to be said, is from the developer's point of view. When they need to fix bugs or make maintenance changes, they only need to make them to a single code library before testing and redistributing the application. The advantage from the designer's point of view is that they may change the XAML style template applied to the buttons to give them a different look, but the user experience is maintained by you, the XA, controlling the collaboration between the developer and designer in Blend.

XAML Representations

So what exactly is XAML? People may be confused to learn that XAML does not contain objects, shapes, UIElements, animations, or transforms for that matter. XAML is simply an instruction set, and definitely not a programming language.

What Is the Difference Between a Scripting and Programming Language?

It's hard to answer this question without the thought of being skewered by certain people who definitely have very strong views on the subject. My defining characteristic is that a script provides a means to drive an application, while a programming language provides for a script to drive it.

I searched high and low, and still could not find a definitive answer. I did find out that the debate is very much alive on compiler newsgroups!—Freaks.

When these instructions are parsed to the WPF presentation engine, they are then converted to an object tree in memory. So you can think of XAML as being a type of serialization format for WPF, taking all the settings that you specify and then producing your application as the result.

Assuming that you understand the concept of XML formatting and you also grasp the concept that XAML is an XML representation of .NET code objects, take a minute to let the following settle into your mind.

.NET classes are represented in XAML as tags. For example:

```
<MyClass></MyClass>
```

Object children and or complex properties are represented by nested elements. For example:

```
<MyClass>
       <MyClass.Child> Additional Children </MyClass.Child>
</MyClass>
```

Attributes are properties or event handlers, for example:

```
<MyClass Property="100" Click="Click_Event">
       <Child> Value </Child>
</MyClass>
```

To put this into better perspective, let us look at a simple application example.

XAML/CLR Example

We are going to examine the same application (shown in Figure 1.6) twice, once in C# and then in XAML.

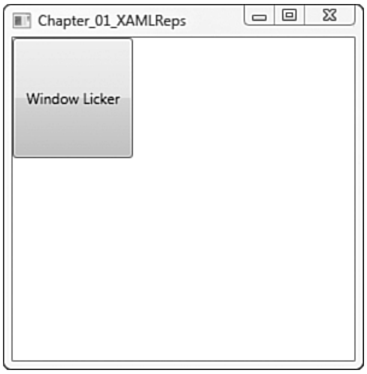


FIGURE 1.6 The simple button application.

As you can see, we have a simple window with a single button shown. The button contains text, but you should bear in mind that we could have used an image or another control (`UIElement`) as the content of the button if a designer required it. Listing 1.1 provides the C# code for this application.

LISTING 1.1 C# Example Application

```
public partial class Window1 : Window
{
    public Window1()
    {
        InitializeComponent();
        //button1 declared within the partial class initializer
        button1.Width = 100;
        button1.Height = 100;
        button1.Content = "Window Licker";
        button1.Click += new RoutedEventHandler(button1_Click);
    }
    void button1_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Simple as that");
    }
}
```

In the code example shown in Listing 1.1, we are setting the properties of the button shown in Figure 1.5, just as a developer would do in a WINForms application. The text is applied by setting the Content property of the button.

Don't worry too much about following the specifics of the code if you are not familiar with C#. Just compare it with the XAML code shown in Listing 1.2, particularly with reference to property names like Width and Height that XAML sets at design time.

LISTING 1.2 XAML Example Application

```
<Window x:Class="Chapter_01_XAMLReps.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Chapter_01_XAMLReps" Height="300" Width="300"
    >

    <Button
       Name="button1"
       Height="100"
       Width="100"
       HorizontalAlignment="Left"
       VerticalAlignment="Top"
       Content="Window Licker"
       Click="button1_Click"/>
</Window>
```

You will note the property names are exactly the same. You may also note there are some layout directives in the form of `VerticalAlignment` and `HorizontalAlignment` that we will look at in future chapters.

This is a very simple example, but one I hope gives you a little insight into the core usage of XAML. You can certainly do a lot more than just declare and set up a button with XAML. As we go through the examples in the book, you will also be looking at both C# and XAML markup to cement your understanding of the XAML/C# relationship.

Code/Markup Integration

When we talk about integration, we are referring to how objects and resources are controlled within an application using either C# or XAML—or both. The key here is to remember that XAML is declarative and can only be applied to those objects and UIElements that are created in the design-time environment.

Although you can apply property setters to objects as well as data binding and dynamic layout instructions, you have to remember that once the application enters runtime mode, you need to apply any new dynamic integration through code alone. This is not to say that you can't add dynamic instructions in XAML; after all, the majority of all animations, trigger events like IsMouseOver for buttons, or even the "Click" event that we used in the Listing 1.2 example, will be declared and set in XAML.

A simple but powerful benefit of the XA role is that because the XA puts all the pieces together, designers and developers are not constrained by timeframes to deliver assets to each other. The XA can add buttons to the scene and apply the style template when the designer is finished. The developers can just get on with writing functionality based on inputs and outputs. Overall the deliverable time of an application is dramatically shortened.

I would like to think that after seeing the brief examples provided in this chapter you are now starting to recognize the purpose of XAML as a declarative script, how simple it will be to use, and how easy it will be for designer/developer collaboration under this workflow. Although some XAML scripts do get quite complicated to follow, once you have a grasp of the core concepts you will be able to follow any of them.

Another point of interest is that Microsoft has tried very hard to ensure that the XAML produced by the Blend designer does not become a mash of utter garbage that you need a deciphering wheel to understand it. You should be able to add a UIElement to your application, change some properties in the property palette within the Blend designer environment, switch to XAML view, and still understand exactly what you have just done. Of course, you can test this by running in Split View mode and seeing the XAML being added as you “draw” controls onto the design surface and vice versa.

The Benefits of Expression Blend

Besides working with a really cool product, you will probably inadvertently start using more and more cutting edge applications to keep in line with the requirements of not only Visual Studio, but also the changing landscape of the .NET Framework and the various operating systems.

You will also pick up new, fresh ideas for designs and methods of working because you will see how working with Blend and WPF has inspired others to reach for new ground in ease-of-use control designs. Expect to see user experience discussions increasing. Those discussions should also generate greater understanding of what *is* important, as opposed to what was never perceived to be so.

Blend will have different levels of engagement for the different types of roles played by those who use it. Interactive designers will use it for one purpose, while a developer will use it in an entirely different manner. In the following sections, try to see how Blend may or may not work for you in some areas and how it might be well-suited to helping in others.

The XAML Architect’s View

The XA will get most of the praise from the commissioners of a project. After all, they (the commissioners) don’t care if an application can split a call into 16 threads to speed up a data return by 0.0005 milliseconds. Most of the time, the big boys and girls (e.g., managers and directors) have no fundamental understanding of what it is you are doing or how it will help with their business. The most important issues to them are higher productivity within their workforce, a strong presence of professionalism, and brand projection. Higher productivity brings enormous cost benefits, which can, by themselves, justify the cost of development.

Perhaps the biggest benefit to a XAML architect is greater responsibility in projects which usually (not always) end with the XA earning more money then a standard developer or designer. XAs need to be able to speak different languages (designer and developer) effectively and act as a mediator between the designer and developer camps. Translating requirements and understanding the designers' vision will enable an XA to make sound architectural judgments about the requirements of the user interface layer.

In any environment in which the power base changes from designer to developer and back, the XA will have the benefit of being in both camps. Even if the XA can't perform as a developer, just understanding their language and concerns is enough. As long as the XA can appreciate and ultimately implement the visual goals of an interactive designer, the XA will see his or her Christmas card list expand!

Blend, along with Visual Studio, will absolutely be the tool of choice for the XA because the workflow between the two products will enable him or her to make quick decisions, test updates, and merge new resources into a solution. Blend becomes more of a XAML management tool for the XA than a visual design interface.

An Example of Power

To test how much of a benefit it is to have a producer/director-like figure (the XAML architect) calling the shots on a project, crack open a game of *Call of Duty 4* and take careful note of the way in which the application has been produced. The game illustrates an immersive, state of the art design at its best. You have no confusion as to what you are—or are about to be—doing.

These games are developed by teams of designers (interactive and graphic) and developers, but are ultimately produced by one or two people that act in the same capacity that an XA would for a WPF solution. Immersive, emotionally driven applications should be the end result that we strive for in standard application development.

The Interactive Designer's View

Some day designers and developers will recognize the fact that unless the user sees it happen on the screen, they will never be 100% sure it actually has happened. What I am talking about is simple user psychology. As an example, I ask you: How many times have you clicked on the Add to Basket button on a website and then had to go check the basket because you were not entirely sure your goodies went in there? With Blend, you can show the user, the item actually flying across the screen and into the basket to dispel any concerns, negative thoughts, and bad experiences that the user may ordinarily have.

I know that is a very simple example—and one that could be overcome using some special DHTML or something else wonderful. But this problem has been part of the user nightmare that interactive designers have been trying to stem, with only moderate success, for years, mainly due to project budgets (time/cost ratio).

An interactive designer might want to get down and dirty with some coding, not so much that the vein in her head will pop, but just enough to make sure the control(s) and/or UIElements perform as they are designed to.

That same designer might want to show some empathy and allow users to apply different control schemes and colors to allow them to *feel* where they are and how they are doing with the application. With Expression Blend, the interactive designer has the freedom to design a control that can show, for example, an ordered list of movies in thumbnails, ensuring that the listing is displayed in the application so that it fits with the overall continuity and visual perception of the project and the vision for the application.

The interactive designer also wants to be able to use his or her favorite graphic design package to create masterpieces of style and then export them into Blend. In this way he or she can ensure that design elements actually look like they are supposed to instead of having to ask a developer to try to make the elements look like the image the interactive designer just printed out.

Designers want to be involved much more in the lifecycle than they are at present. With the power to create fully functioning prototypes with Blend, which therefore expedites production lifecycles, the importance of the interactive designer's role is moved much higher up the food chain.

Going back to the games industry, interactive designers and graphic artists are the super stars, not so much the coders. Interactive designers take a leading role to ensure that the entire application is smooth in its appearance and that every animation, the music, and the fonts in the menus all come together as a perfect match of design integrity.

The Coder's View

I am sure that some of you reading this book as developers have, at one time or another, been told by a project manager or IT director that the application you have slaved over just does not look like the application they envisioned or were sold on. Another common concern is that test users complain about the time it takes to perform a certain task after taking numerous steps in order to navigate a complex screen. It's possible that your application has used controls from various third parties and that they just don't fit the overall visual perception because you can't modify the appearance or style of those controls. Even Microsoft still distributes controls like the CLR TreeView control that doesn't support transparent background colors. These are all very common issues relating to large application development solutions where designs are always changing to address functional, presentational, or usability issues.

Such issues most often occur in Agile environments, because development cycles are short, fluid, and incremental. Developers are often hamstrung by strict adherence to timelines in which they must provide functionality and maintainable levels of bug fixing. As a result, the user interface often gets the least attention, which leads to the poor user experience I keep talking about.

Blend means that developers can finally just focus on making sure that code functions as it is required to in a specification. Their objects get a value in, and then they make sure the correct value goes out. It's as simple as that. The developers can even test the object code to make sure someone else doesn't change the designed functionality. You can be guaranteed a format or a value type every time. The interface can either be designed around that format, or, preferably, the coder's objects can be designed by what they are required to input/output.

A lot of middle tier and backend developers will be extremely pleased that they no longer have to think about the user interface—nothing scares them more. They can just get on with coding and even help with the logical client layer, all without ever seeing the UI that they are developing for. They also have the added bonus of being able to apply unit testing to the logical layer, which fits in well with the psyche of the nonUI developer.

The Workflow

Traditional workflow in the development lifecycle has always depended on the development company's work practices and is based on whether they use standards to ensure quality products. Not all businesses do, and some even think they are not big enough to go to the trouble. As a single developer, I always use standards in my everyday work to ensure that I can provide not only quality results, but also auditable paths of my work to my clients.

You may, at present, be involved with designer and developer teams. Perhaps a move to a new job at a new company in the future will put you in this position. Either way, it is important to understand how this designer/developer workflow is supposed to work.

I have previously stated how Blend will significantly reduce the production time on solutions; but until you understand the time periods that will be removed, it is hard to see that becoming reality. Many interactive designers have their own methods of producing mock-up applications or prototype designs. Some choose to use animated environments such as Flash, some use PowerPoint, some use Visio, and others like to create traditional storyboards to indicate how they see the interface working. End users are then brought in to give initial feedback. Refinements are then made based on this research.

This design process allows the interactive designer to spot user workflow issues and other process improvement opportunities that are vital to an application being accepted by the end user. The process, called the blueprint, may also require a variety of diagrams: a structural diagram (or application map), a process diagram to provide architects an essential view, and sometimes a wireframe diagram to illustrate how different screens will appear.

The only problem with the blueprint now is that almost all the work done during its creation process is wasted when it comes time to actually develop the solution. Developers may (and it's not always the case) be able to use some of the visual assets created, the static images and the color palettes; but they certainly won't be able to use any animation sequences or event-driven reactions that are so crucial to informing the user of what has occurred. Remember the Add to Basket scenario?

Applications like Flash, Visio, and PowerPoint, when used in the interactive design context, are purely conceptual, whereas Expression Design, Expression Blend, and Visual Studio are production tools. Using the production tools, you can reach the end goal a lot quicker and the integrity of the design can be controlled and maintained by the designers instead of the developers.

Using Blend to deliver the blueprint now means that the interactive designer is creating the actual user interface that will be present in the end product. Blend provides all the tools necessary to deliver the animation of objects, providing the rich user experience end users crave; and perhaps more importantly, Blend provides a separate layer of development that the code developers never really need to modify or be involved with.

The workflow between Blend and Visual Studio allows the interactive designer to quickly test object implementations in the data binding scenarios and customized controls that are present in many applications.

The interactive designer along with the XAML architect, then, have the ability to accurately describe the required object model to the development team, which will, in most cases, provide a base object model with testable methods. That allows the interface to provide data, as well as test data being returned, all before the majority of the backend code is written.

The Importance of an Accurate Test Object

We will look at creating CLR data bound objects in Chapter 13, "Data Binding," where we will look at a method for creating test data. It is important to do this so the templates we design in Blend show real-world data and so we can make adjustments to either the data object—or the presentation of that data—without having to involve a developer, a data source such as a database, or a valid connection to a data source to do it. Methods such as these also speed up the development process by removing the reliance of one team on the other.

The End User or Client

End users always want simplicity. The Chinese discovered thousands of years ago that it was simpler to remember topics or points of interest when those topics were displayed as a picture or icon. The old adage, "A picture is worth a thousand words," rings true with today's marketing gurus who believe they can sell you anything if they can show it to you. People need to see vivid imagery to give themselves perspective.

According to George A. Miller (founder of the Center for Cognitive Studies at Harvard), we mere mortals tend to have a memory capacity of 7 + –2 items (Nelson Cowan revised this to 4 + –1 in 2000); yet, we can recall large collections of images, especially those that form part of our habits, which our brains are particularly fond of.

Why would you give the end user a static image button when you could show them an animated response to their choice? Why would the end client be happy having to rely on old data when they could be viewing the information live and in a graphical representation? The quick answer is they won't be happy. If your company's competition provides a better user experience in their products, you shouldn't expect to keep your job for much longer.

When designed correctly and composed with perfect harmony, a WPF application will give users an extremely positive experience. In some cases, people will forget they are using a computer and feel at one with the application. I have witnessed people who are single finger typists who suddenly discover extra fingers. The positive experience has added to their productivity.

You would have at one time or another experienced these really high levels of positive flow and productivity because the most common side effect is losing track of time. Time flies when you are having fun and enjoying yourself.

Again, when applications are built with the cooperation of the designer and the application developer, the end user will have no doubts as to what they are doing, what has just occurred, and how they should proceed next. With WPF there is simply no excuse for not delivering fantastic user experience. If you become a XAML architect, you could ensure the application delivers the appropriate user experience.

Having a well-designed product in terms of usability and visual appeal will ensure that your company earns development costs back in the long term because of the loyalty that a professional, easy to use, and fun application brings. Positive feelings are also transferred to a brand so, in the case of a publicly released piece of software, happy users are much more inclined to use new, different, and simple applications. Most importantly, they are likely to tell their friends and colleagues about it.

The end user demands perfection. We (the collective industry) can't expect them to keep shelling out for new hardware every 18–24 months if we are not going to make them feel good about their purchases by giving them a positive experience.

Would you buy a sports car if the seats where uncomfortable, the suspension was rubbish, and the driving experience was not up to the level that you expected—even if the motor was an absolute beast? Maybe...maybe not.

Maybe next time, instead of buying a PC, the user will go with a *fruitier* choice of computer, to see what it has to offer. If this happens too often, you and or I may lose our jobs.

Summary

Throughout the Interactive Designer CTP's and subsequent Blend 1 and Blend 2 preview releases, people from differing sides of the designer/developer roles have argued about what Blend is and who it is actually for. Maybe they only looked at it from their angle, without taking or trying to experience the other side's arguments and requirements. Either way, it is difficult to create a truly collaborative environment and, perhaps even more difficult, to implement the tools used to facilitate it. Expression Design, Expression Blend, and Visual Studio represent a real step ahead in the right direction to achieving this by speeding up the design/development process and allowing the original designers of a solution to maintain ownership of it.

Blend, used by a XAML architect, allows the collaboration to work in a real-world scenario, be it a small or large solution team. Only experience in working on WPF solutions will bring this requirement forward into discussions in future.

Blend offers you, the graphic or interactive designer, a way to decrease your delivery times while maintaining all your hard work throughout the solution development lifecycle and on to the end user. In other words, what you originally design is what you can now deliver instead of letting developers interpret and implement your visual requirements.

It doesn't mean you will never again need to use any of the tools you have used previously; it just means you will have the option to create, control, maintain, modify, and own your designs—instead of just hoping that someone else will.

Index

Symbols

{ } (curly braces), XAML markup extensions, 98

[ ] (square brackets), collections, 313

_ (underscores), resource names, 202

3D modeling tool, 169

3D objects

- camera properties, 518
- creating, 226-227
- importing, 515-516
- light properties, 518, 521
- materials properties, 516-517
- Viewport3D element, 511
    - 3D mesh, creating, 511-513
    - axis transformation, 513
    - Camera Orbit tool, 514
    - Selection tool, 513

A

actions, triggers, 68

activating elements, 49

active triggers, 67

Add( ) method, 113, 332

Add New Item dialog box, 201

AddChild( ) method, 95

AddHandler( ) method, 308-309

adding

- Button controls, 123
- Child controls
    - comboboxes, 221
    - embedded UIElements, 113-115

Click Event handlers
Button control, 123
ProgressBar control, 124
columns, 135-136
images, 226
items/actions, triggers, 68
KeyFrames, 70
Label controls, 122
NET code to XAML, 43-46
points, gel button, 179
ProgressBar controls, 122
RichTextBox controls, 242
rows, 135-136

adorners, 50

Alignment property (Grid), 112, 134

AllowsTransparency property (windows), 476

ambient light properties (3D objects), 521

animations
elements, 190-191
gel button glow, 191
animation requirements, 194
GlowBase element animation, 196-198
mouse over trigger, 194-196
timeline, adding, 194
handoff animation, 432
KeyFrames
adding, 418-419
erasing, 420-425
repeat animation, 420
overview, 415
production, 6
timelines
controlling in code, 434-436
duplicating, 433
overview, 415-418
reversing, 433
states, 428
triggers
definition, 425
event triggers, 425-428
property triggers, 428-432
recording, 429
timeline states, 428

appearance category (Properties panel), 79-80

applications
building, 41
data binding sample application, 391-396
graphics sample application, 323-325
Hello World, 31-36
align property search, 35
C# code in XAML, 44-46
name/location, setting, 32
properties, 34
TextBox, 33
XAML label control, 38
XAML/Visual Studio interactivity, 39-42
saving in Visual Studio, 41-42
XAMLNotepad
creating, 443-445
menu items, 446-452
overview, 442
XAML packages, 452-457

applying
control templates, 364-365
styles across projects, 374-376

architectural benefits of Blend, 24-25

AreaToApplyEffect property, 348-349

AreaToApplyEffectUnits property, 348-350

arrays, 310

artboard live view, 33

Asset Library, 88-89
overview, 210
tabs
Controls, 210-211
Custom Controls, 211
Local Styles, 211
Media, 211
Recent, 212

axis transformation (3D objects), 513

B

backing field
- CLR properties, 278
- dependency properties, 279

Balance property (MediaElement control), 526

Barker, Stephen, 350

Begin( ) method, 434

BeginTime property (animations), 417

benefits of Expression Blend, 24
- architects, 24-25
- coders, 26-27
- designers, 25-26
- end users, 28-29
- workflow, 27-28

BevelBitmapEffect, 346

binding, 54, 391

Binding extension, 99-101

BitmapEffectGroup, 346-347

BitmapEffectInput property, 348

BitmapImage object, 230

bitmaps, 225
- effects, 16
  - AreaToApplyEffect property, 348-349
  - AreaToApplyEffectUnits property, 348-350
  - BevelBitmapEffect, 346
  - BitmapEffectGroup, 346-347
  - BitmapEffectInput property, 348
  - BlurBitmapEffect, 346
  - DropShadowBitmapEffect, 346
  - EmbossBitmapEffect, 346
  - hidden bitmap effects, 345
  - Input property, 350
  - OuterGlowBitmapEffect, 346
  - overview, 345
- vector graphics, compared, 12

BlurBitmapEffect, 346

bool data type, 271

Border element, 141-142

Bread Crumb controls (Vista), 10

Brush class, Stroke property, 325

Brush Transform tool, 167-168, 181-182

brushes
- gel button creation
  - Fill Property brush/Linear Gradient brush, 177
  - Fill Property transform, 184
  - suggestion table sample one, 176
  - suggestion table sample two, 178
  - suggestion table sample three, 180
  - suggestion table sample four, 181
  - suggestion table sample five, 182
  - suggestion table sample six, 182
  - suggestion table sample seven, 184
  - suggestion table sample eight, 184
  - suggestion table sample nine, 193
  - transforms, 183
- gradient stops, 178
- RadialGradientBrush, 330-334
- SolidColorBrush, 178, 328-330
- strokes, 178, 325-328

brushes category (Properties panel), 77-79

building
- applications, 41
- menu items, 446-452

business mindset of users, 8-10

Button control, 213-215
- adding to TreeView control, 123
- application example, 22
  - C#, 22
  - XAML code, 23
- Click Event handler, 123
- Gel button listing, 214-215
- inheritance hierarchy, 490
- names, 121

C

C#
- adding to XAML example, 44-46
- button application example, 22
- classes, 275
  - creating, 275
  - namespaces, 280
  - scope, 276-277
- collections, 310-311
  - [ ] (square brackets), 313
  - boxing/un-boxing, 311
  - car stock list example, 316-318
  - dictionary, 313-315
  - foreach statements, 315-318
  - generic, 311
  - int values, adding, 312-313
  - strongly typed generic, 311
  - zero-based, 310
- conditional coding, 281-284
- data types, 270-273
- enumerations, 284
  - creating, 285
  - declaration example, 284
  - dependency property, 285
  - switch statement, 286-289
- events, 298
  - child controls, 309-310
  - CLR, 298-303
  - overview, 298
  - raising, 300
  - RoutedEvents, 303, 306-309
- methods, 289
  - amount of, 293
  - calling, 290-293
  - comments, 293
  - mathematical symbols, 298
  - overloading, 290
  - refactoring, 290
  - Show( ), 289
  - try-catch statement, 295-297
  - two blocks of code with same functionality example, 289
  - user input examples, 294-297
  - Width property value, 297
- properties, 277
  - CLR, 278
  - dependency, 279-281
- Tree control example listing, 115-119
- variables, declaring, 270

calling methods, 290-293
- example listing, 292-293
- names, 291
- public, 291
- this keyword, 291

Camera Orbit tool, 168-173
- listing, 169-171
- objects, creating, 171-172
- Viewport3D element, 514

camera properties (3D objects), 518

Camera Type property, 518

CanExecute( ) method, 438

CanExecuteChanged( ) method, 438

Canvas element, 137-138
- positioning properties, 137
- switching between Canvas and Grid modes, 134

Category Collapse option (Interaction Panel), 65

center points, 337-338

char data type, 271

CheckBox control, 216-220
- events, 220
- menu created from checkboxes example listing, 216-219

child collections, 94

child controls
- adding
  - comboboxes, 221
  - embedded UIElements, 113-115
- events, 309-310
- nesting in comboboxes, 221
- Stack panel alignment example, 139-140
- templates, 250
- XAML, 94-96
  - embedded listbox example, 95-96
  - interfaces, 95

child interfaces, 95

Children property (Add( ) method), 113

classes, 275
- Brush
  - RadialGradientBrush, 330-334
  - SolidColorBrush, 328-330
  - Stroke property, 325-328
  - StrokeDashArray, 326
  - StrokeDashCap property, 327
  - StrokeDashOffset property, 327
  - StrokeThickness property, 327
- creating, 275
- MediaPlayerController, 538-540
- namespaces, 280
- .NET, XAML representations, 21
- Object, 490
- OpenFileDialog, 532
- Panel, 132
- Patient, 408-409
- Patient_BINDER, 409-412
- scope, 276-277
- ValidationRule, 400
- Window, 37

Click Event handlers
- Button control, 123
- ProgressBar control, 124

ClipToBounds property, 137

Clock property (MediaElement control), 526

closed event, 483

closing event, 482

CLR (Common Language Runtime), 278
- data binding, 406-412
- events, 298-303
  - constructors, 301-303
  - DoorOpen implementation, 300
  - flow, 299
  - RoutedEvents, compared, 303
- properties, 278

code, 18

coding benefits of Blend, 26-27

collections, 310-311
- [ ] (square brackets), 313
- boxing/un-boxing, 311
- car stock list example, 316-318
- child, 94
- CommandBindingCollection, 438
- DashCollection, 330
- dictionary, 313-315
- foreach statement, 315-318
- generic, 311
- List, 312-313
- MergedDictionaries
  - elements, customizing, 465
  - entries, deleting, 466
  - skins, 462-465
- strongly typed generic, 311
- zero-based, 310

color property (3D objects lighting), 519-520

colors, validating, 398-399

ColumnDefinitionCollection Editor, 122

columns (Grid), 135-136

ComboBox control, 221-222
- adding child elements, 221
- nested child elements, 221
- properties, 222
- unnamed items, creating, 223-225

CommandBindingCollection, 438

commands
- CommandBindingCollection, 438
- defining, 440-442
- ICommand interface, 438
- invoking, 442
- overview, 437-438
- relative, 154
- WPF (Windows Presentation Foundation)
  - document, 439-440
  - input, 438-439

comments, 293

common controls UIElements tools, 150

Common Language Runtime. ***See*** **CLR**

common properties category (Properties panel), 81

CompositionTarget, 541
conditional coding, 281-284
configuring window transparency, 477-480
ConstantAttenuation property (3D objects lighting), 519-520
constructors
  CLR events, 301-303
  default, 301
content controls, 129
ContentControl control, 246-248
  ContentPresenter element, 249-250, 254
    ContentControl child element, 250
    ListBox, adding, 253
    styled rectangle element, 250
    styled templates, applying, 252-253
  ListBox styling, 250
ContentPresenter control, 249-250, 254
  ContentControl child element, 250
  ListBox
    adding, 253
    styling, 250
  styled rectangle element, 250
  styled templates, applying, 252-253
Control level (custom controls), 491
control templates
  applying, 364-365
  creating, 201
  definition, 360
  editing, 361-363
  gel button, creating, 201-202
  overview, 360
  testing, 204-206
controls, 241
  Border, 141-142
  Button, 213-215
    adding to TreeView control, 123
    application example, 22-23
    Click Event handler, 123
    Gel button listing, 214-215
    inheritance hierarchy, 490
    names, 121
  Canvas, 137-138
    positioning properties, 137
    switching between Canvas and Grid modes, 134
  CheckBox, 216-220
    events, 220
    menu created from checkboxes example listing, 216-219
  Child
    adding to comboboxes, 221
    adding to embedded UIElements, 113-115
    events, 309-310
    nesting in comboboxes, 221
    Stack panel alignment example, 139-140
    templates, 250
    XAML, 94-96
  ComboBox, 221-222
    adding child elements, 221
    nested child elements, 221
    properties, 222
    unnamed items, creating, 223-225
  ContentPresenter, 249-250, 254
    ContentControl child element, 250
    ListBox, 250, 253
    styled rectangle element, 250
    styled templates, applying, 252-253
  control templates
    applying, 364-365
    creating, 360-361
    definition, 360
    editing, 361-363
    overview, 360
  custom text box, 492
    complete control listing, 504-508
    creating, 492
    dictionary functionality, 495
    dynamic history dictionary, 498
    filtering the dictionary, 499-500
    ListBox control, 496-497

new class, creating, 494
OnKeyUp event handler, 500-501
Popup control, 497-498
testing, 501-503
TextBox class inheritance, 494
customizing, 489
Control level, 491
FrameworkElement level, 491
inheritance hierarchy for button controls, 490
Inherited Control level, 491
UserControl level, 491
data binding
data validation, 397-400
sample application, 391-396
two-way binding, 396-397
Dock panel, 140-141
FlowDocument, 243-245
GlowBase, 196-198
Grid, 132-137
Alignment property, 134
creating, 133
layout constraints, 135
Margin property, 134
rows/columns, adding, 135-136
switching between Canvas and Grid modes, 134
Width and Height properties, 134
ItemsCollection, 222-225
Label, 231
adding to TreeView control, 122
mnemonics, 231
names, 121
limitations, 129
Listbox, 232-233
custom text box control, 496-497
embedded, creating, 95-96
ListBoxItems, 232-233
ListBoxItem, 232-233
ListView, 233-238
data templates, 237-238
four column example, 236
hidden default view, 235
ListViewItems, 234-238
view columns, 235
view example, 234
ListViewItems, 234-238
data templates, 237-238
four column example, 236
hidden default view, 235
view columns, 235
view example, 234
media player, 528-532
completed design, 531
menu item, creating, 532
names, 529
required, 530
row/column definitions, 529
viewing, 531
MediaElement control
event handlers, 533-535
properties, 525-527
OpenFileDialog with user input, 532-533
PasswordBox, 238-239
Popup, 264-266
custom text box control, 497-498
defining, 264
force closing, 265
properties, 266
ProgressBar, 240
adding to TreeView control, 122
Click Event handler for, 124
prefix naming convention, 121
RadioButton, 241
RichTextBox, 241-245
adding, 242
alignment, 242
element types, 243
FlowDocument element, 243-245
ScrollViewer, 141, 266-268
creating, 267
properties, 268
Slider, 245-246
properties, 246
values, customizing, 245

Stack panel, 138-140
  child element alignment example, 139-140
  Orientation property, 138
styles, 461
TabControl
  HeaderedContentControl control, 254-256
  properties, 255
  TabItems, adding, 255
TabIndex property, 232
TabItems, 255
TreeView, 115, 257
  Button control, adding, 123
  C#, XAML listing, 115-119
  child object management, 126-128
  Click Event handler for Button control, 123
  Click Event handler for ProgressBar control, 124
  code behind file, 119
  ColumnDefinitionCollection Editor, 122
  Label Content property, 125
  Label control, 120-122
  ProgressBar control, adding, 122
  TreeViewItem Items content value, setting, 121-122
  XAML for entire control listing, 125
  XAML representation listing, 119
TreeViewItem, 258-259
  creation listing, 258
  properties, 259
Uniform Grid, 142
ViewBox, 143-144
Viewport3D, 511
  3D mesh, creating, 511-513
  axis transformation, 513
  Camera Orbit tool, 514
  Selection tool, 513
WPF
  ContentControl, 246-250, 254
  Expander, 259-261
  GridSplitter, 262-263
  HeaderedContentControl, 254-256
  HeaderedItemsControl, 256-259
  Popup, 264-266
  ScrollViewer, 266-268
Wrap panel, 140
XAML, 92-93

Controls tab (Asset Library), 210-211

Create Data Binding dialog box, 54

Create Data Template dialog box, 55

Create Style Resource dialog box, 201, 428

curly braces { }, XAML markup extensions, 98

Custom Controls tab (Asset Library), 211

customizing
  controls, 489
    Control level, 491
    FrameworkElement level, 491
    inheritance hierarchy for button controls, 490
    Inherited Control level, 491
    UserControl level, 491
  resources, 73
  skins, 460-462
  slider values, 245
  text box control, 492
    complete control listing, 504-508
    creating, 492
    dictionary functionality, 495
    dynamic history dictionary, 498
    filtering the dictionary, 499-500
    ListBox control, 496-497
    new class, creating, 494
    OnKeyUp event handler, 500-501
    Popup control, 497-498
    testing, 501-503
    TextBox class inheritance, 494
  Window properties, 110-113
  Z-order, 132

cycling, workspace, 36

D

DashCollection, 330
data binding
  CLR (Common Language Runtime) data binding, 406-412
  Data Context, 401-404
  data sources, 391
  definition, 54
  elements
    data validation, 397-400
    sample application, 391-396
    two-way binding, 396-397
  overview, 391
Data category (Project panel), 76
Data Context, 401-404
data sources
  creating, 378-388
  overview, 391
  setting, 376-378
data templates
  creating, 378-388
  data sources, 376-378
  overview, 376
data types
  Blend, finding, 274-275
  C#, 270
    int, 270
    primitive, 270-273
data validation, 397-400
default constructors, 301
defining commands, 440-442
delegates, 299
deleting
  gradient stops, 178
  items/actions, triggers, 68
  KeyFrames, 420-425
  MergedDictionary entries, 466
  segment points, 158
  triggers, 66
Dependency Properties, 279
design/split view (panels), 63
designers, 6
designer benefits of Blend, 25-26
dialog boxes
  Add New Item, 201
  Create Data Binding, 54
  Create Data Template, 55
  Create Style Resource, 201, 428
  Edit Repeat, 420
  New Project, 32
  Storyboard Picker, 69
dictionaries, 313-315
diffuse material property, 516
Direct Selection tool, 179, 154-158
Direction (x,y,z) property, 518-520
directional light properties (3D objects), 521
directives, 224
Dock panel, 140-141
docking Interaction Panel, 65
document commands (WPF), 439-440
DoorOpen CLR event implementation, 300
dot notation, 290
double data type, 271
DPs (Dependency Properties), 279-281
  backing field, 279
  enumerations, 285
  public property wrapper, 279
  values, assigning, 281
DragIncrement property (GridSplitter control), 263
DrawingBrush resource, 229
DropShadowBitmapEffect, 346
duplicating timelines, 433

E

Edit Repeat dialog box, 420
editing
  control templates, 361-363
  styles, 367-369

elements. *See also* controls
activating, 49
animating, 190-191
locking/unlocking, 70
names, 121
triggers, selecting, 68
viewing/hiding, 70
windows, 475
multiple, creating, 480
names, 480
properties, 476-477
switching between in code, 480-485
transparency, configuring, 477-480
child, 94-96
nesting, 94
viewing, 36-37
Ellipse tool, 164-166
ellipses, creating, 339
embedding
fonts, 84
listboxes, 95-96
UIElements, 106-107
child controls, adding, 113-115
example listing, 106
positioning, 107-110
TreeView control example, 115
Window properties, customizing, 110-113
EmbossBitmapEffect, 346
emissive material property, 516
end user benefits, 28-29
enumerations, 284
creating, 285
declaration example, 284
dependency property, 285
switch statement, 286-289
events, 298
CheckBox, 220
child controls, 309-310
closed, 483
closing, 482
CLR, 298-303
constructors, 301-303
DoorOpen implementation, 300
flow, 299
RoutedEvents, compared, 303
handlers
Button control, 123
MediaElement, 533-535
MediaPlayerController class, 538-540
OnKeyUp, 500-501
ProgressBar control, 124
RoutedEvent example, 308-309
TextChanged, 397-398
XAML, 21
MouseMove, 206-207
overview, 298
raising, 300
RoutedEvents, 198
CLR events, compared, 303
declaring, 306-307
event handler method, 308-309
gel button creation, 199-201
overview, 303
raising, 307
website, 310
window state, 483-484
timeline, 190-191
triggers, 66, 425-428
tunneling, 203
XAML, 97-98
Execute( ) method, 438
executing commands, 442
ExpandDirection property (Expander control), 261
Expander control, 259-261
controlling, 261
creating, 260
properties, 261
Expression Blend, 6
benefits, 24
architects, 24-25
coders, 26-27
designers, 25-26

end users, 28-29
workflow, 27-28
design environment, 13
Visual Studio, compared, 17

Expression Design, 6
definition, 350
vector objects, building, 350-356

external skin files, loading, 468, 471-473
code listing, 470
ComboBox selection event, 471-473

Eyedropper tool, 166-167

F

Far Clipping Plane property, 518

File Explorer (Vista), 10

files
external skin, loading, 468, 471-473
code listing, 470
ComboBox selection event, 471-473
media formats, 524
MediaPlayerController.xaml.cs, 548-553
referencing, 502
resources, linking, 204
Window1.xaml.cs, 543-548

Files category (Project panel), 74-75

Fill Property brush, 177

filtering resources, 72

finding Blend data types, 274-275

FindResource( ) method, 434

flipping objects, 339

FlowDocument control (RichTextBox controls), 243-245

FocusVisualStyle, 370-374

fonts, embedding, 84

foreach statement (collections), 315-318

FrameworkElement level, 491

Freeze( ) method, 332

FromRgb( ) method, 332

G

gel button, creating, 176-178
adding to Resource Dictionary, 201-202
animated glow, 191
animation requirements, 194
GlowBase element animation, 196-198
mouse over trigger, 194-196
timeline, adding, 194
animation, 190-191
arrow shape, 179
base, 178
base brush suggestions, 180
brush stroke property settings, 181
brush suggestion tables
one, 176
five, 182
six, 182
seven, 184
eight, 184
nine, 193
brush transforms, 183
Button control listing, 214-215
fill color, 178
Fill Property brush/Linear Gradient brush, 177
Fill Property transform, 184
gradient stops, 178
highlights, 181-182
Make Button tool, 188-190
points, adding, 179
RoutedEvents, 199-201
shading, 178
shapes completed, 179
smoothing, 179
stroke thickness, 178
testing, 203
MouseMove event, 206-207
Resource Dictionary/control template, 204-206
XAML code listing, 185-188

generic collections, 311

GlowBase element, 196-198
gradient brushes, creating, 330-334
gradient stops, 178
graphical UI layer, 19-20
graphics, 225
- adding, 226
- animations
  - handoff animation, 432
  - KeyFrames, 418-425
  - overview, 415
  - timelines, 415-418, 433-436
  - triggers, 425-432
- BitmapImage, 230
- bitmaps, 225
  - effects, 345-350
  - vector graphics, compared, 12
- brushes
  - RadialGradientBrush, 330-334
  - SolidColorBrush, 328-330
  - strokes, 325-328
- converting to 3D, 226-227
- creating dynamically, 227-229
- ellipses, creating, 339
- ImageBrush example, 230
- motion paths, 339-344
- overview, 323
- properties, 230
- sample application, 323-325
- transformations
  - center points, 337-338
  - definition, 334
  - flip, 339
  - LayoutTransform property, 334
  - RenderTransform property, 334
  - rotation, 336-337
  - scale, 337
  - skew, 337
  - translation, 335-336
- vector objects
  - bitmaps, compared, 12
  - creating, 350-356
- XAML vector, 225

Grid, 132-137
- Alignment property, 134
- creating, 99, 133
- layout
  - constraints, 135
  - properties, 112-113
- Margin property, 134
- prefix naming conventions, 121
- rows/columns, adding, 135-136
- switching between Canvas and Grid modes, 134
- Width and Height properties, 134

GridSplitter control, 262-263
- creating, 262
- properties, 263

H

handoff animation, 432
hardware, next generation, 4
HeaderedContentControl control, 254
- HeaderedItemsControl control, compared, 256
- TabControl, 254-256

HeaderedItemsControl control, 256-257
- HeaderedContentControls, compared, 256
- TreeView controls, 257
- TreeViewItem controls, 258-259

Height property (Grid), 112, 134
Hello World application, 31-36
- align property search, 35
- C# code in XAML, 44-46
- name/location, setting, 32
- properties, 34
- TextBox, 33
- XAML
  - label control, 38
  - Visual Studio interactivity, 39-42

hidden bitmap effects, 345
HorizontalOffset property (Popup control), 266
HorizontalScrollBarVisibility property (ScrollViewer control), 268

I

IAddChild interface, 95
ICollection interface, 95
ICommand interface, 438
Icon property (windows), 476
if-else condition, 281-284
ImageBrush example, 230
images. *See* graphics
implementing DoorOpen CLR event, 300
importing 3D objects, 515-516
  camera properties, 518
  light properties, 518, 521
  materials, 516-517
inheritance hierarchy for button elements, 490
Inherited Control level (custom controls), 491
InnerConeAngle property (3D objects lighting), 520
input commands (WPF), 438-439
Input property, 350
int data type, 270-271
IntelliSense functionality in Visual Studio, 120
Interaction Panel, 65
interfaces
  child, 95
  IAddChild, 95
  ICollection, 95
  ICommand, 438
interpolation values (KeyFrames), 421
invoking commands, 442
IsDirectionReversed property (Slider control), 246
IsDropDownOpen property (comboboxes), 222
IsEditable property (comboboxes), 222
IsExpanded property
  Expander control, 261
  TreeViewItem control, 259
IsIndeterminate property (ProgressBar control), 240
IsMuted property (MediaElement), 526
IsOpen property (Popup control), 266
IsSelected property (TreeViewItem control), 259
IsSnapToTickEnabled property, 246, 397
Items property (TabControl element), 255
ItemsCollection control, 222-225

J–K

KeyFrames (animations)
  adding, 70, 418-419
  erasing, 420-425
  repeat animation, 420
KeySpline editor, 423-425
keywords
  this, 291
  try, 295

L

Label Content property (TreeView control), 125
Label control, 231
  adding to TreeView control, 122
  mnemonics, 231
  names, 121
languages
  programming versus scripting, 21
  selecting, 13, 32
LargeChange property (Slider control), 246
layers
  gel button creation, 176-178
    arrow shape, 179
    base, 178
    base brush suggestions, 180
    brush stroke property settings, 181
    brush suggestions sample one, 176
    brush suggestions sample five, 182
    brush suggestions sample six, 182
    brush suggestions sample seven, 184
    brush suggestions sample eight, 184

brush transforms, 183
fill color, 178
Fill Property brush/Linear Gradient brush, 177
Fill Property transform, 184
gradient stops, 178
highlights, 181-182
points, adding, 179
shading, 178
shapes completed, 179
smoothing, 179
stroke thickness, 178
graphical UI, 19-20
logic, 20-21
layout category (Properties panel), 80
layout modes (panels), 63
layout properties (Grid), 112-113
LayoutTransform property (transformations), 334
Left property (windows), 476
levels of custom controls, 491
light properties (3D objects), 518, 521
limitations of content controls, 129
Line tool, 152-154
mini-language syntax, 153-154
strokes, 152
Linear Gradient brush, 177
LinearAttenuation property (3D objects lighting), 520
linking file resources, 204
List collections, 312-313
ListBox control, 232-233
custom text box control, 496-497
embedded, creating, 95-96
ListBoxItems, 232-233
listings, 59
adding
brushes to Window's resource collection, 98
int values to List collections, 312-313
Binding extension, 99-101
C#
button application example, 22
code in XAML, 44-45
primitive data type, creating, 272-273
Camera Orbit tool, 169-171
car stock list collection example, 316-318
checkboxes
events, 220
menu created from, 216-219
classes
declaring, 276
instantiating, 277
CL
events, 300-303
properties, 278
comboboxes
adding child elements, 221
nesting child elements, 221
unnamed combobox items, creating, 223-225
control styles, 461
creating new grids during runtime, 99
custom text box control
complete listing, 504-508
dictionary functionality, 495
dynamic history dictionary, 498
filtering the dictionary, 499-500
ListBox control, 496-497
OnKeyUp event handler, 500-501
Popup control, 497-498
testing, 503
TextBox class inheritance, 494
dependency properties
backing field, 279
public wrappers, 279
values, 281
dictionaries, creating, 313-315
Direct Selection tool, 154
embedded listbox, 95-96
enumerations
creating, 285
declaration, 284

dependency property, 285
Expander control, 260-261
external skin files, loading
ComboBox selection event, 471-473
loading resources from files, 470
gel button creation
brush suggestion table sample one, 176
brush suggestion table sample two, 178
brush suggestion table sample three, 181
brush suggestion table sample four, 181
brush suggestion table sample five, 182
brush suggestion table sample six, 182
brush suggestion table sample seven, 184
brush suggestion table sample eight, 184
brush suggestion table sample nine, 193
LayoutRoot element, 203
XAML code, 185-188
Grid rows/columns, adding, 135
GridSplitter control, 262
ImageBrush example, 230
Line tool, 153
ListBoxItems, 232
ListViewItems
data templates, 237-238
four column example, 236
view example, 234
media player project
button state, 535
controls, column/row definitions, 529
event handlers, 533-535
media duration, checking, 541
media failure errors, 540
MediaPlayerController event handlers, 538-540
MediaPlayerController.xaml.cs, 548-553
Open File Dialog, 533
safe media control, 542-543
user playback position changes, 537-538
Window1.xaml.cs, 543-548
MergedDictionaries property from sample App.xaml file, 462
MergedDictionaries, resources, swapping, 463-465
methods
calling example, 292-293
overloading, 290
two blocks of code with same functionality, 289
user input validation example, 294-295
MouseMove event, testing, 206-207
nested elements, 94, 97
.NET equivalent for XAML button declaration, 93
PasswordBox example, 239
Popup control
defining, 264
force closing, 265
ProgressBar values, 240
RadioButtons, 241
RichTextBox control
adding, 242
FlowDocument element, 243-245
RoutedEvents
declaring, 306-307
event handler method, 308-309
raising, 307
window state, 483-484
ScrollViewer control, creating, 267
skin customizations, 467
slider values, 245
Stack panel child element alignment, 139-140
Static extension, 98
switch statement
code readability, 288
object instance global in scope, 287
TabControl control, 254-255
TabItem element, 255
Tree control example
C# and XAML, 115-119
Label content value, setting, 121

XAML, 119
TreeView control, 257
Button control event handler, 124
child object management, 126-128
Label Content property, 125
ProgressBar control event handler, 124
XAML for entire control, 125
TreeViewItem control, 258
UIElement embedding, 106
ViewBox control, 143
windows
state changes, handling, 484-485
viewing, 485
Wrap panel example, 140
XAML
button application example, 23
button declaration, 92
views, 36
ListView control, 233
data templates, 237-238
four column example, 236
hidden default view, 235
ListViewItems, 234-238
live artboard view, 33
LoadComponent( ) method, 469
LoadedBehavior property (MediaElement), 526
loading external skin files, 468, 471-473
code listing, 470
ComboBox selection event, 471, 473
Local Styles tab (Asset Library), 211
locking elements, 70
logic layer, 20-21

M

Make Button tool, 188-190
managing UIElement embedding, 106-107
child controls, adding, 113-115
example listing, 106
positioning, 107-110
TreeView control example, 115
Window properties, customizing, 110-113
Margin property (Grid), 113, 134
margins, 134
markup, 18
markup extensions (XAML), 98-101
adding brushes to Windows resource collection, 98
Binding, 99-101
creating new grids during runtime, 99
Static, 98
material properties (3D objects), 516-517
MaxDropDownHeight property (comboboxes), 222
media duration, checking, 541
media failure errors, 540
media formats, 524
media player project, 523
button state, 535
event handlers, 533-535
media duration, checking, 541
media failure errors, 540
MediaPlayerController event handlers, 538-540
MediaPlayerController.xaml.cs, 548-553
OpenFileDialog control with user input, 532-533
playback position slider, 537
player controls, 528-532
completed design, 531
menu item, creating, 532
names, 529
required, 530
row/column definitions, 529
viewing, 531
safe media control, 542-543
user playback position changes, 537-538
volume, 536
Window1.xaml.cs, 543-548
XAML requirements, 524
creating, 524
MediaElement properties, 525-527

Media tab (Asset Library), 211

MediaElement control

event handlers, 533-535

properties, 525-527

MediaPlayerController class, 538-540

MediaPlayerController.xaml.cs file, 548-553

menu items, building, 446-452

MergedDictionaries collection, skins, 462

elements, customizing, 465

entries, deleting, 466

MergedDictionaries property from sample App.xaml file, 462

resources, swapping, 463-465

methods, 289

Add( ), 113, 332

AddChild( ), 95

AddHandler( ), 308-309

amount of, 293

Begin( ), 434

calling, 290-293

example listing, 292-293

names, 291

public, 291

this keyword, 291

CanExecute( ), 438

CanExecuteChanged( ), 438

comments, 293

Execute( ), 438

FindResource( ), 434

Freeze( ), 332

FromRgb( ), 332

LoadComponent( ), 469

mathematical symbols, 298

overloading, 290

refactoring, 290

returning, 292

SetPlayerControlState( ), 535

SetPlayerPosition( ), 537

Show( ), 289

ShowDialog( ), 485

try-catch statement, 295-297

TryFindResource( ), 435

TryPlayMedia( ), 542-543

two blocks of code with same functionality example, 289

user input examples, 294-297

ValidateColorValue( ), 398-399

ValidateWidth( ), 297

Width property value, 297

mini-language syntax, 153-154

miscellaneous property category (Properties panel), 84-87

motion paths, 339-344

MouseMove event, testing, 206-207

multiple gradient stops, 178

multiple windows, creating, 480

N

names

buttons, 121

elements, 121

Grid, 121

labels, 121

media player controls, 529

methods, 291

progressbar, 121

resources, 73, 202

tunneling events, 203

windows, 480

namespaces (classes), 280

navigation elements (Vista), 12

Near Clipping Plane property, 518

nesting elements

child elements, 221

XAML, 21, 94

.NET

classes, XAML representations, 21

code, adding to XAML, 43-46

equivalent to XAML button declaration listing, 93

Framework, 13

trees, 114

New Project dialog box, 32
next generation hardware, 4
next generation user interfaces, 4-5

O

Object class, 490
object data type, 271
objects, 350. *See also* graphics
  3D
    camera properties, 518
    importing, 515-517
    light properties, 518, 521
    materials properties, 516
    Viewport3D element, 511
  BitmapImage, 230
  delegates, 299
  properties (XAML), 97-98
  transformations, 337
  vector objects, creating, 350-356
Objects and Timelines panel, 68-69
  elements
    activating, 71
    selecting, 71
    locking/unlocking, 70
    parent elements, 71
    viewing/hiding, 70
  playhead position, 70
  Scope Up button, 71
  storyboards
    actions, 70
    closing, 70
    current, 69
    KeyFrames, adding, 70
    new, 70
    opening, 69
  timeline, 69-70
  XAML/Z-order arrangement, 70
OneTime data binding, 396
OneWay data binding, 396
OneWayToSource data binding, 397
OnKeyUp event handler (custom text box control), 500-501
opacity (gradient stops), 178
Opacity property (animations), 417
OpenFileDialog class, 532
OpenFileDialog control with user input, 532-533
opening
  storyboards, 69
  XAML packages, 452-457
Orientation property (Stack panel), 138
OuterConeAngle property (3D objects lighting), 520
OuterGlowBitmapEffect, 346
overloading methods, 290

P

Paint Bucket tool, 167
Pan tool, 151
panel-based containers/controls/constructs. *See* PBCs
panel-based UIElements tools, 149
Panel class, 132
panels, 63
  Asset Library, 88-89
  design/split view, 63
  Interaction, 65
  layout modes, 63
  Objects and Timelines, 68-69
    activating elements, 71
    element selection, 71
    KeyFrames, adding, 70
    locking/unlocking elements, 70
    parent elements, 71
    playhead position, 70
    Scope Up button, 71
    storyboards, 69-70
    timeline, 69-70

viewing/hiding elements, 70
XAML/Z-order arrangement, 70
Project, 74
Data category, 76
Files category, 74-75
Properties, 34, 76
appearance category, 79-80
assembly types, 76
brushes category, 77-79
case insensitive searches, 76
clearing search phrase, 77
common properties category, 81
event view mode, 77
layout category, 80
miscellaneous category, 84-87
property categories, 77
property view mode, 76
text category, 82, 84
Resources, 72-74
customizing, 73
filtering, 72
names, 73
previews, 73
scope levels, 73
Results, 87
Storyboard Actions, 72
Storyboard Picker, 71
triggers, 66
actions applied, 68
active, 67
adding/deleting items/actions, 68
defined, 68
deleting, 66
element selection, 68
event, 66
properties, 66-68
recording, 66
timelines, 68
types, 66

partial trust sandboxes, 15

PasswordBox control, 238-239

paths
motion paths, 339-344
smoothing, 162
tools, 149
Pen, 158-163
Pencil, 163-164

Patient_BINDER class, 409-412

Patient class, 408-409

PBCs (panel-based containers/controls/constructs), 131
Border element, 141-142
Canvas element, 137-138
positioning properties, 137
switching between Canvas and Grid modes, 134
Dock, 140-141
Grid element, 132-137
Alignment property, 134
creating, 133
layout constraints, 135
Margin property, 134
rows/columns, adding, 135-136
switching between Canvas and Grid modes, 134
Width and Height properties, 134
ScrollViewer control, 141
Stack, 138, 140
child element alignment example, 139-140
Orientation property, 138
switching between panels, 144
Uniform Grid, 142
ViewBox, 143-144
Wrap, 140

Pen tool, 158-163
closed complex shapes, 161
gel button, 179
smoothing paths, 162

Pencil tool, 163-164

Perspective Field of View (POV), 518

Placement property (Popup control), 266

playback position slider (media player), 537

playhead position, 70
point light properties (3D objects), 521
points (gel button, adding), 179
Popup control, 264-266
  custom text box control, 497-498
  defining, 264
  force closing, 265
  properties, 266
PopupAnimation property (Popup control), 266
Position property (MediaElement control), 526
Position (x,y,z) property, 518-520
positioning embedded UIElements, 107-110
POV (Perspective Field of View), 518
previewing resources, 73
primitive data types (C#), 270-273
  common, 271
  creating, 271-273
programming languages, 21
ProgressBar control, 240
  adding to TreeView control, 122
  Click Event handler, 124
  prefix naming convention, 121
Project panel, 74
  Data category, 76
  Files category, 74-75
properties, 277, 348
  3D objects
    camera, 518
    light, 518, 521
    materials, 516-517
  animations, 417
  AreaToApplyEffect, 348-349
  AreaToApplyEffectUnits, 348-350
  BitmapEffectInput, 348
  Brush class, 325-327
  Camera Type, 518
  Canvas positioning, 137
  Children (Add( ) method), 113
  ClipToBounds, 137
  CLR, 278
  comboboxes, 222
  dependency, 279-281
    backing field, 279
    enumerations, 285
    public property wrapper, 279
    values, assigning, 281
  Direction (x,y,z) property, 518-520
  Expander control, 261
  Far Clipping Plane, 518
  Grid, 112-113, 134
  GridSplitter control, 263
  images, 230
  Input, 350
  IsExpanded
    Expander control, 261
    TreeViewItem control, 259
  IsIndeterminate (ProgressBar control), 240
  IsSnapToTickEnabled, 246, 397
  Items (TabControl element), 255
  Label Content (TreeView control), 125
  MediaElement control, 525-527
  Near Clipping Plane, 518
  Orientation (Stack panel), 138
  Popup control, 266
  Position (x,y,z), 518-520
  Property Search box, 35
  public property wrappers, 279
  recording state, 67-68
  ScrollViewer control, 268
  SelectedIndex
    comboboxes, 222
    TabControl element, 255
  Slider control, 246
  Stretch
    images, 230
    MediaElement, 527
  StretchDirection
    images, 230
    MediaElement, 527
  TabControl element, 255
  TabIndex (controls), 232
  Tag, 386
  transformations, 334

TranslateX, 335
TranslateY, 335
TreeViewItem control, 259
triggers, 66-68, 428-432
Up Vector (x,y,z), 518
Width
Grid, 134
method example, 297
Window, 110-113
windows, 476-477
XAML, 21

Properties panel, 34, 76
appearance category, 79-80
assembly types, 76
brushes category, 77-79
case insensitive searches, 76
clearing the search phrase, 77
common properties category, 81
event view mode, 77
layout category, 80
miscellaneous category, 84-87
property categories, 77
property view mode, 76
text category, 82-84

Property Search box, 35
prototyping, 6
public methods, 291
public property wrappers, 279

Q–R

QuadraticAttenuation property (3D objects lighting), 520-521
RadialGradientBrush, 330-334
RadioButton control, 241
raising
events, 300
RoutedEvents, 307

Range property (3D objects lighting), 520-521
RDs (Resource Dictionaries), 7, 201
folders, 204
gel button control template, adding, 201-202
testing, 204-206

recent controls elements, 151
Recent tab (Asset Library), 212
recording
timelines, 69
triggers, 66

refactoring, 290
referencing files, 502
relative commands, 154
RenderTransform property (transformations), 334
repeat animation, 420
ResizeDirection property (GridSplitter control), 263
ResizeMode property (windows), 476
resizing Interaction Panel, 65
resolution (timelines), 70
Resource Dictionaries. *See* RDs
resources
customizing, 73
DrawingBrush, 229
files
linking, 204
managing, 7
filtering, 72
MergedDictionaries, 463-465
names, 73, 202
previewing, 73
scope levels, 73
skins
control style attributes, deleting, 466-468
customizations at runtime, 460-462
external skin files, loading, 468-473
MergedDictionaries collection, 462-466

Resources panel, 72-74
customizing, 73
filtering, 72

names, 73
previews, 73
scope levels, 73
Results panel, 87
returning methods, 292
reversing timelines, 433
RichTextBox control, 241-245
adding, 242
alignment, 242
element types, 243
FlowDocument element, 243-245
roles (XA), 17
rotation, 336-337
RoutedEvents, 198
CLR events, compared, 303
declaring, 306-307
event handler method, 308-309
gel button creation, 199-201
overview, 303
raising, 307
website, 310
window state, 483-484
rows (Grid), 135-136
RSS reader for twitter project, 49
adorners, 50
artboard, viewing, 52
data
binding, 53
structure, viewing, 53
templates, 55
elements, activating, 49
Grid element, 49
pink border padding, 51
position, 50
Label elements, 57
LayoutRoot element, 49
Properties panel, 58
refresh capability, adding, 59-60
required fields, 56
RowDefinition row, creating, 51
RSS URL, 52-53
twitter post content, 55

S

saving
Visual Studio applications, 41-42
XAML packages, 452-457
scaling objects, 337
scope
classes, 276-277
resources, 73
Scope Up button (Objects and Timelines panel), 71
scripting languages, 21
ScrollViewer control, 141, 266-268
creating, 267
properties, 268
ScrubbingEnabled property (MediaElement control), 526
segment points, deleting, 158
SelectedIndex property
comboboxes, 222
TabControl element, 255
selecting
element triggers, 68
languages, 13, 32
panels, 144
targets, 32
Selection tool (Viewport3D element), 513
SetPlayerControlState( ) method, 535
SetPlayerPosition( ) method, 537
shape tools, 149
Ellipse, 164-166
Line, 152-154
Show( ) method, 289
ShowDialog( ) method, 485
ShowInTaskbar property (windows), 476
ShowsPreview property (GridSplitter control), 263
Silverlight, 15-16
Simple Styles collection, 361
SizeToContent property (windows), 477
skewing objects, 337

skins
- control style attributes, deleting, 466-468
- customizing, 460-462
- external skin files, loading, 468, 471-473
  - code listing, 470
  - ComboBox selection event, 471-473
- MergedDictionaries collection, 462
  - elements, customizing, 465
  - entries, deleting, 466
  - MergedDictionaries, 463-466
  - MergedDictionaries property from sample App.xaml file, 462
  - resources, swapping, 463-465

Slider control, 245-246
- properties, 246
- values, customizing, 245

SmallChange property (Slider control), 246

smoothing
- gel button, 179
- paths, 162

Solid Color brush, 178, 328-330

Source property (MediaElement control), 526

Source Safe integration, 19

specular material property, 516

SpeedRation property (MediaElement control), 526

spot light properties (3D objects), 521

square brackets [ ] in collections, 313

Stack panel, 138-140
- child element alignment example, 139-140
- Orientation property, 138

Static extension, 98

StaysOpen property (Popup control), 266

Storyboard Actions panel, 72

storyboard animations
- handoff animation, 432
- KeyFrames
  - adding, 418-419
  - erasing, 420-425
  - repeat animation, 420
- overview, 415
- timelines, 415-418
- triggers
  - definition, 425
  - event, 425-428
  - property, 428-432
  - recording, 429
  - timeline states, 428

Storyboard Picker, 71

Storyboard Picker dialog box, 69

storyboards
- actions, 70
- animations. *See* storyboard animations
- closing, 70
- current, 69
- new, 70
- opening, 69

Stretch property
- images, 230
- MediaElement, 527

StretchDirection property
- images, 230
- MediaElement, 527

string data type, 271

Stroke property (Brush class), 325-328

StrokeDashArray (Brush class), 326

StrokeDashCap property (Brush class), 327

StrokeDashOffset property (Brush class), 327

strokes
- brushes, 178, 325-328
- lines, 152

StrokeThickness property (Brush class), 327

strongly typed generic collections, 311

styles
- controls, 461
- creating, 369-371
- editing, 367-369
- Simple Styles collection, 361
- style templates
  - applying styles across projects, 374-376
  - creating new styles, 369-371
  - definition, 366
  - editing existing styles, 367-369

example, 366-367
FocusVisualStyle, 370-374
switch statement, 286-289
switching between windows in code, 480
event handler error dialog box, 482
Label element, adding, 481
new child elements, 481
reloading the application in Visual Studio, 482
RoutedEvent for window state, 483-484
state change, handling, 484-485
viewing windows, 485
syntax (XAML)
child elements, 94-96
elements, 92-93
events, 97-98
markup extensions, 98-101
nested elements, 94
object properties, 97-98

T

Tab key, 64
TabControl control
HeaderedContentControl control, 254-256
properties, 255
TabItems, adding, 255
TabIndex property (controls), 232
TabItems control, 255
tabs (Asset Library)
Controls, 210-211
Custom Controls, 211
Local Styles, 211
Media, 211
Recent, 212
TabStripPlacement property (TabControl element), 255
Tag property, 386
tags (XAML), 21
TargetName property (animations), 417
targets, selecting, 32
templates
child elements, 250
control
applying, 364-365
creating, 201, 360-361
definition, 360
editing, 361-363
gel button, creating, 201-202
overview, 360
testing, 204-206
data
creating, 378-388
data sources, 376-378
overview, 376
overview, 359
style templates
applying styles across projects, 374-376
creating new styles, 369-371
definition, 366
editing existing styles, 367-369
example, 366-367
FocusVisualStyle, 370-374
TestButtonClicked RoutedEvent, 306-307
testing
custom text box control, 501-503
gel button example, 203
MouseMove event, 206-207
Resource Dictionary/control template, 204-206
text-based UIElements tools, 149
text box control (custom), 492
complete control listing, 504-508
creating, 492
dictionary
functionality, 495
dynamic history, 498
filtering, 499-500
ListBox control, 496-497
new class, creating, 494
OnKeyUp event handler, 500-501
Popup control, 497-498
testing, 501-503
TextBox class inheritance, 494

text category (Properties panel), 82-84
TextChanged event handlers, 397-398
this keyword, 291
TickFrequency property (Slider control), 246
TickPlacement property (Slider control), 246
timelines, 70
  actions, 68
  animating elements, 190-191
  controlling in code, 434-436
  duplicating, 433
  overview, 415-418
  playhead position, 70
  recording status, 69
  resolution, 70
  reversing, 433
  snapping, 70
  states, 428
  triggers, 68
  zooming, 70
Toolbox, 147
  common controls UIElements, 150
  panel-based UIElements, 149
  path, 149
  recent controls elements, 151
  shape, 149
  text-based UIElements, 149
  usability, 148
tools
  Brush Transform, 181-182
  Camera Orbit, 514
  common controls UIElements, 150
  Direct Selection, 179
  Expression Blend, 6
  Expression Design, 6
  Make Button, 188-190
  panel-based UIElements, 149
  path, 149
  Pen tool, 158-163
    closed complex shapes, 161
    gel button, 179
    smoothing paths, 162
  Pencil tool, 163-164
  recent control elements, 151
  Selection, 513
  shape, 149
    Ellipse, 164-166
    Line, 152-154
  text-based UIElements, 149
  usability, 148
    Brush Transform, 167-168
    Camera Orbit, 168-173
    Direct Selection, 154-158
    Eyedropper, 166-167
    Paint Bucket, 167
    Pan, 151
    Zoom, 152
  Visual Studio, 6
  ZAM3D, 169
Top property (windows), 476
TopMost property (windows), 477
transformations
  center points, 337-338
  definition, 334
  flip, 339
  LayoutTransform property, 334
  RenderTransform property, 334
  rotation, 336-337
  scale, 337
  skew, 337
  translation, 335-336
TranslateX property, 335
TranslateY property, 335
translation, 335-336
transparency (windows), 477-480
trees (.NET versus WPF), 114
TreeView control example, 115
  Button control, adding, 123
  C#, XAML listing, 115-119
  child object management, 126-128
  Click Event handler
    Button control, 123
    ProgressBar control, 124

code behind file, 119
ColumnDefinitionCollection Editor, 122
Label Content property, 125
Label control
adding, 122
content value, setting, 120
ProgressBar control, adding, 122
TreeViewItem Items content value, setting, 121-122
XAML
entire control listing, 125
representation listing, 119

TreeView controls, 257

TreeViewItem control, 258-259
creation listing, 258
properties, 259

triggers, 66
actions applied, 68
active, 67
adding/deleting items/actions, 68
defined, 68, 425
deleting, 66
element selection, 68
event, 66
event triggers, 425-428
gel button animated glow, 194-196
properties, 68
recording state, 67-68
triggers, 66-68, 428-432
recording, 66, 429
timelines
actions, 68
selector, 68
states, 428
types, 66

try-catch statements, 295-297

try keyword, 295

TryFindResource( ) method, 435

TryPlayMedia( ) method, 542-543

tunneling events, 203

twitter RSS reader project, 49
adorners, 50
artboard, viewing, 52
data
binding, 53
structure, viewing, 53
templates, 55
elements, activating, 49
Grid
element, 49
pink border padding, 51
position, 50
Label elements, 57
LayoutRoot element, 49
Properties panel, 58
refresh capability, adding, 59-60
required fields, 56
RowDefinition row, creating, 51
RSS URL, 52-53
twitter post content, 55
twitter website, 48

two-way binding, 396-397

types
projects, 46-47
triggers, 66

U

UI process development, 6

UIElements, embedding, 106-107
child controls, adding, 113-115
example listing, 106
positioning, 107-110
TreeView control example, 115
Window properties, customizing, 110-113

underscores (_), resource names, 202

undocking Interaction Panel, 65

Uniform Grid control, 142

UnloadedBehavior property (MediaElement control), 527

unlocking elements, 70

unnamed combobox items, creating, 223-225

Up Vector (x,y,z) property, 518

usability tools, 148
- Brush Transform, 167-168
- Camera Orbit, 168-173
  - creating objects, 171-172
  - listing, 169-171
- Direct Selection, 154-158
- Eyedropper, 166-167
- Paint Bucket, 167
- Pan, 151
- Zoom, 152

UserControl level (custom controls), 491

users
- business mindset, 8-10
- interfaces
  - next generation, 4-5
  - visual perception, 9
- media player input, 532-533

V

ValidateColorValue( ) method, 398-399

ValidateWidth( ) method, 297

validating data, 397-400

ValidationRule class, 400

variables, declaring in C#, 270

vector graphics
- bitmaps, compared, 12
- creating, 350-356

VerticalOffset property (Popup control), 266

VerticalScrollBarVisibility property (ScrollViewer control), 268

ViewBox control, 143-144

viewing XAML elements, 36-37

Viewport3D element, 511
- 3D mesh, creating, 511-513
- Camera Orbit tool, 514
- Selection tool, 513

views
- live artboard, 33
- panels, 63

Vista, 10
- Bread Crumb controls, 10
- File Explorer, 10
- navigation elements, 12
- Windows XP similarities, 10

visual asset creation, 7

visual perception, 9

Visual Studio, 6
- applications, saving, 41-42
- C# primitive data types, creating, 271-273
- classes, creating, 275
- design environment, 13
- Expression Blend, compared, 17
- IntelliSense functionality, 120
- project types, 46-47
- versions, 41
- XAML
  - compatibility, 39
  - interactivity example, 39-42

volume (media player), 536

Volume property (MediaElement control), 526

W

websites
- media formats, 524
- RoutedEvents, 310
- twitter, 48
- XAML Express, 173

Width property
- Grid, 112, 134
- method example, 297

Window class, declaring, 37

Window properties, customizing, 110-113

Window1.xaml.cs file, 543-548

windows, 475
  multiple, 480
  names, 480
  properties, 476-477
  switching between in code, 480
    event handler error dialog box, 482
    Label element, adding, 481
    new child elements, 481
    reloading the application in Visual Studio, 482
    RoutedEvent for window state, 483-484
    state change, handling, 484-485
    viewing windows, 485
  transparency, configuring, 477-480
Windows Presentation Foundation. *See* WPF
Windows XP and Vista similarities, 10
WindowStartupLocation property (windows), 477
WindowState property (windows), 477
WindowStyle property (windows), 476
workflow benefits, 27-28
workspace, cycling, 36
WPF (Windows Presentation Foundation), 12-15
  controls
    ContentControl, 246-250, 254
    Expander, 259-261
    GridSplitter, 262-263
    HeaderedContentControl, 254-256
    HeaderedItemsControl, 256-259
    Popup, 264-266
    ScrollViewer, 266-268
  document commands, 439-440
  input commands, 438-439
  Silverlight, 15-16
  trees, 114
Wrap panel, 140

X

XA (XAML architect), 7, 17
XAML, 13
  architect, 7
  button application example, 22
    C#, 22
    XAML code, 23
  elements, 92-93
    child, 94-96
    nesting, 94
    viewing, 36-37
  event handlers, 21
  events, 97-98
  Express website, 173
  Hello World application with label control, 38
  history, 91-92
  integration, 23-24
  layers
    graphical UI, 19-20
    logic, 20-21
  listings
    adding brush to Window's resource collection, 98
    Binding extension, 99-101
    brush suggestion table sample one, 176
    brush suggestion table sample two, 178
    brush suggestion table sample three, 181
    brush suggestion table sample four, 181
    brush suggestion table sample five, 182
    brush suggestion table sample six, 182
    brush suggestion table sample seven, 184
    brush suggestion table sample eight, 184
    brush suggestion table sample nine, 193
    button declaration, 92
    C# code in XAML, 44-45

complex button shapes, 185-188
creating new grids during runtime, 99
embedded listbox, 95-96
gel button LayoutRoot element, 203
nested elements, 94, 97
RichTextBox controls, adding, 242
Static extension, 98
Tree control example, 115-119
TreeView control entire control, 125
views, 36
markup extensions, 98-101
adding brush to Windows resource collection, 98
Binding, 99-101
creating new grids during runtime, 99
Static, 98
media player requirements, 524
creating, 524
MediaElement properties, 525-527
nested elements, 21
.NET code, adding, 43-46
Notepad application
creating, 443-445
menu items, building, 446-452
overview, 442
XAML packages, saving and opening, 452-457
object properties, 97-98
overview, 18-19
packages, 452-457
properties, 21
representations, 21
Source Safe integration, 19
tags, 21
vector images, 225
Visual Studio
compatibility, 39
interactivity example, 39-42

Y–Z

ZAM3D tool, 169
zero-based collections, 310
Zoom tool, 152
zooming in/out, 33, 70
Z-Order, 108, 132