

Adam Nathan



Full Color

Figures and code appear as they do in Visual Studio.

XAML

UNLEASHED

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Adam Nathan

XAML

UNLEASHED

SAMS

800 East 96th Street, Indianapolis, Indiana 46240 USA

XAML Unleashed

Copyright © 2015 by Pearson Education, Inc.

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

ISBN-13: 978-0-672-33722-2

ISBN-10: 0-672-33722-3

Library of Congress Control Number: 2014953616

Printed in the United States of America

First Printing December 2014

Trademarks

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

Warning and Disclaimer

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

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

EDITOR-IN-CHIEF

Greg Wiegand

EXECUTIVE EDITOR

Joan Murray

DEVELOPMENT EDITOR

Mark Renfrow

MANAGING EDITOR

Kristy Hart

SENIOR PROJECT EDITOR

Betsy Gratner

INDEXER

Tim Wright

PROOFREADER

Chuck Hutchinson

TECHNICAL EDITOR

Jeremy Likness

EDITORIAL ASSISTANT

Cindy Teeters

COVER DESIGNER

Mark Shirar

COMPOSITOR

Nonie Ratcliff

Contents at a Glance

Introduction	1
Part I The XAML Language	
1 What Is XAML?.....	5
2 Child Elements and Keywords	15
3 Mixing XAML with Code.....	23
4 Extending XAML	31
Part II Graphics	
5 Arranging Elements.....	47
6 Automatic Layout	67
7 2D Graphics	99
8 3D Graphics	131
Part III Controls	
9 Content Controls	197
10 Items Controls	207
11 Images	221
12 Text	253
13 Audio and Video	281
14 Other Controls	311
15 User Controls and Custom Controls	327

Part IV Exploiting XAML Features

- 16 Resources341
- 17 Data Binding.....357
- 18 Styles, Templates, and Visual States379
- 19 Animation405

Part V Advanced Features

- 20 Layout with Custom Panels447
- 21 Fun with XAML Readers and Writers465
- Index479

Table of Contents

Introduction	1	Part II Graphics	
Who Should Read This Book?	2	5 Arranging Elements	47
Code Examples	2	Controlling Size	48
How This Book Is Organized	2	Controlling Position	52
Conventions Used in This Book	4	Applying 2D Transforms	55
		Applying 3D Transforms	63
		Summary	66
Part I The XAML Language		6 Automatic Layout	67
1 What Is XAML?	5	Canvas	68
Elements and Attributes	6	StackPanel	71
Namespaces	9	DockPanel	72
Property Elements	11	Grid	75
Summary	13	VariableSizedWrapGrid and WrapPanel	84
2 Child Elements and Keywords	15	Primitive Panels	87
Children of Object Elements	15	Handling Content Overflow	89
XAML Keywords	19	Summary	97
Summary	22	7 2D Graphics	99
3 Mixing XAML with Code	23	Shapes	99
Loading and Parsing XAML at Runtime	23	Geometries	107
Compiling XAML	26	Brushes	115
Summary	30	Summary	129
4 Extending XAML	31	8 3D Graphics	131
Type Converters	31	Getting Started with 3D Graphics	131
Using Arbitrary Types in XAML	34	Cameras and Coordinate Systems	135
Markup Extensions	38	Transform3D	147
Some Notes About XAML2009	40	Model3D	157
Summary	45	Visual3D	181
		Viewport3D	186
		2D and 3D Coordinate System Transformation	187
		Summary	195

Part III Controls

9 Content Controls	197
Button	198
HyperlinkButton	199
RepeatButton	200
ToggleButton	201
CheckBox	201
RadioButton	202
ToolTip	203
Summary	205
10 Items Controls	207
Items in the Control	208
Items Panels	209
ComboBox	212
ListBox	214
ListView	215
GridView	219
Summary	220
11 Images	221
The Image Element	222
Multiple Files for Multiple Environments	231
Decoding Images	236
Encoding Images	244
Summary	252
12 Text	253
TextBlock	253
RichTextBlock	265
TextBox	270
RichEditBox	276
PasswordBox	279
Summary	279

13 Audio and Video	281
Playback	281
Capture	292
Transcoding	303
Summary	308
14 Other Controls	311
Range Controls	311
Popup	314
Hub	316
DatePicker	322
TimePicker	323
ProgressRing	324
ToggleSwitch	325
Summary	326
15 User Controls and Custom Controls	327
Creating a User Control	328
Creating a Custom Control	331
Summary	340

Part IV Exploring XAML Features

16 Resources	341
Binary Resources	341
Logical Resources	348
Summary	355
17 Data Binding	357
Introducing Binding	357
Controlling Rendering	366
Customizing the View of a Collection	374
Summary	378

18	Styles, Templates, and Visual States	379	21	Fun with XAML Readers and Writers	465
	Styles	380		System.Xaml Overview	465
	Templates	386		The Node Loop	468
	Visual States	395		Reading XAML	469
	Summary	404		Writing to Live Objects	473
19	Animation	405		Writing to XML	475
	Theme Transitions	406		XamlServices	476
	Theme Animations	417		Summary	478
	Custom Animations	422		Index	479
	Custom Keyframe Animations	435			
	Easing Functions	439			
	Manual Animations	444			
	Summary	446			

Part V Advanced Features

20	Layout with Custom Panels	447
	Communication Between Parents and Children	448
	Creating a SimpleCanvas	451
	Creating a SimpleStackPanel	455
	Creating a UniformGrid	458
	Summary	463

About the Author

Adam Nathan is a principal software architect for Microsoft, a best-selling technical author, and a prolific developer of apps for Windows. He introduced XAML to countless developers through his books on a variety of Microsoft technologies. Currently a part of the Windows team, Adam has previously worked on Visual Studio and the Common Language Runtime. He was the founding developer and architect of Popfly, Microsoft's first Silverlight-based product, named by *PCWorld* as one of its year's most innovative products. He is also the founder of PINVOKE.NET, the online resource for .NET developers who need to access Win32. His apps have been featured on Lifehacker, Gizmodo, ZDNet, ParentMap, and other enthusiast sites.

Adam's books are considered required reading by many inside Microsoft and throughout the industry. Adam is the author of *Windows 8.1 Apps with XAML and C# Unleashed* (Sams, 2013), *101 Windows Phone 7 Apps* (Sams, 2011), *Silverlight 1.0 Unleashed* (Sams, 2008), *WPF 4.5 Unleashed* (Sams, 2013), *.NET and COM: The Complete Interoperability Guide* (Sams, 2002), and several others. You can find Adam online at www.adamnathan.net or @adamnathan on Twitter.

Dedication

To Tyler and Ryan.

Acknowledgments

I'd like to give special thanks to Ashish Shetty, Tim Heuer, Mark Rideout, Jonathan Russ, Joe Duffy, Chris Brumme, Eric Rudder, Joan Murray, Loretta Yates, Betsy Gratner, Bill Chiles, Valery Sarkisov, Dwayne Need, Daniel Lehenbauer, Andy Sterland, Tim Rice, and Michelle McCarthy. As always, I thank my parents for having the foresight to introduce me to Basic programming on our IBM PCjr when I was in elementary school. That small decision changed the entire course of my life.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@sampublishing.com

Mail: Joan Murray
 Executive Editor
 800 East 96th Street
 Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

INTRODUCTION

Ever since XAML was publicly introduced in 2003 (as part of the framework that would eventually be named Windows Presentation Foundation), it has gotten considerable attention for the ways in which it revolutionizes the process of creating software—especially for traditional Windows programmers. It’s relatively easy to create fun, useful, and shareable XAML samples that demonstrate all kinds of techniques that are difficult to accomplish in other technologies.

Over the years, Microsoft has shipped many (*too many*) UI frameworks based on XAML that are extremely similar but not identical. The introduction of universal Windows apps hopes to consolidate people’s development efforts on a single technology, but there’s still plenty of WPF and Silverlight development out in the real world.

That’s where this book comes in. It examines the XAML language common to a number of technologies, although many of its examples are based on the latest XAML-based UI framework. I wrote this book with the following goals in mind:

- To provide a solid grounding in the underlying concepts, in a practical and approachable fashion
- To answer the questions most people have when learning XAML and to show how commonly desired tasks are accomplished
- To be an authoritative source, thanks to input from members of the teams involved with XAML over the years
- To be an easily navigated reference that you can constantly come back to

In This Chapter

- Who Should Read This Book?
- Code Examples
- How This Book Is Organized
- Conventions Used in This Book

Who Should Read This Book?

This book is for software developers who are interested in creating user interfaces for Windows. Regardless of whether you're creating line-of-business applications, consumer-facing applications, or reusable controls, and no matter what types of devices you're targeting, this book contains a lot of content that helps you get the most out of XAML. It's designed to be understandable even for folks who are new to Microsoft technologies. Examples in this book appear in XAML and C#.

Code Examples

The source code for examples in this book can be downloaded from www.informit.com/title/9780672337222. (Click the Downloads tab.)

How This Book Is Organized

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

Part I: The XAML Language

This part includes the following chapters:

- Chapter 1, "What Is XAML?"
- Chapter 2, "Child Elements and Keywords"
- Chapter 3, "Mixing XAML with Code"
- Chapter 4, "Extending XAML"

This part of the book delves into the syntax of XAML and examines it broadly across its applications.

Part II: Graphics

This part includes the following chapters:

- Chapter 5, "Arranging Elements"
- Chapter 6, "Automatic Layout"
- Chapter 7, "2D Graphics"
- Chapter 8, "3D Graphics"

Part II equips you with the knowledge to assemble and arrange controls (and other elements) in a user interface.

Part III: Controls

This part includes the following chapters:

- Chapter 9, “Content Controls”
- Chapter 10, “Items Controls”
- Chapter 11, “Images”
- Chapter 12, “Text”
- Chapter 13, “Audio and Video”
- Chapter 14, “Other Controls”
- Chapter 15, “User Controls and Custom Controls”

Part III provides a tour of controls built into various XAML-based UI frameworks. There are many that you’d expect to have available, plus several that you might not expect. It ends with details on how to create your own controls.

Part IV: Exploiting XAML Features

This part includes the following chapters:

- Chapter 16, “Resources”
- Chapter 17, “Data Binding”
- Chapter 18, “Styles, Templates, and Visual States”
- Chapter 19, “Animation”

This part of the book covers the XAML features that typically get the most attention. They enable you to create a stunning experience, and/or greatly enhance the development process.

Part V: Advanced Features

This part includes the following chapters:

- Chapter 20, “Layout with Custom Panels”
- Chapter 21, “Fun with XAML Readers and Writers”

The topics covered in Part V are relevant for advanced developers, or just curious people who want to delve into some lesser-known functionality.

Conventions Used in This Book

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

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

When a line of code is too long to fit on a line in the printed book, a code-continuation arrow (➡) is used.

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



What is a FAQ sidebar?

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

Digging Deeper Sidebars



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



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



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

This page intentionally left blank

Chapter 2

In This Chapter

→ Children of Object Elements

→ XAML Keywords

CHILD ELEMENTS AND KEYWORDS

A XAML file, like all XML files, must have a single root object element. Therefore, it should come as no surprise that object elements can support children that are more than just the property elements introduced in the preceding chapter. Property elements aren't true children as far as XAML is concerned.

This chapter examines the types of children that object elements can have. It also summarizes all the keywords in the XAML language namespace, although many of them are discussed in depth in the two chapters that follow.

Children of Object Elements

An object element can have three types of children:

- A value for a content property
- Collection items
- A value that can be type-converted to the object element

We'll look at the first two types of children now, but save the third type for Chapter 4, "Extending XAML."

The Content Property

Most classes designate a property (via a custom attribute) that should be set to whatever content is inside the XML element. This property is called the *content property*, and it is really just a convenient shortcut to make the XAML representation more compact.

Button's Content property is (appropriately) given this special designation, so the following Button:

```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  Content="OK"/>
```

could be rewritten as follows:

```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  OK
</Button>
```

Or, more usefully, this Button with more complex content:

```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
<Button.Content>
  <Rectangle Height="40" Width="40" Fill="Black"/>
</Button.Content>
</Button>
```

could be rewritten as follows:

```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Rectangle Height="40" Width="40" Fill="Black"/>
</Button>
```

There is no requirement that the content property must actually be called Content; classes such as ComboBox and ListBox use their Items property as the content property. For example, this ListBox:

```
<ListBox Width="100">
  <ListBox.Items>
    <ListBoxItem>1</ListBoxItem>
    <ListBoxItem>2</ListBoxItem>
    <ListBoxItem>3</ListBoxItem>
  </ListBox.Items>
</ListBox>
```

is equivalent to the following ListBox:

```
<ListBox Width="100">
  <ListBoxItem>1</ListBoxItem>
  <ListBoxItem>2</ListBoxItem>
  <ListBoxItem>3</ListBoxItem>
</ListBox>
```

Both produce the result in Figure 2.1.

Collection Items

XAML enables you to add items to the two main types of collections that support indexing: lists and dictionaries.

Lists

A *list* is any collection that implements `System.Collections.IList`, such as `System.Collections.ArrayList` or numerous collection classes defined by various frameworks. For example, the following XAML adds two items to a `ListBox` control whose `Items` property is an `ItemCollection` that implements `IList`:

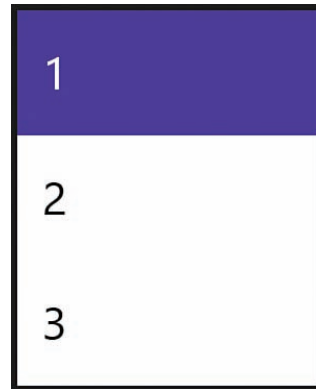


FIGURE 2.1 A simple `ListBox` has three children in its `Items` collection.

```
<ListBox xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
<ListBox.Items>
  <ListBoxItem Content="Item 1"/>
  <ListBoxItem Content="Item 2"/>
</ListBox.Items>
</ListBox>
```

This is equivalent to the following code:

C# (WPF and Silverlight):

```
System.Windows.Controls.ListBox listBox = new System.Windows.Controls.ListBox();
System.Windows.Controls.ListBoxItem item1 =
    new System.Windows.Controls.ListBoxItem();
System.Windows.Controls.ListBoxItem item2 =
    new System.Windows.Controls.ListBoxItem();
item1.Content = "Item 1";
item2.Content = "Item 2";
listbox.Items.Add(item1);
listbox.Items.Add(item2);
```

C# (Windows Store and Universal Apps):

```
Windows.UI.Xaml.Controls.ListBox listBox =
    new Windows.UI.Xaml.Controls.ListBox();
Windows.UI.Xaml.Controls.ListBoxItem item1 =
    new Windows.UI.Xaml.Controls.ListBoxItem();
Windows.UI.Xaml.Controls.ListBoxItem item2 =
    new Windows.UI.Xaml.Controls.ListBoxItem();
item1.Content = "Item 1";
```

```

item2.Content = "Item 2";
listbox.Items.Add(item1);
listbox.Items.Add(item2);

```

Furthermore, because `Items` is the content property for `ListBox`, you can shorten the XAML even further, as follows:

```

<ListBox xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <ListBoxItem Content="Item 1"/>
  <ListBoxItem Content="Item 2"/>
</ListBox>

```

In all these cases, the code works because `ListBox`'s `Items` property is automatically initialized to any empty collection object. If a collection property is initially `null` instead (and is read/write, unlike `ListBox`'s read-only `Items` property), you need to wrap the items in an explicit element that instantiates the collection. `ListBox` does not act this way, so an imaginary `OtherListBox` element demonstrates what this could look like:

```

<OtherListBox>
<OtherListBox.Items>
  <ItemCollection>
    <ListBoxItem Content="Item 1"/>
    <ListBoxItem Content="Item 2"/>
  </ItemCollection>
</OtherListBox.Items>
</OtherListBox>

```

Dictionaries

`ResourceDictionary` is a commonly used collection type in all XAML-based frameworks that you'll see more of in Chapter 16, "Resources." It implements `System.Collections.IDictionary`, so it supports adding, removing, and enumerating key/value pairs in procedural code, as you would do with a typical hash table. In XAML, you can add key/value pairs to any collection that implements `IDictionary`. For example, the following XAML adds two `Colors` to a `ResourceDictionary`:

```

<ResourceDictionary
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Color x:Key="1">White</Color>
  <Color x:Key="2">Black</Color>
</ResourceDictionary>

```

This leverages the XAML `key` keyword (defined in the secondary XML namespace), which is processed specially and enables us to attach a key to each `Color` value. (The `Color` type does not define a `key` property.) Therefore, the XAML is equivalent to the following code:

C# (WPF and Silverlight):

```
System.Windows.ResourceDictionary d = new System.Windows.ResourceDictionary();
System.Windows.Media.Color color1 = System.Windows.Media.Colors.White;
System.Windows.Media.Color color2 = System.Windows.Media.Colors.Black;
d.Add("1", color1);
d.Add("2", color2);
```

C# (Windows Store and Universal Apps):

```
Windows.UI.Xaml.ResourceDictionary d = new Windows.UI.Xaml.ResourceDictionary();
Windows.UI.Color color1 = Windows.UI.Colors.White;
Windows.UI.Color color2 = Windows.UI.Colors.Black;
d.Add("1", color1);
d.Add("2", color2);
```

Special Attributes Defined by the W3C

In addition to keywords in the XAML language namespace, XAML also supports two special attributes defined for XML by the World Wide Web Consortium (W3C): `xml:space` for controlling whitespace parsing and `xml:lang` for declaring the document's language and culture. The `xml` prefix is implicitly mapped to the standard XML namespace; see <http://www.w3.org/XML/1998/namespace>.

XAML Keywords

The XAML language namespace (<http://schemas.microsoft.com/winfx/2006/xaml>) defines a handful of keywords that must be treated specially by any XAML compiler or parser. Table 2.1 lists them all, along with the chapter that discusses each one further. Several of them require a specific context in order to be usable. For example, several are only valid for WPF.

TABLE 2.1 Keywords in the XAML Language Namespace, Assuming the Conventional `x` Namespace Prefix

Keyword	Valid As	Chapter	Meaning
<code>x:Array</code>	An element	4	Represents an array. An <code>x:Array</code> element's children are the elements of the array. It must be used with <code>x:Type</code> to define the type of the array.
<code>x:AsyncRecords</code>	Attribute on root element	3	Controls the size of asynchronous XAML-loading chunks.
<code>x:Arguments</code>	Attribute on or element inside any element	4	Specifies an argument (or multiple arguments in the element syntax) to be passed to the element's constructor. When used with <code>x:FactoryMethod</code> , specifies argument(s) for the factory method.
<code>x:Boolean</code>	An element	4	Represents a <code>System.Boolean</code> .
<code>x:Byte</code>	A XAML2009 element	4	Represents a <code>System.Byte</code> .
<code>x:Char</code>	A XAML2009 element	4	Represents a <code>System.Char</code> .
<code>x:Class</code>	Attribute on root element	3	Defines a class for the root element that derives from the element type, optionally prefixed with a namespace.
<code>x:ClassAttributes</code>	Attribute on root element and must be used with <code>x:Class</code>	N/A	Contains attributes relevant for Windows Workflow Foundation activities.
<code>x:ClassModifier</code>	Attribute on root element and must be used with <code>x:Class</code>	3	Defines the visibility of the class specified by <code>x:Class</code> (which is <code>public</code> by default). The attribute value must be specified in terms of the procedural language being used (for example, <code>public</code> or <code>internal</code> for C#).
<code>x:Code</code>	Element anywhere in XAML but must be used with <code>x:Class</code>	3	Embeds procedural code to be inserted into the class specified by <code>x:Class</code> .
<code>x:ConnectionId</code>	Attribute	N/A	Not for public use.
<code>x:Decimal</code>	A XAML2009 element	4	Represents a <code>System.Decimal</code> .
<code>x:Double</code>	An element	4	Represents a <code>System.Double</code> .
<code>x:FactoryMethod</code>	Attribute on any element	4	Specifies a static method to be called to retrieve the element instance instead of its constructor.

Keyword	Valid As	Chapter	Meaning
<code>x:FieldModifier</code>	Attribute on any nonroot element but must be used with <code>x:Name</code> (or equivalent)	3	Defines the visibility of the field to be generated for the element (which is internal by default). As with <code>x:ClassModifier</code> , the value must be specified in terms of the procedural language (for example, <code>public</code> , <code>private</code> , ... for C#).
<code>x:Int16</code>	A XAML2009 element	4	Represents a <code>System.Int16</code> .
<code>x:Int32</code>	An element	4	Represents a <code>System.Int32</code> .
<code>x:Int64</code>	A XAML2009 element	4	Represents a <code>System.Int64</code> .
<code>x:Key</code>	Attribute on an element whose parent implements <code>IDictionary</code>	2	Specifies the key for the item when added to the parent dictionary.
<code>x:Members</code>	Inside an <code>Activity</code> class	4	Defines additional members for the root class specified by <code>x:Class</code> .
<code>x:Name</code>	Attribute on any nonroot element but must be used with <code>x:Class</code>	3	Chooses a name for the field to be generated for the element, so it can be referenced from procedural code.
<code>x:Null</code>	A property element or attribute value	4	Represents a null reference.
<code>x:Object</code>	A XAML2009 element	4	Represents a <code>System.Object</code> .
<code>x:Property</code>	Inside an <code>x:Members</code> element	4	Defines a new property.
<code>x:Reference</code>	A property element or attribute value	17	A reference to a named element.
<code>x:Shared</code>	Attribute on any element in a <code>ResourceDictionary</code> , but only works if XAML is compiled	16	A WPF-specific concept that doesn't really belong in the XAML language namespace.
<code>x:Single</code>	A XAML2009 element	4	Represents a <code>System.Single</code> .
<code>x:Static</code>	A property element or attribute value	4	References any static property, field, constant, or enumeration value.
<code>x:String</code>	An element	4	Represents a <code>System.String</code>
<code>x:Subclass</code>	Attribute on root element and must be used with <code>x:Class</code>	3	Specifies a subclass of the <code>x:Class</code> class that holds the content defined in XAML, optionally prefixed with a namespace (used with languages without support for partial classes).
<code>x:SynchronousMode</code>	Attribute on root element	3	Specifies whether the XAML content is allowed to be loaded asynchronously.

Keyword	Valid As	Chapter	Meaning
<code>x:TimeSpan</code>	A XAML2009 element	4	Represents a <code>System.TimeSpan</code> .
<code>x:Type</code>	A property element or attribute	4	Represents a <code>System.Type</code> , just like the <code>typeof</code> operator in C#.
<code>x:TypeArguments</code>	Attribute on any element in XAML2009, or attribute on root element that must be used with <code>x:Class</code> in XAML2006	4	Makes the class generic (for example, <code>List<T></code>) with the specified generic argument instantiations (for example, <code>List<Int32></code> or <code>List<String></code>). Can be set to a comma-delimited list of generic arguments, with XML namespace prefixes for any types not in the default namespace.
<code>x:Uid</code>	Attribute on any element	16	Marks an element with an identifier used for localization.
<code>x:Uri</code>	A XAML2009 element	4	Represents a <code>System.Uri</code> .
<code>x:XData</code>	Element used as the value for any property of type <code>IXmlSerializable</code>	17	An arbitrary XML data island that remains opaque to the XAML parser.

Summary

XAML excels at expressing deep hierarchies of objects concisely, thanks to the features described in this chapter.

The concept of an element's *content* is a crucial one in XAML, for more reasons than simply the syntactic shortcut enabled by content properties. The behavior of styling, the topic of Chapter 18, "Styles, Templates, and Visual States," and a large set of controls, covered in Chapter 9, "Content Controls," are centered around the notion of an element's content.

The ability to fill collections with items is leveraged throughout XAML-based frameworks. You'll see many examples in Chapter 10, "Items Controls," and Chapter 16, "Resources."

INDEX

Numerics

2D graphics

- 2D-to-3D coordinate system conversion
 - Visual3D.TransformToAncestor method, 191-195
 - Visual3D.TransformToDescendant method, 191-195
 - Visual.TransformToAncestor method, 187-191
- color brushes, 115-124
 - LinearGradientBrush, 116-123
 - RadialGradientBrush, 123-124
 - SolidColorBrush, 116
- drawing, 131-132
- Geometries, 107-115
 - GeometryGroup, 111-113
 - PathFigure, 108-111
 - PathSegment, 108-111
- Shapes, 99-106
 - Ellipse, 101
 - filling, 110
 - Line, 101-102
 - Path, 104
 - Polygon, 103-104
 - Polyline, 102-103
 - Rectangle, 100-101
 - Stroke property, 104-106
- tile brushes
 - DrawingBrush, 128-129
 - ImageBrush, 124-128
 - VisualBrush, 129

2D transforms, 55-63

- CompositeTransform, 61-62
- MatrixTransform, 62-63
- RotateTransform, 57-58
- ScaleTransform, 58-60
- SkewTransform, 60
- TransformGroup, 62
- TranslateTransform, 61

3D graphics, 131-135

- Cameras, 135-146
 - LookDirection property, 138-141
 - MatrixCamera, 147
 - OrthographicCamera, comparing with PerspectiveCamera, 144-147
 - Position property, 136-137
 - UpDirection property, 141-144
 - Z-fighting, 139
- GeometryModel3D, 165-178
 - Material, 165-172
- MeshGeometry3D, 172-178
 - normals, 174-177
 - Positions property, 172-174
 - seams, removing, 175
 - TextureCoordinates property, 177-178
 - TriangleIndices property, 174-175
- Model3D, Lights, 157-164
- Model3DGroup, 178-180
- Transform3D, 147-156
 - MatrixTransform3D class, 156
 - RotateTransform3D class, 153-156
 - ScaleTransform3D class, 150-153
 - TranslateTransform3D class, 149-150
- Viewport3D, 186-187
- Visual3D, 181-186
 - ModelVisual3D class, 181-182
 - UIElement3D class, 182-184
 - Viewport2DVisual3D, 184-186

3D transforms, 63-66

A

absolute positioning, 57

absolute sizing, 80

accessing binary resources

- in another assembly, 344
- from procedural code, 345-346
- at the site of origin, 344-345

ActualHeight property, 49

ActualWidth property, 49

AddDeleteThemeTransition, 412-414

adjusting camera settings, 299-301

alignment

- content alignment, 53-54
- HorizontalAlignment property, 52-53
- Stretch alignment, 52-53
 - interaction with ScaleTransform, 60
- VerticalAlignment property, 52-53

AmbientColor property (DiffuseMaterial), 169

AmbientLight, 163-164

AngleX property, 60

AngleY property, 60

animation

- custom animations, 422-427
 - controlling duration, 424-425
 - dependent versus independent animations, 423-424
- easing functions, 439-444
- keyframes
 - discrete keyframes, 438-439
 - easing keyframes, 439
 - linear keyframes, 435-436
 - spline keyframes, 436-437
- manual animations, 444-446
- property paths, 431-433
- Storyboards, tweaking, 433-434
- theme animations, 417-422
 - To property, 425-427
 - From property, 425-427
 - Storyboards, 417-419
 - tweaking, 421-422

- theme transitions, 406-416
 - AddDeleteThemeTransition, 412-414
 - applying to elements, 406-407
 - ContentThemeTransition, 410
 - EdgeUIThemeTransition, 410-411
 - EntranceThemeTransition, 408-409
 - PaneThemeTransition, 411-412
 - PopupThemeTransition, 409-410
 - ReorderThemeTransition, 415-416
 - RepositionThemeTransition, 414-415

total Timeline length, calculating, 429-430

app files

- local files, 346-347
- packaged files, 346
- roaming files, 347
- temporary files, 348

applying

- background color to cells, 82
- styles, 382, 385-386
- theme transitions to elements, 406-407

arrange step (layout), 450-451

arrays, declaring, 37

assigning automation ID to FrameworkElement, 215

asynchronous XAML processing, 25

attached properties, 68-69

attributes, XAML, 19

audio

- capturing, 302-303
- playback, 281-292
 - customizing, 283-284
 - effects, adding, 285-286
 - markers, 285
 - media content, 282-283
 - MediaPlayer, 288-292

Auto property (ScrollViewer), 92

automation ID, assigning to FrameworkElement, 215

AutoReverse property (Timeline), 428

autosizing (Grid panel), 80

avoiding logic in property wrappers, 338

B

BackEase function, 442

background color, applying to cells, 82

BAML (Binary Application Markup Language), 28

BeginTime property (Timeline), 427-428

behavior, creating

- for custom control, 331-332
- for user controls, 329

Bézier curves, 108

BGRA8, 229

binary resources

- accessing
 - in another assembly, 344
 - from procedural code, 345-346
 - at the site of origin, 344-345
- in Windows Store, 346-348
- in WPF, 341-344

Binding markup extension, 357-360

- in C#, 359-360
- Mode property, 361
- RelativeSource property, 359

binding to collections, 363-366

bitmap-based graphics, 130

BitmapDecoder class, 236-244

- getting pixel data, 237-240

BitmapEncoder class, 244-252

BitmapTransform class, 239-240

BounceEase function, 443

Brushes

- color brushes, 115-124
 - LinearGradientBrush, 116-123
 - RadialGradientBrush, 123-124
 - SolidColorBrush, 116
- tile brushes, 124-129
 - DrawingBrush, 128-129
 - ImageBrush, 124-128
 - VisualBrush, 129

built-in panels

- Canvas, 68-71, 83
- DockPanel, 72-75, 84

- Grid, 75-84
 - absolute sizing, 80
 - autosizing, 80
 - child layout properties, 80
 - percentage sizing, 82
 - proportional sizing, 81
- StackPanel, 71-72
 - mimicking with Grid, 83
- VariableSizedWrapGrid, 84-87
- WrapPanel, 84-87

built-in System data types, 37, 42

Button control, 7-8, 198-199

buttons, password reveal buttons, 279

C

C#, Binding markup extension, 359-360

cached composition, 130, 185

calculating

- final reflected color, 168
- total animation Timeline length, 429-430

camera settings, adjusting, 299-301

CameraCaptureUI

- capturing photos, 292-294
- capturing video, 295-296

Cameras, 135-146

- LookDirection property, 138-141
- MatrixCamera, 147
- OrthographicCamera, comparing with PerspectiveCamera, 144-147
- Position property, 136-137
- UpDirection property, 141-144
- Z-fighting, 139

Camera.Transform property, 143

CAML (Compiled Application Markup Language), 29

CanPlayType method, 282

Canvas class, 68-71, 83

CaptureElement, 296-303

- adjusting camera settings, 299-301
- capturing audio, 302-303
- capturing photos, 298-301
- capturing video, 301-302
- showing a preview, 296-298

capturing

- audio, 302-303
- photos, 292-294, 298-301
- video, 295-298, 301-302

cells, applying background color, 82

CenterX property, 57

CenterY property, 57

changing media file format, 306-307

CheckBox control, 201-202

child elements, 15-19

- collection items, 17-19
- dictionaries, 18-19
- lists, 17-18
- communication with parents
 - arrange step, 450-451
 - measure step, 448-450
- content property, 16-17
- stacking, 71-72
- XAML processing rules, 34
- Z order, 70

choosing

- decoders, 236
- encoding options, 245

CircleEase function, 443

classes, 47

- BitmapDecoder class, 236-244
- BitmapEncoder class, 244-252
- BitmapTransform class, 239-240
- Canvas class, 68-71, 83
- Control class
 - HorizontalAlignment property, 53-54
 - VerticalContentAlignment property, 53-54
- DockPanel class, 72-75, 84
- Drawing class, 107
- DrawingBrush class, 128-129
- Ellipse class, 101
- FrameworkElement class
 - FlowDirection property, 54-55
 - Height property, 48-50
 - Margin property, 50-51
 - Padding property, 50-51
 - Visibility property, 51-52
 - Width property, 48-50

- generics, support for in XAML2009, 41
- GeometryGroup class, 111-113
- Grid class, 75-84
 - absolute sizing, 80
 - autosizing, 80
 - child layout properties, 80
 - percentage sizing, 82
 - proportional sizing, 81
- ImageBrush class, 124-128
- Line class, 101-102
- LinearGradientBrush class, 116-123
- MatrixTransform3D class, 156
- ModelVisual3D class, 181-182
- partial class definitions, 29
- Path class, 104
- PathFigure class, 108-111
- PathSegment class, 108-111
- PlaneProjection class, 63
- Polygon class, 103-104
- Polyline class, 102-103
- RadialGradientBrush class, 123-124
- Rectangle class, 100-101
- RotateTransform3D class, 153-156
- ScaleTransform3D class, 150-153
- SolidColorBrush class, 116
- StackPanel, mimicking with Grid, 83
- StackPanel class, 71-72
- Style class, 380-386
- TextElement class, 261-264
- TextPointer class, 265
- TranslateTransform3D class, 149-150
- UIElement3D class, 182-184
- VariableSizedWrapGrid class, 84-87
- Viewport2DVisual3D class, 184-186
- VisualBrush class, 129
- VisualState class, 396
- WrapPanel class, 84-87
- WritableBitmap class, 228-231
- XamlServices, 476-478
- Click events, 198-199**
- clipping, 89-91**
- clr-namespace directive, 35-36**
- code-behind file, 27**
- “code-inside” feature (XAML), 30**
- Collapsed elements, 51**
- collection items, 17-19**
 - binding to, 363-366
 - customizing view of
 - grouping, 374-378
 - navigating views, 378
 - dictionaries, 18-19
 - lists, 17-18
- color brushes, 115-124**
 - LinearGradientBrush, 116-123
 - RadialGradientBrush, 123-124
 - SolidColorBrush, 116
- Color property (Material), 168**
- combining**
 - Materials, 172
 - Transform3Ds, 156
 - transforms
 - CompositeTransform, 61-62
 - MatrixTransform, 62-63
- ComboBox control, 212-214**
- commands**
 - Geometry string commands, 114
 - spaces, 115
- communication between parents and children**
 - arrange step, 450-451
 - measure step, 448-450
- comparing**
 - DateTime and DateTimeOffset, 241
 - frameworks, 7-8
 - Model3DGroup and ModelVisual3D, 182
 - OrthographicCamera and PerspectiveCamera, 144-147
 - property elements and object elements, 12
- compiling XAML, 26-30**
 - BAML, 28
 - code-behind file, 27
 - generated source code, 29-30
- CompositeTransform, 61-62**
- constructors**
 - GridLength, constructing, 82
 - XAML2009, 42-43

consuming

- custom controls, 335-336
- user controls, 330

content controls, 197

- Button control, 198-199
- CheckBox control, 201-202
- HyperlinkButton control, 199-200
- RadioButton control, 202-203
- RepeatButton control, 200
- ToggleButton control, 201
- ToolTip control, 203-205

content overflow, handling, 89-97

- clipping, 89-91
- scaling, 94-97
- scrolling, 91-94

content property, 16-17**ContentPresenter element, 390****ContentThemeTransition, 410****Control class**

- HorizontalAlignment property, 53-54
- VerticalContentAlignment property, 53-54

control points, 108**control templates, 386-394**

- hijacking existing properties, 393-394
- named elements, 404
- respecting target control's properties, 388-393
- Template property, setting inside a style, 394-395
- tweaking, 395

controlling

- origin of transforms, 57
- position
 - FlowDirection property, 54-55
 - HorizontalAlignment property, 52-53
 - VerticalAlignment property, 52-53
- rendering
 - data templates, 366-370
 - value converters, 370-374
- size
 - Height property, 48-50
 - Margin property, 50-51
 - Padding property, 50-51
 - Visibility property, 51-52
 - Width property, 48-50

controls**content controls**

- Button control, 7-8, 198-199
- CheckBox control, 201-202
- HyperlinkButton control, 199-200
- RadioButton control, 202-203
- RepeatButton control, 200
- ToggleButton control, 201
- ToolTip control, 203-205

custom controls

- behavior, creating, 331-332
- consuming, 335-336
- dependency properties, 336-340
- "lookless" appearance, 336-340
- user interface, creating, 332-335

items controls, 207-208

- ComboBox control, 212-214
- GridView control, 219-220
- item containers, 209
- items panels, 209-212
- ListBox control, 214-215
- ListView control, 215-219

ListBox control, 17-18**Range controls**

- DatePicker control, 322-323
- Hub control, 316-319
- HubSections control, 319-322
- Popup control, 314-316
- ProgressBar control, 312
- ProgressRing control, 324-325
- Slider control, 312-314
- TimePicker control, 323-324
- ToggleSwitch control, 325-326

ScrollViewer, 91-94, 96-97**text controls**

- PasswordBox control, 279
- RichEditBox control, 276-279
- RichTextBlock control, 265-270
- TextBlock control, 253-265
- TextBox control, 270-276

user controls

- consuming, 330
- creating, 328-330

visual states, 395-404

 responding to changes, 396-399

 visual transitions, 399-404

coordinate systems, 135-136. See also Cameras

2D-to-3D conversion

 Visual3D.TransformToAncestor method,
 191-195

 Visual3D.TransformToDescendant method,
 191-195

 Visual.TransformToAncestor method,
 187-191

 scaling along nonprincipal axes, 153

 world space, 135

 Z-fighting, 139

creating

 custom controls, 331-340

 translucent DiffuseMaterial, 168-169

 user controls

 behaviors, 329

 user interface, 328-329

custom animations, 422-427

custom controls, 331-340

 behavior, creating, 331-332

 consuming the control, 335-336

 dependency properties, 336-340

 dependent versus independent animations,
 423-424

 “lookless” appearance, 336-340

 user interface, creating, 332-335

custom type converters, 32-33

customizing

 collection views, 374-378

 ScrollViewer control, 91-93

CustomResource markup extension, 354

controlling rendering

 data templates, 366-370

 value converters, 370-374

customizing the data flow, 361

OneTime binding, 361

OneWay binding, 361

source properties, 358, 362-363

target properties, 358

TwoWay binding, 361

data templates, 366-370

data types, built-in System data types, 37

data virtualization, ListView control, 218

DateTime data type, 241

DateTimeOffset data type, 241

DatePicker control, 322-323

declaring arrays, 37

decoding images, 236-244

 choosing a decoder, 236

 getting pixel data, 237-240

 reading BitmapProperties from a decoder,
 242-243

 reading metadata, 240-244

defining properties in XAML2009, 44

dependency properties, 336-340

 hijacking, 393-394

dependent animations, 423-424

DesiredSize property, 49

dictionaries, 18-19

DiffuseMaterial, 166-169

DirectionalLight, 158-159

Disabled property (ScrollViewer), 92

discrete keyframes, 438-439

DockPanel class, 72-75, 84

DragItemThemeAnimation, 420

DragOverThemeAnimation, 420

drawing 2D graphics, 131-132

Drawing class, 107

DrawingBrush class, 128-129

DropTargetItemThemeAnimation, 421

duration of animations, controlling, 424-425

D

data binding, 357-360

 Binding markup extension in C#, 359-360

 binding to collections, 363-366

 binding to plain properties, 360

dynamic images

- generating with `RenderTargetBitmap`, 231
- generating with `WriteableBitmap`, 228-231

dynamic resources, 352**E****easing functions, 439-444**

- power easing functions, 440-441

easing keyframes, 439**EdgeUIThemeTransition, 410-411****effects, adding to audio/video playback, 285-286****ElasticEase function, 443****elements (XAML), 6-9**

- naming, 25-26
- type converters, 33-34
- Z order, 70

Ellipse class, 101**embedding UIElements, 266-267****EmissiveMaterial, 169-170****encoding images, 244-252**

- choosing encoding options, 245
- transcoding, 249-252
- writing metadata, 247-248
- writing pixel data, 246-247

EntranceThemeTransition, 408-409**escaping open curly braces {}, 39****EvenOdd filling, 110****event attributes, 7-8****event handlers**

- processing order, 8
- XAML2009, 44

event triggers, 419**events, MediaElement, 284-285****explicit Runs, 261****explicit sizing, 49****ExponentialEase function, 443****extending XAML**

- markup extensions, 38-40
- type converters, 31-34
 - custom type converters, 32-33
 - `LengthConverter`, 33

F**factory methods, 43-44****FadeInThemeAnimation, 419****FadeOutThemeAnimation, 419****files, referencing with URIs, 222-225****FillBehavior property (Timeline), 430****filling Shapes, 110****final reflected color, calculating, 168-169****FindName method, 25-26****fixed-function lighting, 172****flow direction, 52-55**

- alignment, 52-53
- content alignment, 53-54
- `FlowDirection` property, 54-55
- `RightToLeft`, 86

Footer property (ListView), 217**forcibly closing ToolTips, 203-205****FrameworkElement class**

- `FlowDirection` property, 54-55
- `Height` property, 48-50
- `HorizontalAlignment` property, 52-53
- `Margin` property, 50-51
- `Padding` property, 50-51
- `VerticalAlignment` property, 52-53
- `Visibility` property, 51-52
- `Width` property, 48-50

frameworks

- comparing, 7-8
- .NET, clr-namespace directive, 35-36
- WPF
 - binary resources, 341-344
 - static fields, referencing, 37-38

From property, 425-427
full generics support (XAML2009), 41
functions, easing functions, 439-444

G

generated source code, 29-30
generating dynamic images
 with RenderTargetBitmap, 231
 with WriteableBitmap, 228-231
generics, support for in XAML2009, 41
Geometries, 107-115
 GeometryGroup, 111-113
 PathFigure, 108-111
 PathSegment, 108-111
 representing as strings, 113-115
GeometryGroup class, 111-113
GeometryModel3D, 165-178
 Material, 165-172
 combining, 172
 DiffuseMaterial, 166-169
 EmissiveMaterial, 169-170
 SpecularMaterial, 170-172
gestures
 pinch-to-zoom, interactive zooming, 96-97
 press-and-hold, 217
GetPixelDataAsync method, 237-238
GetThumbnailAsync method, 240
Grid class, 75-84
 absolute sizing, 80
 autosizing, 80
 background color, applying to cells, 82
 child layout properties, 80
 percentage sizing, 82
 proportional sizing, 81
 StackPanel, mimicking, 83
GridView control, 219-220, 376

H

handedness of coordinate systems, 136-137
handling content overflow, 89-97
 clipping, 89-91
 scaling, 94-97
 scrolling, 91-94
Header property (ListView), 217
Height property, 48-50
Hidden elements, 51
Hidden property (ScrollViewer), 92
hijacking dependency properties, 393-394
HorizontalChildrenAlignment property
 (VariableSizedWrapGrid), 86
HorizontalContentAlignment property, 53-54
Hub control, 316-319
HubSections control, 319-322
HyperlinkButton control, 199-200

I

IBuffer, 228
Image element
 BitmapTransform class, 239-240
 decoding images, 236-244
 choosing a decoder, 236
 getting pixel data, 237-240
 reading BitmapProperties from a decoder,
 242-243
 reading metadata, 240-244
 reading raw metadata with metadata
 query language, 243-244
 encoding images, 244-252
 choosing encoding options, 245
 transcoding, 249-252
 writing pixel data, 246-247
 loading file variations
 loading automatically, 232-234
 loading manually, 234-235

- nine-grid, 225-228
- referencing files with URIs, 222-225
- resource packages, 235
- resource qualifiers, 233-234
- scaling, 231-232
- WriteableBitmap class, 228-231
- writing metadata, 247-248

- ImageBrush class, 124-128**
- ImageProperties, reading from a file, 240-241**
- immediate-mode graphics systems, 129-130**
- implicit Runs, 261**
- implicit styles, 385-386**
- incremental item rendering, 219**
- independent animations, 423-424**
- indirection, 381**
- inheritance, style inheritance, 383-385**
- Inlines property (TextBlock), 259-261**
- IntelliSense, 13**
- interactive zooming, 96-97**
- IRandomAccessStream, 224**
- IsColorFontEnabled property (TextBlock), 258-259**
- ItemHeight property (VariableSizedWrapGrid), 85**
- items controls, 207-208**
 - ComboBox control, 212-214
 - GridView control, 219-220
 - item containers, 209
 - items panels, 209-212
 - ListBox control, 214-215
 - ListView control, 215-219
 - data virtualization, 218
 - Header and Footer properties, 217
 - incremental item rendering, 219
 - reordering items, 217-218
 - ScrollIntoView, 216
 - selection, 217
- items panels, 209-212**
- ItemWidth property (VariableSizedWrapGrid), 86**

J-K

jumping to HubSections, 319-322

just-in-time compiler, 230

keyframes

- discrete keyframes, 438-439
- easing keyframes, 439
- linear keyframes, 435-436
- spline keyframes, 436-437

keywords

- XAML, 19-22
- x:Arguments keyword, 42-43
- x:FactoryMethod keyword, 43-44

L

layout, 47

2D transforms

- CompositeTransform, 61-62
- MatrixTransform, 62-63
- RotateTransform, 57-58
- ScaleTransform, 58-60
- SkewTransform, 60
- TransformGroup, 62
- TranslateTransform, 61

3D transforms, 63-66

arrange step, 450-451

measure step, 448-450

panels

- Canvas, 68-71
- DockPanel, 72-75
- Grid, 75-84
- SimpleCanvas panel, creating, 451-455
- SimpleStackPanel, creating, 455-458
- StackPanel, 71-72
- UniformGrid panel, creating, 458-463
- VariableSizedWrapGrid, 84-87
- WrapPanel, 84-87

- position, controlling
 - FlowDirection property, 54-55
 - HorizontalAlignment property, 52-53
 - VerticalAlignment property, 52-53
- size, controlling
 - Height property, 48-50
 - Margin property, 50-51
 - Padding property, 50-51
 - Visibility property, 51-52
 - Width property, 48-50

left-handed coordinate systems, 136-137

LengthConverter, 33

Lights, 157-164

- AmbientLight, 163-164
- DirectionalLight, 158-159
- fixed-function lighting, 172
- PointLight, 159-160
- SpotLight, 160-162

limiting text box input, 275

Line class, 101-102

linear keyframes, 435-436

LinearGradientBrush class, 116-123

ListBox control, 17-18, 214-215

lists, 17-18

ListView control, 215-219

- data virtualization, 218
- Header and Footer properties, 217
- incremental item rendering, 219
- reordering items, 217-218
- ScrollIntoView, 216
- selection, 217

live objects, writing to, 473-475

live video feeds, displaying, 296-298

loading

- file variations
 - loading automatically, 232-234
 - loading manually, 234-235
- XAML at runtime, 23-25

local files, 346-347

logic, avoiding in property wrappers, 338

logical resources, 348-355

- dynamic resources, 352
- resource lookup, 351-352
- static resources, 352
- theme resources, 352-355

LookDirection property (Cameras), 138-141

“lookless” appearance, creating for custom controls, 336-340

M

managed-to-unmanaged code transitions, performance impact of, 230

managing VisualState transitions, 399-404

manual animations, 444-446

Margin property, 50-51

markers for audio/video playback, 285

markup extensions, 38-40

- Binding, 357-360
 - in C#, 359-360
 - RelativeSource property, 359
- CustomResource, 354
- in procedural code, 40

Material, 165-172

- Color property, 168
- combining, 172
- DiffuseMaterial, 166-169
- EmissiveMaterial, 169-170
- SpecularMaterial, 170-172

Matrix3DProjection, 66

MatrixCamera, 147

MatrixTransform, 62-63

MatrixTransform3D class, 156

MaximumRowsOrColumns property (VariableSizedWrapGrid), 86

measure step (layout), 448-450

media capture

- CameraCaptureUI
 - capturing photos, 292-294
 - capturing video, 295-296

CaptureElement

- adjusting camera settings, 299-301
- capturing audio, 302-303
- capturing photos, 298-301
- capturing video, 301-302
- showing a preview, 296-298

media extensions, 281**media files**

- adding effects, 308
- changing format, 306-307
- reducing size of, 303-306
- trimming, 307-308

Media Foundation components, 281**Media Player, 288-292****media players**

- MediaElement, 286-288
- MediaPlayer, 288-292

MediaElement, 281-283

- compatibility with MIME types, 282
- events, 284-285
- as media player, 286-288
- playback
 - customizing, 283-284
 - effects, adding, 285-286
 - prolonged viewing, 287
- states, 284-285

MediaPlayer, playing custom media "formats, 289-292**mesh, winding, 174****MeshGeometry3D, 172-178**

- normals, 174-177
- Positions property, 172-174
- seams, removing, 175
- TextureCoordinates property, 177-178
- TriangleIndices property, 174-175

metadata

- reading, 240-244
- reading with metadata query language, 243-244
- writing, 247-248

metal-looking surfaces, creating, 171**methods**

- CanPlayType, 282
- factory methods, 43-44
- FindName, 25-26
- GetPixelDataAsync, 237-238
- GetThumbnailAsync, 240
- Visual3D.TransformToAncestor method, 191-195
- Visual3D.TransformToDescendant method, 191-195
- Visual.TransformToAncestor method, 187-191

MIME types, compatibility with MediaElement, 282**mimicking**

- Canvas with Grid, 83
- DockPanel with Grid, 84
- StackPanel with Grid, 83

Mode property (Binding), 361**Model3D, Lights, 157-164**

- AmbientLight, 163-164
- DirectionalLight, 158-159
- PointLight, 159-160
- SpotLight, 160-162
- unlit areas, troubleshooting, 161-162

Model3DGroup, 178-180**ModelVisual3D class, 181-182****MSIL (Microsoft Intermediate Language), 29****N****named elements in templates, 404****namespaces, 9-10**

- .NET, 35-37
- WPF, 10

naming XAML elements, 25-26**navigating views, 378****negative coordinates, 135****.NET, 6**

- clr-namespace directive, 35-36
- namespaces, 35-37

nine-grid, 225-228
node loops, 468-469
Nonzero filling, 110
normals, 174-177

O

object elements, 7-8

- children, 15-19
 - collection items, 17-19
 - content property, 16-17
- versus property elements, 12
- type converters, 33-34

OneTime binding, 361
OneWay binding, 361
open curly braces ({}), escaping, 39
Orientation property (VariableSizedWrapGrid), 85
origin of transforms, controlling, 57
OrthographicCamera. *See also* Cameras

- comparing with PerspectiveCamera, 144-147

overriding styles, 382

P

packaged files, 346
Padding property, 50-51
Panel class, 47
panels, 47

- Canvas, 68-71, 83
- DockPanel, 72-75, 84
- Grid, 75-84
 - absolute sizing, 80
 - autosizing, 80
 - child layout properties, 80
 - percentage sizing, 82
 - proportional sizing, 81
- items panels, 209-212
- layout
 - arrange step, 450-451
 - measure step, 448-450
- SelectiveScrollingGrid, 88
- SimpleCanvas, creating, 451-455
- SimpleStackPanel, creating, 455-458
- StackPanel, 71-72, 83
- TabPanel, 87
- ToolBarOverflowPanel, 87
- ToolBarPanel, 87
- ToolBarTray, 88
- UniformGrid, 88
- UniformGrid panel, creating, 458-463
- VariableSizedWrapGrid, 84-87
- WrapPanel, 84-87

PaneThemeTransition, 411-412
parent elements, communication with child elements

- arrange step, 450-451
- measure step, 448-450

parsing XAML,.XamlReader, 23-25
partial class definitions, 29
password reveal buttons, 279
PasswordBox control, 279
Path class, 104
PathFigure class, 108-111
PathSegment class, 108-111
percentage sizing, Grid panel, 82
performance

- cached composition, 130, 185
- scalability issues with vector graphics, 130

perspective transforms, 63-66
PerspectiveCamera. *See also* Cameras

- comparing with OrthographicCamera, 144-147

photos, capturing, 292-294, 298-301
pinch-to-zoom gestures, interactive zooming, 96-97
pixels

- BGRA8, 229
- retrieving data, 237-240
- writing pixel data, 246-247

- plain properties, data binding, 360**
- plane projections, 63-66**
- PlaneProjection class, 63**
- plastic-looking surfaces, creating, 171**
- autoplay (audio/video), 281-292**
 - customizing, 283-284
 - markers, 285
 - media content, 282-283
 - prolonged viewing, 287
- playing custom media formats, 289-292**
- PlayingCard control**
 - behavior, creating, 331-332
 - consuming, 335-336
 - “lookless” appearance, 336-340
 - user interface, creating, 332-335
- PointerDownThemeAnimation, 420**
- PointerUpThemeAnimation, 420**
- PointLight, 159-160**
- Polygon class, 103-104**
- Polyline class, 102-103**
- PopInThemeAnimation, 420**
- PopOutThemeAnimation, 420**
- Popup control, 314-316**
- PopupThemeTransition, 409-410**
- position, controlling, 52-55**
 - FlowDirection property, 54-55
 - HorizontalAlignment property, 52-53
 - VerticalAlignment property, 52-53
- Position property (Cameras), 136-137**
- Positions property (MeshGeometry3D), 172-174**
- power easing functions, 440-441**
- press-and-hold gesture, 217**
- primary XML namespace, 10**
- procedural code**
 - accessing binary resources, 345-346
 - markup extensions, 40
 - type converters, 33
- programmatically clicking a button, 198-199**
- ProgressBar control, 312**
- ProgressRing control, 324-325**
- projections**
 - Matrix3DProjection, 66
 - plane projections, 63-66

- properties**
 - attached properties, 68-69
 - defining in XAML2009, 44
 - dependency properties, hijacking, 393-394
 - Mode property (Binding), 361
 - plain properties, data binding, 360
 - processing order, 8
 - Selector class, 207
 - source properties, 358, 362-363
 - Stroke property (Shapes), 104-106
 - target properties, 358
 - Template property, setting inside a style, 394-395
 - Timeline properties, 427-430
- property attributes, 7-8**
- property elements, 11-13**
- property paths, 431-433**
- proportional sizing, 81**

Q-R

- quadratic Bézier curves, 108**
- RadialGradientBrush class, 123-124**
- RadioButton control, 202-203**
- Range controls**
 - DatePicker control, 322-323
 - Hub control, 316-319
 - HubSections control, 319-322
 - Popup control, 314-316
 - ProgressBar control, 312
 - ProgressRing control, 324-325
 - Slider control, 312-314
 - TimePicker control, 323-324
 - ToggleSwitch control, 325-326
- raw metadata, reading with metadata query language, 243-244**
- readers (XAML), 465-467**

reading

- BitmapProperties from a decoder, 242-243
- metadata, 240-244
- raw metadata with a metadata query language, 243-244
- XAML, 469-473

Rectangle class, 100-101**reducing media file size, 303-306****referencing**

- files with URIs, 222-225
- namespaces in XAML, .NET, 35-37

reflected light

- DiffuseMaterial, 166-167
- final reflected color, calculating, 168

relative positioning, 57**RelativeSource property (Binding), 359****removing seams in MeshGeometry3D, 175****rendering, controlling**

- data templates, 366-370
- value converters, 370-374

RenderSize property, 49**RenderTransformOrigin property, 57****ReorderThemeTransition, 415-416****RepeatBehavior property (Timeline), 428-429****RepeatButton control, 200****repeating animations, 428-429****RepositionThemeAnimation, 420****RepositionThemeTransition, 414-415****representing Geometries as strings, 113-115****resource lookup, 351-352****resource packages, 235****resource qualifiers, 233-234****resources**

- binary resources
 - accessing at the site of origin, 344-345
 - accessing from procedural code, 345-346
 - accessing in another assembly, 344
 - in WPF, 341-344
- logical resources, 348-355
 - dynamic resources, 352
 - resource lookup, 351-352
 - static resources, 352
 - theme resources, 352-355

responding to visual state changes, 396-399**restricting text box input, 275****retained-mode graphics systems, 129-130****reusable XAML-based graphics, 115****RichEditBox control, 276-279****RichTextBlock control, 265-270**

- embedding UIElements, 266-267
- text overflow, 267-270

right taps, 217**right-hand rule, 174****right-handed coordinate systems, 136-137****right-to-left environments, wrapping, 86****roaming files, 347****RotateTransform, 57-58****RotateTransform3D class, 153-156****Runs, 259****S****ScaleTransform, 58-60****ScaleTransform3D class, 150-153****scaling, 94-97**

- along nonprincipal axes, 153
- Image elements, 231-232

scrollbars, customizing, 91-93**scrolling, 91-94****ScrollIntoView, 216****ScrollViewer control, 91-94, 96-97****seams, removing in MeshGeometry3D, 175****SelectiveScrollingGrid panel, 88****Setters, 381-382****Shapes, 99-106**

- Ellipse, 101
- filling, 110
- Line, 101-102
- Path, 104
- Polygon, 103-104
- Polyline, 102-103
- Rectangle, 100-101
- Stroke property, 104-106

- sharing source properties with
 - DataContext, 362-363
- shortcuts, Geometry string commands, 114
- showing a preview, 296-298
- Silverlight, 5, 9
- SimpleCanvas panel, creating, 451-455
- SimpleStackPanel, creating, 455-458
- SineEase function, 443
- sizing, percentage sizing, 82
- SkewTransform, 60
- Slider control, 312-314
- SolidColorBrush class, 116
- source properties, 358, 362-363
- spaces in Geometry strings, 115
- SpecularMaterial, 170-172
- SpeedRatio property (Timeline), 428
- spline keyframes, 436-437
- SplitOpenThemeAnimation, 421
- SpotLight, 160-162
- StackPanel, mimicking with Grid, 83
- StackPanel class, 71-72
- star sizing, 81
- states (MediaElement), 284-285
- static fields, referencing, 37-38
- static resources, 352
- Storyboards, 417-419
 - with multiple animations, 430-431
 - tweaking with Timeline properties, 433-434
- Stretch alignment, 52-53
 - custom stretching with nine-grid, 225-228
 - interaction with ScaleTransform, 60
- StretchDirection property (Viewbox), 95
- strings
 - representing Geometries as, 113-115
 - type converters, 31-34
- Stroke property (Shapes), 104-106
- Style class, 380-386
- style selectors, 382
- styles, 380-386
 - applying, 382
 - implicit styles, 385-386
 - indirection, 381
 - inheritance, 383-385

- overriding, 382
 - Setters, 381-382
 - style selectors, 382
 - using base TargetType, 383
- SVG (Scalable Vector Graphics), 5
- SVG-to-XAML converters, 115
- SwipeBackThemeAnimation, 420
- SwipeHintThemeAnimation, 420
- syntax
 - property elements, 12
 - star syntax, 81
- System.Xaml
 - readers, 465-467
 - writers, 466

T

- TabPanel, 87
- target properties, 358
- Template property, setting inside a style, 394-395
- template selectors, 370
- templates
 - control templates, 386-394
 - hijacking existing properties, 393-394
 - respecting target control's properties, 388-393
 - Template property, setting inside a style, 394-395
 - tweaking, 395
 - visual states, 395-404
 - visual transitions, 399-404
- data templates, 366-370
- named elements, 404
- temporary files, 348
- text controls
 - PasswordBox control, 279
 - RichEditBox control, 276-279
 - RichTextBlock control, 265-270
 - embedding UIElements, 266-267
 - text overflow, 267-270

- TextBlock control, 253-265
 - Inlines property, 259-261
 - IsColorFontEnabled property, 258-259
 - text selection, 264-265
 - TextElement class, 261-264
 - TextLineBounds property, 256-258
 - TextReadingOrder property, 258
 - whitespace, 261
- TextBox control, 270-276
- TextBlock control, 253-265**
 - Inlines property, 259-261
 - IsColorFontEnabled property, 258-259
 - text selection, 264-265
 - TextElement class, 261-264
 - TextLineBounds property, 256-258
 - TextReadingOrder property, 258
 - whitespace, 261
- TextBox control, 270-276**
- TextElement class, 261-264**
- TextLineBounds property (TextBlock), 256-258**
- TextPointer class, 265**
- TextReadingOrder property (TextBlock), 258**
- texture coordinates in WPF, 178-179**
- TextureCoordinates property (MeshGeometry3D), 177-178**
- theme animations, 417-422**
 - To property, 425-427
 - From property, 425-427
 - Storyboards, 417-419
 - tweaking, 421-422
- theme resources, 352-355**
- theme transitions, 406-416**
 - AddDeleteThemeTransition, 412-414
 - applying to elements, 406-407
 - ContentThemeTransition, 410
 - EdgeUIThemeTransition, 410-411
 - EntranceThemeTransition, 408-409
 - PaneThemeTransition, 411-412
 - PopupThemeTransition, 409-410
 - ReorderThemeTransition, 415-416
 - RepositionThemeTransition, 414-415
- ThemeResource, 354**
- tile brushes**
 - DrawingBrush, 128-129
 - ImageBrush, 124-128
 - VisualBrush, 129
- time zones, DateTime data type, 241**
- Timeline properties, 427-430**
 - Storyboards, tweaking, 433-434
- TimePicker control, 323-324**
- TimeSpan property, 425**
- To property, 425-427**
- ToggleButton control, 201**
- ToggleSwitch control, 325-326**
- ToolBarOverflowPanel, 87**
- ToolBarPanel, 87**
- ToolBarTray panel, 88**
- ToolTip control, 203-205**
- transcoding, 249-252, 303-308**
 - adding effects, 308
 - changing media file format, 306-307
 - reducing media file size, 303-306
 - trimming, 307-308
- Transform3D, 147-156**
 - combining, 156
 - MatrixTransform3D class, 156
 - RotateTransform3D class, 153-156
 - ScaleTransform3D class, 150-153
 - TranslateTransform3D class, 149-150
- TransformGroup, 62**
- transforms**
 - 2D transforms, 55-63
 - RotateTransform, 57-58
 - ScaleTransform, 58-60
 - SkewTransform, 60
 - TranslateTransform, 61
 - 2D-to-3D coordinate system conversion
 - Visual3D.TransformToAncestor method, 191-195
 - Visual3D.TransformToDescendant method, 191-195
 - Visual.TransformToAncestor method, 187-191

3D transforms, 63-66

combining

CompositeTransform, 61-62

MatrixTransform, 62-63

TransformGroup, 62

origin of transforms, controlling, 57

Transform3D, 147-156

RotateTransform3D class, 153-156

ScaleTransform3D class, 150-153

TranslateTransform3D class, 149-150

TranslateTransform, 61

TranslateTransform3D class, 149-150

translucent DiffuseMaterial, creating, 168-169

TriangleIndices property (MeshGeometry3D), 174-175

triangles, vertices

normals, 174-177

winding, 173-174

trimming, 307-308

troubleshooting

unlit model areas, 161-162

winding, 174

tweaking

Storyboards with Timeline properties, 433-434

theme animations, 421-422

tweaking control templates, 395

TwoWay binding, 361

type converters, 31-34

custom type converters, 32-33

LengthConverter, 33

MatrixTransform, 63

in procedural code, 33

U

UIElement3D class, 182-184

UIElements

embedding, 266-267

MediaElement, 281-283

UNC (Universal Naming Convention), 343

underlining text, 260

UniformGrid panel, 88, 458-463

universal apps

binary resources, 346-348

image element, nine-grid, 225-228

unlit model areas, troubleshooting, 161-162

UpDirection property (Cameras), 141-144

URLs, referencing files with, 222-225

user controls

behavior, creating, 329

consuming, 330

user interface, creating, 328-329

user interface, creating

for custom controls, 332-335

for user controls, 328-329

using directive, 36-37

V

value converters, 370-374

VariableSizedWrapGrid panel, 84-87

vector graphics

versus bitmap-based graphics, 130

color brushes, 115-124

LinearGradientBrush, 116-123

RadialGradientBrush, 123-124

SolidColorBrush, 116

Geometries, 107-115

GeometryGroup, 111-113

PathFigure, 108-111

PathSegment, 108-111

performance scalability issues, 130

Shapes, 99-106

Ellipse, 101

filling, 110

Line, 101-102

Path, 104

Polygon, 103-104

Polyline, 102-103

Rectangle, 100-101

Stroke property, 104-106

- tile brushes
 - DrawingBrush, 128-129
 - ImageBrush, 124-128
 - VisualBrush, 129

VerticalChildrenAlignment property (VariableSizedWrapGrid), 86

VerticalContentAlignment property, 53-54

vertices

- normals, 174-177
- winding, 173-174

video

- capturing, 295-296, 301-302
- playback, 281-292
 - customizing, 283-284
 - effects, adding, 285-286
 - markers, 285
 - media content, 282-283
 - MediaPlayer, 288-292
 - prolonged viewing, 287

video stabilization, 285-286

view model, 368

Viewbox element, 94-97

Viewport2DVisual3D class, 184-186

Viewport3D, 186-187

views, 374-378

Visibility property, 51-52

Visible elements, 51

Visible property (ScrollViewer), 92

Visual State Manager, 395-404

visual states, 395-404

- responding to changes, 396-399
- visual transitions, 399-404

Visual3D

- ModelVisual3D class, 181-182
- UIElement3D class, 182-184
- Viewport2DVisual3D class, 184-186

Visual3D.TransformToAncestor method, 191-195

Visual3D.TransformToDescendant method, 191-195

VisualBrush class, 129

VisualState class, 396

VisualStateManager, 395-404

Visual.TransformToAncestor method, 187-191

vocabularies (XAML), 9

VSM, 395-404

W

WCF (Windows Communication Foundation), 5

WF (Windows Workflow Foundation), 5

whitespace in TextBlocks, 261

WIC (Windows Imaging Component), 243-244

Width property, 48-50

winding, 173-174

Windows Media, 281

Windows Media API, transcoding, 303-308

- adding effects, 308
- changing media file format, 306-307
- reducing media file size, 303-306
- trimming, 307-308

Windows Store

- binary resources, 346-348
- Image element
 - image data, rendering from
 - IRandomAccessStream, 224
 - nine-grid, 225-228
 - scale factors, 231-232

WinRT (Windows Runtime), 7, 35-37

world space, 135

WPF (Windows Presentation Foundation), 5

- 3D transforms, 63-66
- arrays, declaring, 37
- binary resources, 341-344
- coordinate systems, 135-136
- Drawing class, 107
- logical resources
 - dynamic resources, 352
 - resource lookup, 351-352
 - static resources, 352
 - theme resources, 352-355
- namespaces, 10
- SelectiveScrollingGrid panel, 88
- static fields, referencing, 37-38

TabPanel, 87
 texture coordinates, 178-179
 ToolBarOverflowPanel, 87
 ToolBarPanel, 87
 ToolBarTray panel, 88
 UniformGrid panel, 88
 Visibility property, 51-52
 XamlReader, 25

WrapPanel, 84-87

WriteableBitmap class, 228-231

writers (XAML), 466

writing

ImageProperties to a file, 240-241
 to live objects, 473-475
 metadata, 247-248
 node loops, 468-469
 pixel data, 246-247
 to XML, 475-476

X

XAML, 5-6

attributes, 19
 “code-inside” feature, 30
 compiling, 26-30
 BAML, 28
 code-behind file, 27
 generated source code, 29-30
 elements, 6-9
 naming, 25-26
 type converters, 33-34
 event handlers, processing order, 8
 extending
 markup extensions, 38-40
 type converters, 31-34
 keywords, 19-22
 namespaces, 9-10
 .NET, 35-37
 WinRT, 35-37
 node loops, 468-469
 object elements, 7-8, 15-19

parsing at runtime, 23-25
 properties, processing order, 8
 property elements, 11-13
 reading, 469-473
 reusable XAML-based graphics, 115
 xml:lang attribute, 19
 xml:space attribute, 19

XAML2009, 40-41

built-in System data types, 42
 constructors, 42-43
 dictionary keys, 41-42
 event handlers, 44
 factory methods, 43-44
 full generics support, 41
 parser, 42
 properties, defining, 44
 x:Arguments keyword, 42-43

XamlReader, 23-25

XamlServices class, 476-478

XamlXmlReader, 467

x:Arguments keyword, 42-43

x-axis, 135, 136-137

XBF (XAML Binary Format), 28

x:FactoryMethod keyword, 43-44

XML, writing to, 475-476

xml:lang attribute, 19

XmlnsDefinitionAttribute, 36

xml:space attribute, 19

Y-Z

y-axis, 135-137

Z order, 70

z-axis, 135-137

Z-fighting, 139

zooming in, interactive zooming, 96-97