WPF Architecture

Brian Noyes
www.idesign.net

About Brian

- Chief Architect, IDesign Inc. (www.idesign.net)
- Microsoft Regional Director (http://www.microsoftregionaldirectors.com/) / MVP
- Publishing
  - Smart Client Deployment with ClickOnce, Addison Wesley, January 2007
  - Data Binding in Windows Forms 2.0, Addison Wesley, January 2006
- Speaking
  - Microsoft TechEd US, Europe, Malaysia, Visual Studio Connections, DevTeach, INETA Speakers Bureau, MSDN Webcasts
- Participates in Microsoft Design Reviews
- E-mail: brian.noyes@idesign.net
- Blog: http://briannoyes.net
Architecture in the Presentation Layer

- Is there “architecture” inside the presentation layer?
  - “The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.”

  [From the book Software Architecture in Practice (2nd edition), (Bass, Clements, Kazman; Addison-Wesley 2003.]

- WPF has a rich and extensible architecture for designing compelling UI applications
- Your WPF applications need to have a planned architecture for the elements you will use to compose them
Agenda

- Architecture context
- WPF Overview
- WPF Application Architecture
- WPF Framework Architecture Constructs

WPF Overview

What is it?

- New Windows UI Framework
  - Alternative to Windows Forms / ASP.NET?
- Runtime capabilities part of .NET 3.0
- Released in November 2006
  - No released design tools at the time
WPF Features

Why do you need it?

- State of the art graphics
  - DirectX under the covers
  - Controls, 2D, 3D graphics
  - Rich Media Integration (images, audio, video)
  - Animation
  - Visual Styling
- Takes advantage of modern video cards and GPUs
  - Enhance application graphics and performance

WPF Features

What is different about it?

- New rendering engine
  - Retained mode compositing model
    - Drawn visuals are cached
  - Vector graphics based
  - Logical pixels
  - Transforms, overlays, opacity
  - Very different from User32 and GDI32
  - DirectX under the covers
  - Takes advantage of GPU for graphics processing

Copyright © 2006 Brian Noyes, IDesign Inc
WPF Rendering Architecture

- Your WPF Application Code (UI Thread)
- Presentation Framework
- Presentation Core
- CLR
- Milcore.dll (Rendering Thread)
- User32
- DirectX

WPF 3D vs DirectX

- WPF 3D focused on integrating 3D elements with 2D elements and window constructs
  - Higher level abstractions
  - Not as high perf as DirectX raw
WPF Features
What is different about it?

- New programming language - XAML
  - eXtensible Application Markup Language
  - Not specific to WPF
  - Allows better separation of design from implementation
    - Separate design team from development team
    - Key design goal of WPF

XAML Language

- Declarative object syntax
- Use to define the static structure and configuration of an object hierarchy
  - Such as window layout
- Not specific to WPF
  - Used by WF as well
- Easier to write development tools
- Relatively easy to edit and understand
- Expresses object hierarchies in a more compact form
- Objects must have default constructor
- Requires type conversion
XAML Constructs

```xml
<Window x:Class="XAMLConstructs.Window2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="XAMLConstructs" Height="300" Width="300">
  <x:Code>
    <![CDATA[
    void OnClick(object sender, RoutedEventArgs e) {
      MessageBox.Show("Hello from inline code");
    }
    ]]>
  </x:Code>
  <Grid Grid.ColumnDefinitions="<ColumnDefinition />
        <ColumnDefinition />
  " RowDefinitions="<RowDefinition />
             <RowDefinition />
  ">
    <Button Grid.Column="0" Grid.Row="0" Content="{Binding Path=Name}"/>
  </Grid>
</Window>
```

WPF Programming Models

- Code only
- XAML only
- XAML + Code
### WPF Programming Models

**Code Only**
- WPF objects are all defined as .NET classes
- Can instantiate using a programming model similar to Windows Forms
- Instructive for understanding WPF, not practical for real world development
  - Not productive
  - Design tools will all be based on XAML markup
  - Element hierarchy can be more compactly and cleanly represented in XAML

**XAML only**
- Can load raw XAML “pages” into Internet Explorer
  - No design time compilation required
  - Limited set of functionality
  - Static content + data binding + animations
  - No code behind or script allowed
  - Real applications require event handling, data access, calling services, accessing files, etc., which require programmatic code
- Can embed C# or VB code in script blocks within XAML
  - Avoid
  - Must compile into assembly
  - Messy for maintenance
  - Lose IntelliSense, color coding, compile time checking
WPF Programming Models

XAML + Code
- Sweet spot
  - XAML very expressive for static layout of UI and initial configuration of properties
    - “Human readable”
    - Easier for tools to parse
  - Programmatic code needed for dynamic behavior of applications
    - Hooks in to XAML similar to ASP.NET

WPF Compilation

XAML
- Compiled to BAML + procedural code
- Static element hierarchy and property setting in BAML
  - Runtime execution to create object hierarchy
- Inline code and event subscriptions in .g.cs (or .g.vb) file
  - Compile time merging as partial class with code behind

Code
- Normal .NET compilation process
  - Metadata + MSIL

Result
- EXE or DLL with metadata + BAML + IL
WPF Compilation

- Dynamic loading of XAML
  - XamlReader
  - Get back loaded object (Window, Page, etc.)
  - Dynamic compilation

WPF Limitations

- Design tools somewhat immature
  - Even when Orcas releases
  - Lacking features that Windows Forms and ASP.NET form designers have

- Missing controls
  - A number of controls that are standard in Windows Forms and ASP.NET have no equivalent yet in WPF
  - 3rd Party vendor market filling the void rapidly

- Needs high end graphics hardware to take advantage of its strengths
WPF Tools

- Visual Studio 2005 with Extensions
  - Interim solution
- Visual Studio 2008
- Expression Blend
- Expression Design

WPF Assemblies and Namespaces

- Primary assemblies
  - PresentationCore
  - PresentationFramework
  - WindowsBase
- Namespaces
  - System.Windows
    - All child namespaces except System.Windows.Forms
WPF Deployment Models

- **XAML Pages**
  - Xcopy deployment to host site
- **WPF Windows Applications**
  - ClickOnce deployment
  - Windows Installer (msi) deployment
- **XAML Browser Applications**
  - Accessed via the browser
  - ClickOnce behind the scenes
- All require .NET 3.0 installed on the client machine

What about Silverlight?

- Really a different technology than WPF
- Lots of overlaps from a programming perspective
  - Harness .NET / WPF development skills
  - Subset of WPF elements
  - Same XAML syntax / constructs
  - Different execution / rendering engine
- Can be thought of as an alternative deployment model for WPF apps
  - With restricted execution environment and object model limitations accounted for
Agenda

- Architecture context
- WPF Overview
- WPF Application Architecture
- WPF Framework Architecture Constructs

WPF Application Architecture

- Application
- Window or Page
  - Canvas for Silverlight
- Controls
- FrameworkElements
- Visuals
Application

- Starts the execution environment for the application
- Centralized message pump for Windows
- Application scoped resources and properties
- Unhandled exceptions
- Command line parameters and exit codes
- Navigation in XAML Browser Apps
- StartupUri property
  - Point to main window XAML
- Startup event

Window

- Top level window construct
- Standard items (frame, system menu, minimize, maximize, icon, title, etc.)
- InitializeComponent method in constructor required with XAML definition
- Client area for content
**Page**

- Top level construct for navigation style application
  - Equivalent to Window for many things
- Can be hosted in a NavigationWindow, Frame, or browser
- Navigation
  - Uri
  - Object instance (Page)
  - Hyperlink
  - History
- Application NavigationService manages current page and navigation path

**Visuals**

- Lowest level graphics display elements
- Base class for UIElement, FrameworkElement, Control
- Used for best performance drawings
FrameworkElements

- Adds support for common user interactive features:
  - Input focus
  - Layout
  - Styles
  - Data binding
  - Resources
  - Routed Events

Controls

- Adds common properties for user interface controls
  - Foreground
  - Background
  - Font properties
- Adds support for control templates
Agenda

- Architecture context
- WPF Overview
- WPF Application Architecture
- WPF Framework Architecture Constructs

Logical Tree

- Represents the element hierarchy of the application
- Based on parent-child object hierarchy
  - Mimics XAML declarations
- Nodes can be visuals or non-visual objects
  - String, geometry, storyboard, etc.
- Use for routed event bubbling and property inheritance
Visual Tree

- Just the visual elements that something is composed
- Not used as often as logical tree
- Used for custom control templates
- Replicated to the render thread for async rendering
  - Only visible elements
  - Don’t replicate images and some data, share for efficiency

Dependency Properties

- Add semantics to basic .NET properties
- Enabled by DependencyObject base class
  - Defines SetValue/GetValue methods
- Numerous uses
  - Memory usage reduction
  - Attached
  - Data binding
  - Metadata
  - Animations
  - Notifications
  - Validation
  - Coersion
  - Default values
  - Value provider hierarchy
  - Property inheritance
Freezables

- Special kind of object
  - Freezable base class
- Most drawing constructs in WPF derive from
  - Geometries, Models, Brushes, Paths, etc.
  - Non-visual elements, support data structures
- Two states: frozen and unfrozen
- Allows sharing of objects across threads
- Improves performance in rendering graphics and animation
  - Synchronization locks not needed
- Frozen objects cannot be modified
  - InvalidOperationException
  - Can clone to get a modifiable copy

Routed Events

- Adds semantics to events similar to how dependency properties add semantics to properties
- Based on RoutedEventHandler
  - RoutedEventArgs or derived type
- Bubbling
  - Start at the element and bubble out to containing elements
- Tunneling
  - Start at the root of the element tree and tunnel down to the actual element
  - Prefixed with PreviewXXX
- Direct
  - Single target handler
Routed Commands

- Ties operation to multiple elements that trigger the operation
- Similar to MFC commands
- Set command on elements
- Set command handlers in the containing hierarchy

Triggers

- New construct for dynamic behaviors
  - Could just be done through event handling
  - Allows behaviors to be tied to elements declaratively
    - Through XAML
  - Facilitates designers defining triggers

- Three kinds:
  - Property
  - Data
  - Event
Triggers

- Property
  - Invoked when dependency property changes
  - Executes a collection of property setters when a specific property is set to a specific value

- Data
  - Invoked when normal .NET property changes
  - Executes a collection of property setters when a specific property is set to a specific value
  - Can only set dependency property values inside the trigger
  - Use data binding syntax to specify the property that triggers

Triggers

- Event
  - Invoked when routed event is fired
  - Used primarily with animations
  - Contains Actions collection
    - Storyboards
    - Sounds
Data Binding

- Rich support for data binding in WPF
- Decouples UI data binding from source objects
- Explicit control when needed
- Several pieces
  - Data sources – provided through DataContext
  - Property setting – provided by Bindings
  - Context management – provided by collection views

Resources

- Any reusable chunk of information
  - Data / media files
  - Element hierarchy
- Two kinds
  - Binary
  - Logical
**Styles**

- Groups together property values to be set as a group, instead of individually
- Includes non-visual properties
- Can drive dynamic behaviors of the application
  - Combine with triggers
- Contains property setters
- Can be shared across many elements
- Defined through Resources
- Similar capabilities to CSS for the web

**Templates**

- Control Template
  - Defines a new visual element hierarchy for a control
  - Allows you to specialize the presentation of a control without needing to derive a class
  - Important capability since most WPF controls are sealed!
- Data template
  - Defines a reusable element hierarchy
  - Used with ItemsControls
  - Cloned once for each item in a control collection
Themes

- Collections of styles and templates
- Applied to create a consistent user interface throughout the application
- Defined through resource dictionaries
- Can be internal (in the assembly) or external (another assembly)
- Several themes built-in
  - Default selected based on operating system by WPF
- Can create custom themes

2D Graphics

- Vector and raster graphics primitives
- Drawings
  - Data structures that describe what needs to be drawn
  - No rendering behavior of their own
- Visuals
  - Low level infrastructure for rendering anything onto the screen
- Shapes
  - Derive from FrameworkElement – more heavyweight
  - Easier to use, more features built in
2D Graphics

- Brushes
  - Main primitive for specifying fills of visuals
  - Colors – solid or gradient
  - Tiles – Drawing, Image, or Visual

- Pens
  - Special case of Brush
  - Used for lines, borders
  - Specified as a Brush instance

2D Graphics

- Images
  - Many image formats supported
  - Defined in terms of an ImageSource
  - BitmapImage most common type used to represent raster image format
    - Derived from ImageSource

- Image Processing
  - ImageSource pipeline
  - Transforms
  - Effects
3D Graphics

- Coordinate System
  - Right hand rule
    - X axis horizontal – increasing to right
    - Y axis vertical – increasing up
    - Z axis out of page – increasing towards you
  - Axes can be transformed based on camera

- Cameras
  - Determine the rendering of a model in 2 space (the screen)
  - Orientation: Position, LookDirection, UpDirection
  - Projection: Perspective and Orthographic

3D Graphics

- Transform3Ds
  - Rotate, Scale, Translate
  - Matrix
  - Group

- Model3Ds
  - Lights
  - Geometries

- Visual3Ds
  - Visual tree element for rendering 3D models

- Viewport3D
  - FrameworkElement for rendering 3D model into the 2D visual tree
Animations

- Time-based modification of element properties
- Properties being animated must be dependency properties
- Several forms of animation
  - From/To/By
  - KeyFrame
  - Path
- Built in animation types
  - Double
  - Color
  - Point

Media

- MediaElement
  - Primary control for rendering video or playing audio
  - Supports XAML usage, layout system, etc. like any other FrameworkElement
  - Source, Play, Pause, Stop, Position, SpeedRatio, Volume control through properties and methods
- MediaPlayer
  - Ties into drawing subsystem
  - Cannot be used from XAML
Documents

- Two kinds
  - Fixed
    - \(\ddot{y}\) Renders the same regardless of resolution
    - \(\ddot{y}\) DocumentViewer control
  - Flow
    - \(\ddot{y}\) Optimizes rendering based on resolution for readability

Ink

- Tablet PC input
- InkCanvas element provides an input surface
- Collect strokes as StrokeCollection
  - Each stroke contains collection of StylusPoint objects
    - \(\ddot{y}\) StylusPoint has X, Y, Pressure properties
- Can overlay other controls on the InkCanvas
- Can place InkCanvas inside other content controls
- Collected stroke information can be applied to other controls
- InkAnalyzer provides handwriting recognition engine
  - Recognizes gestures as well
Interop

Windows Forms
- Easily host Windows Forms control in WPF app
- Easily host WPF control in Windows Forms app

Win32
- Bi-directional support as well
- Slightly more complicated when dealing with threading and message passing

Resources

- .NET Rocks! TV: http://dnrtv.com (episodes 56, 59)

E-mail: brian.noyes@idesign.net
Blog: http://briannoyes.net